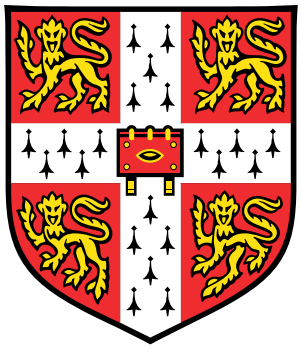
Representation, learning, description and criticism of probabilistic models with applications to networks, functions and relational data



James Robert Lloyd

University of Cambridge

This dissertation is submitted for the degree of

Doctor of Philosophy

Trinity College

December 2014

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other University. 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 dissertation contains less than 65,000 words and has less than 150 figures.

James Robert Lloyd
December 2014

Acknowledgements

First and foremost I would like to thank my supervisor, Professor Zoubin Ghahramani. He has been very supportive whilst also giving me the freedom to let my mind wander. Working with Zoubin has made PhD life quite the antithesis of the vision presented in PhD comics and other derogatory depictions.

Secondly I would like to thank all of my collaborators. Special thanks go to Peter Orbanz and Daniel Roy for helping me find and co-supervising my first project and subsequent investigation. I am also very glad to have worked with David Duvenaud on my later projects; our contrasting styles of working and writing were a great source of synergy. I would also like to thank Roger Grosse for showing me a higher standard of research programming, Joshua Tenenbaum for his infectious enthusiasm and José Miguel Hernández-Lobato for his unequaled diligence.

I would also like to thank everyone at the computational and biological learning group for creating such an enjoyable working environment. Special mentions go to Carl Rasmussen for sharing his knowledge of Gaussian processes, Alex Matthews for joining me in discussions on the philosophy of statistics, Yarin Gal for an endless source of (productive) contrarianism and Alex Davies for teaching me the ways of data mining (AHRF) and for being a most excellent friend.

Finally, I would like to thank all of the other friends I made during my studies. In particular I would like to thank Alexa Pohl and Matthew Griffiths for keeping conversation scholarly longer than most, Will Sonnex for valiant efforts to explain the elegance of type theory, Alisa Logvinenko for her unbounded curiosity and Rosie Lintott and Mike Collins for reminding me that there is more to life than mathematics.

Abstract

This thesis makes contributions to a variety of aspects of probabilistic inference. When performing probabilistic inference, one must first represent one's beliefs with a probability distribution. Specifying the details of a probability distribution can be a difficult task in many situations, but when expressing beliefs about complex data structures it may not even be apparent what form such a distribution should take. This thesis starts by demonstrating how representation theorems due to Aldous, Hoover and Kallenberg can be used to specify appropriate models for data in the form of networks. These theorems are then extended in order to reveal appropriate probability distributions for arbitrary relational data or databases.

A simpler data structure to specify probability distributions for is that of functions; many probability distributions for functions have been used for centuries. We demonstrate that many of these distributions can be expressed in a common language of Gaussian process kernels constructed from a few base elements and operators. The structure of this language allows for the effective automatic construction of probabilistic models for functions. Furthermore, the formal mathematical language of kernels can be mapped neatly onto natural language allowing for automatic descriptions of the automatically constructed models.

By further automating the construction of statistical models, the need to be able to effectively check or criticise these models becomes greater. This thesis demonstrates how kernel two sample tests can be used to demonstrate where a probabilistic model most disagrees with data allowing for targeted improvements to the model. In proposing a new method of model criticism this thesis also briefly discusses the philosophy of model criticism within the context of probabilistic inference.

Contents

List of Figures

List of Tables

Notation

Distribution notation is often abused when specifying complex joint distributions. For example

$$\begin{aligned} X &\sim \mathcal{N}(0,1) \\ Y &\sim \mathcal{N}(0,1) \end{aligned}$$

would often be read as X and Y being *independent* standard normals. But, for the sake of pedantry, this statement has not actually specified anything about the joint distribution; $X = Y$ is consistent with the statements above. Since this could become confusing in longer statements, I will write probability distributions as imperative programs i.e. each line indicates how a random variable is to be generated and following each line in order specifies a single sample from the joint distribution. This means that each statement is a conditional distribution, conditioned on all previously defined random variables. To make this clear, I will use the following notation

$$\begin{aligned} X &\rightsquigarrow \mathcal{N}(0,1) \\ Y &\rightsquigarrow \mathcal{N}(0,1) \end{aligned}$$

where the symbol $\rightsquigarrow$ has been chosen since it is similar both to the distribution symbol $\sim$ and programmatic assignment $\leftarrow$[1].

I will use curly braces $\{\}$ to indicate sets and parentheses $()$ to indicate lists. The index set of lists or sets will occasionally be omitted when clear from context e.g. (U_i) might be short hand for $(U_i)_{i \in \mathbb{N}}$.

[1] Ideally I would have used the unicode character 'LEFTWARDS WAVE ARROW' (U+219C) but this does not appear to be in any standard LaTeX packages that do not interfere with my current document setup.

Chapter 1

Introduction

This thesis concerns the challenge of drawing inferences from uncertain data. We take a probabilistic and Bayesian approach throughout which is a natural extension of drawing logical inferences from certain information (e.g. Jaynes, 2003). To perform probabilistic inference faithfully one must be able to express all of one's prior information about some situation in the form of a probability distribution. In chapters 2 and 3 we demonstrate how to construct probability distributions that respect symmetry conditions appropriate for data in the form of networks or databases. In chapters 4 and 5 we develop expressive probability distributions over functions and demonstrate that they express patterns that can be easily understood by humans. In chapter 6 we demonstrate the application of these distributions over functions to a data mining competition. Finally, chapter 7 considers the problem of criticising a probabilistic model when we have reason to believe that we were either unable to faithfully express our prior information probabilistically or perform inference to a sufficient level of accuracy. We elaborate on these points in the rest of this chapter.

1.1 Bayesians and frequentists

All chapters of this thesis apart from chapters 6 and 7 take an entirely (or at least approximately) Bayesian (or more generally probabilistic) approach to statistical inference. The situations in which it is wise to perform inference in a Bayesian manner as opposed to using a method justified by its frequentist properties are commonly poorly understood. As a result, many people still talk about a Bayesians versus frequentists debate as if one paradigm could be declared better than the other[1]. I will therefore devote a small amount of space to the justification of Bayesian inference and also discuss

its limitations.

How should we perform rational inferences with uncertain information? A solution to this question is provided by Cox's theorem (Cox, 1946) which derives a calculus of reasoning about the plausibility of statements from three simple axioms of rational behaviour. A version of these axioms are colloquially stated by Jaynes (2003) as

1. Representation of degrees of plausibility by real numbers
2. Qualitative correspondence with common sense
3. Consistency [of reasoning]

The calculus that one can logically derive from these axioms is isomorphic to probability theory; plausibilities satisfy the sum and product rules that are typically stated as axioms of probability theory.

This result tells us that if we can accurately specify our beliefs as probability distributions then there is only one way to rationally (in the sense of Cox's axioms) update those beliefs in the light of new data i.e. by using probability theory (which includes Bayes' rule). Thus, we are compelled to perform inference using the rules of probability theory when the following three conditions hold

1. We are willing to accept Cox's axioms
2. We can faithfully express our prior beliefs about a problem using probability distributions
3. Exact probabilistic inference or a sufficiently close approximation is computationally tractable

There are some who have argued against Cox's axioms (see comments in Jaynes (2003)) but no widely accepted alternatives currently exist. Conditions 2 and 3 however are often false; in response one may wish to develop or use a method with frequentist guarantees or one could attempt research which makes them true in a larger number of situations[2]. This thesis is concerned with the latter strategy.

[1] See for example a recent blog post on the subject `http://lesswrong.com/lw/jne/a_fervent_defense_of_frequentist_statistics/` which caused some ill-informed debate around the internet e.g. `http://www.reddit.com/r/statistics/comments/1ye692/a_fervent_defense_of_frequentist_statistics/` and `https://news.ycombinator.com/item?id=7263490`

[2] Or one could attempt to revise statistical theory in such a way as to take account of computational and epistemic limitations. These lines of enquiry are potentially very fruitful but still in their infancy (e.g. Jordan, 2013).

1.2 An introduction to exchangeability and representation theorems

Suppose we receive a sequence of data $(X_1, X_2, \dots)$ and we wish to specify our beliefs about this data with a probability distribution. A commonly appropriate prior assumption is that we do not believe the order of the data contains any information, or rather that any probability distribution encoding our beliefs about this data should be invariant to reorderings of the data. That is, we may often believe that our data is an exchangeable sequence:

$$(X_1, X_2, \dots) \stackrel{\mathrm{d}}{=} (X_{p(1)}, X_{p(2)}, \dots) \qquad \forall\, p \in \mathbb{S}_\infty \tag{1.2.1}$$

where $\stackrel{\mathrm{d}}{=}$ denotes equality in distribution, and $\mathbb{S}_\infty$ is the set of all permutations of the natural numbers, $\mathbb{N}$, which permute at most a finite number of elements.

de Finetti's theorem (e.g. Kallenberg, 2005) characterises all probability distributions of sequences with this property; $(X_i)_{i\in\mathbb{N}}$ is an exchangeable sequence if and only if there exists a random probability measure Θ such that $X_1, X_2, \dots \stackrel{\mathrm{iid}}{\sim} \Theta$. In other words, observations are conditionally i.i.d. given some random Θ. For the purposes of probabilistic modelling, this implies that our beliefs about exchangeable sequences of data can be expressed in the form of a distribution over probability measures.

In chapter 2 we demonstrate that a generalisation of de Finetti's theorem due to Aldous and Hoover can be used to construct probability distributions of networks and arrays where weaker symmetry assumptions are appropriate. In chapter 3 we use results by Kallenberg to extend the results of Aldous and Hoover to the case of probability distributions for general databases or arbitrary relational data. We also demonstrate how these representation theorems are compatible with other formalisms and assumptions e.g. longitudinal / time-varying data or conditional probability distributions.

1.3 An introduction to using Gaussian processes to model functions

There are a great many (and many great) introductions to Gaussian processes; the most comprehensive and relevant to this thesis at the moment is Rasmussen and Williams (2006). Rather than repeat them, I will give an intuitive derivation of Gaussian processes starting from linear regression. Those with an understanding of Gaussian processes may

safely skip this section.

Let us consider a simple version of Bayesian linear regression. Suppose we have a data set D consisting of real-valued (input, output) pairs denoted by $D = ((x_i, y_i) \in \mathbb{R}^2 : i = 1, \ldots, n)$. A simple probabilistic linear regression model of this data could be of the form

$$m \rightsquigarrow \mathcal{N}(0, 1) \tag{1.3.1}$$

$$\varepsilon_i \overset{\text{iid}}{\rightsquigarrow} \mathcal{N}(0, \sigma_\varepsilon^2) \tag{1.3.2}$$

$$y_i \leftarrow mx_i + \varepsilon_i \tag{1.3.3}$$

i.e. we believe our data will be linearly related by a line passing through the origin.

For this simple model it is trivial to update our beliefs in the light of data according to the calculus of probability theory upon which we find

$$m \,|\, D \sim \mathcal{N}\left(\frac{\sum_i x_i y_i}{\sum_i x_i^2 + \sigma_\varepsilon^2}, \frac{\sigma_\varepsilon^2}{\sum_i x_i^2 + \sigma_\varepsilon^2}\right) \tag{1.3.4}$$

$$y_j \,|\, D \sim \mathcal{N}\left(\frac{x_j \sum_i x_i y_i}{\sum_i x_i^2 + \sigma_\varepsilon^2}, \frac{x_j^2 \sigma_\varepsilon^2}{\sum_i x_i^2 + \sigma_\varepsilon^2} + \sigma_\varepsilon^2\right) \quad \text{where } j \notin \{1, \ldots, n\}. \tag{1.3.5}$$

The updates of the prior belief for m (1.3.1) into the posterior belief (1.3.4) is demonstrated visually in figure 1.1; as more data is observed one's belief about the parameter m becomes more concentrated.

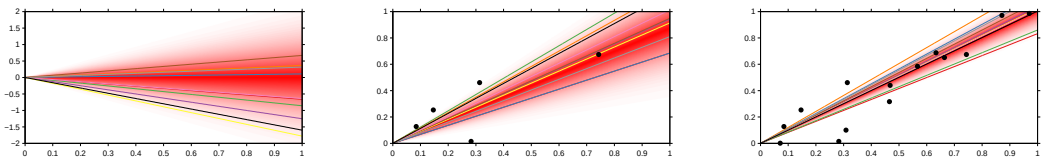


Figure 1.1: Bayesian linear regression. *Left*: Visualisation of prior distribution on gradient m. Lines are random samples from the prior probability distribution, the red shading is a representation of the probability distribution. *Middle*: Visualisation of the posterior distribution for the gradient m after 5 samples of data have been observed. *Right*: The posterior after 15 observations.

By integrating out m, we can rewrite the equations for the noisy linear relationship

between $y = (y_i)$ and $x = (x_i)$ in the following equivalent form

$$y_i \sim \mathcal{N}(0, x_i^2 + \sigma_\varepsilon^2) \tag{1.3.6}$$

$$\mathrm{Cov}(y_i, y_j) = x_i x_j \quad \forall\, i \neq j \tag{1.3.7}$$

i.e. the prior distribution for y has been assumed to be a multivariate Gaussian distribution

$$y \sim \mathcal{N}(0, K) \tag{1.3.8}$$

where $K_{ij} = x_i x_j + \delta_{ij}\sigma_\varepsilon^2$ and δ_{ij} is the Kronecker delta. Note that we can write this equation for the covariance as a function of pairs of inputs

$$K_{ij} = k(x_i, x_j) = x_i x_j + \delta_{x_i x_j}\sigma_\varepsilon^2 \tag{1.3.9}$$

and this relationship holds for datasets of arbitrary finite size[3]. We now may reasonably ask whether or not we could define a probability distribution over y when x is infinite e.g. when $x = \mathbb{R}$. The answer to this question is yes.

Definition 1.3.1 (Gaussian process)**.** A Gaussian process (GP) is a collection of random variables, any finite subset of which have a joint Gaussian distribution.

To specify a Gaussian process we must specify the mean and covariance of any finite subset of random variables which we achieve with a mean function $\mu : \mathcal{X} \to \mathbb{R}$ and a covariance function or kernel $k : \mathcal{X}^2 \to \mathbb{R}$ where $\mathcal{X}$ is the space inhabited by the inputs x which can be written

$$y_i \sim \mathcal{N}(\mu(x_i), k(x_i, x_i)) \tag{1.3.10}$$

$$\mathrm{Cov}(y_i, y_j) = k(x_i, x_j) \tag{1.3.11}$$

or alternatively

$$(y_i : i = 1, \ldots, n) \sim \mathcal{N}\Big(\big(\mu(x_i) : i = 1, \ldots, n\big), K\Big) \tag{1.3.12}$$

where $K_{ij} = k(x_i, x_j)$; note that the kernel, k, must be such that the matrix K is always positive (semi) definite to define a valid probability distribution. Since these

[3]We are implicitly assuming that the x_i are disjoint. This can be relaxed without any conceptual difficulty.

equations hold for any value of the inputs x we can see that by writing $y_i = f(x_i)$ we have essentially specified a probability distribution over functions f. We denote this distribution as

$$f \sim \mathrm{GP}(\mu, k). \tag{1.3.13}$$

It is now reasonable to enquire as to the properties of this probability distribution over functions as we change μ and k. The role of the mean function μ is particularly easy to characterise; it specifies the pointwise mean of this distribution over functions. The covariance or kernel k is more interesting. Random samples from Gaussian processes with a zero mean function but different kernels are shown in figure 1.2; the randomly sampled functions display qualitatively different patterns of linearity, smoothness and periodicity.

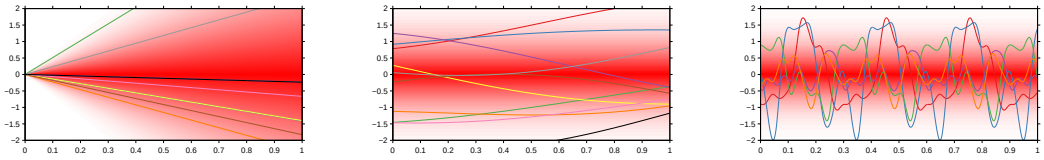


Figure 1.2: Random samples from zero mean Gaussian processes with different covariance functions. Lines are random samples from the Gaussian process distributions, the red shading is a representation of the pointwise probability distribution. *Left*: Linear functions. *Middle*: Smooth and slowly varying functions. *Right*: Smooth but exactly periodic functions.

The distribution over linear functions was constructed using a covariance of the form

$$k(x_i, x_j) = x_i x_j \tag{1.3.14}$$

which is the same as that in equation (1.3.9) with the noise variance equal to zero.

The distribution over smooth functions was constructed with a kernel of the form

$$k(x_i, x_j) = \exp(-\alpha |x_i - x_j|^2) \tag{1.3.15}$$

where α is a constant. When the distance between x_i and x_j is small, the covariance (and correlation) between $f(x_i)$ and $f(x_j)$ will be approximately equal to 1. As a result, nearby values of the function will be similar to each other. As the distance $|x_i - x_j|$ increases the correlation monotonically decreases to zero; the exact way in which the

correlation decreases determines the characteristic smoothness one observes in figure 1.2.

Finally, the distribution over periodic functions was constructed using a covariance of the form

$$k(x_i, x_j) = \exp\left(-\frac{2\sin^2(\pi(x_i - x_j)/p)}{\ell^2}\right) \tag{1.3.16}$$

where ℓ and p are constants. Whenever $|x_i - x_j|$ is an integer multiple of p the correlation is equal to 1 resulting in exact periodicity of the random functions.

In chapter 4 we demonstrate how to learn an appropriate kernel for a particular dataset by constructing new kernels from old. In chapter 5 we demonstrate that by carefully choosing which kernels we compose with each other we can always automatically produce natural language descriptions of the probability distributions they encode for. In chapter 6 we demonstrate how these ideas were applied to a data mining competition and where this reveals scope for future work.

Chapter 2

Statistical models for networks

In much of the machine learning literature to date, data can naturally be viewed as a sequence or a list. Of course, there are many other types and structures of data, such as those which are most naturally represented as a network. In this chapter we make the distinction between these types of data precise by symmetry arguments and invoke a theorem due to Aldous and Hoover to demonstrate appropriate forms of statistical model for these data types. In particular, we propose a nonparametric model of networks as a literal interpretation of this theorem. In chapter 3 we extend these results to more complex data structures.

This chapter is based on joint work with Peter Orbanz, Zoubin Ghahramani and Daniel Roy (Lloyd et al., 2012).

2.1 Introduction

Relational data records relations between entities. Such data arises in a variety of contexts, including graph-valued data (e.g. Airoldi et al., 2008; Hoff et al., 2002), micro-array data, tensor data (e.g. Xu et al., 2012) and collaborative filtering (e.g. Salakhutdinov and Mnih, 2008). Pairwise relations can be summarised in a 2-dimensional array (often also called a matrix); more generally, relations between d-tuples are recorded as d-dimensional arrays. We consider probabilistic models of infinite 2-arrays $\{X_{ij} : i, j \in \mathbb{N}\}$, where entries X_{ij} take values in a space $\mathbf{X}$. Each entry X_{ij} describes the relation between objects i and j. Finite samples—relational measurements for n objects—are $n \times n$-arrays. As the sample size increases, the data aggregates into a larger and larger array. Graph-valued data, for example, corresponds to the case $\mathbf{X} = \{0, 1\}$ where the values indicate whether or not two objects are connected by an edge. In collaborative filtering problems, the set

of objects is subdivided into two disjoint sets, e.g. users and items. This corresponds to a bipartite graph if the relations are additionally binary.

Latent variable models for relational data explain observations by means of an underlying structure or summary, such as a low-rank approximation to an observed matrix or an embedding into a Euclidean space. This structure is formalised as a latent (unobserved) variable. Examples of such models include probabilistic matrix factorisation (PMF) (Salakhutdinov and Mnih, 2008), the eigenmodel of Hoff (2007), the mixed-membership stochastic blockmodel (MMSB) (Airoldi et al., 2008) and the Gaussian process latent variable model (GPLVM) (e.g. Lawrence and Urtasun, 2009).

Hoff (2007) first noted that latent variable models of relational data can be justified by an exchangeability or symmetry argument. Building on this connection, we consider nonparametric models for graphs and arrays. Results of Aldous (1981), Hoover (1979) and Kallenberg (1992) show that random arrays which satisfy an exchangeability property can be represented in terms of a random function. The results can be regarded as a generalisation of de Finetti's theorem to array-valued data. Their implication for probabilistic modelling is that *we can specify a probability distribution for an exchangeable random array by specifying a probability distribution on (measurable) functions.* In this chapter, we model this function explicitly using a nonparametric distribution over functions.

2.2 Background: Exchangeable sequences and arrays

2.2.1 Reminder: de Finetti's theorem

Repeating from the main introduction to this thesis, suppose one receives a sequence of data $(X_1, X_2, \dots)$ and we wish to specify our beliefs about this data with a probability distribution. An often reasonable prior assumption is that we do not believe the order of the data contains any information, or rather that any probability distribution encoding our beliefs about this data should be invariant to reorderings of the data. That is, we may often believe that our data is an exchangeable sequence:

$$(X_1, X_2, \dots) \stackrel{\mathrm{d}}{=} (X_{p(1)}, X_{p(2)}, \dots) \qquad \forall\, p \in \mathbb{S}_\infty \tag{2.2.1}$$

where $\stackrel{\mathrm{d}}{=}$ denotes equality in distribution, and $\mathbb{S}_\infty$ is the set of all permutations of $\mathbb{N}$ which permute at most a finite number of elements.

de Finetti's theorem (e.g. Kallenberg, 2002) characterises all probability distributions of sequences with this property; $(X_i)_{i\in\mathbb{N}}$ is an exchangeable sequence if and only if there exists a random probability measure Θ such that $X_1, X_2, \ldots \stackrel{\text{iid}}{\sim} \Theta$. In other words, observations are conditionally i.i.d. given some random Θ. For the purposes of probabilistic modelling, this implies that our beliefs about exchangeable sequences of data can be expressed in the form of a distribution over probability measures.

2.2.2 de Finetti-type representations for exchangeable arrays

To specify probabilistic models for graph or array valued data, it would be helpful to have a suitable counterpart to de Finetti's theorem applicable when the infinite random sequences in (2.2.1) are substituted by infinite random arrays $X = (X_{ij})_{i,j\in\mathbb{N}}$. For such data, the invariance assumption (2.2.1) is typically too restrictive: In the graph case $X_{ij} \in \{0, 1\}$, for example, the probability of X would then depend only on the proportion of present edges in the graph, but not on the graph structure. Instead, we define exchangeability of random 2-arrays in terms of the *simultaneous* application of a permutation to rows and columns. More precisely:

Definition 2.2.1. An array $X = (X_{ij})_{i,j\in\mathbb{N}}$ is called a *jointly exchangeable array* if

$$(X_{ij}) \stackrel{\mathrm{d}}{=} (X_{p(i)p(j)}) \quad \forall\, p \in \mathbb{S}_\infty\ . \tag{2.2.2}$$

The assumption of exchangeability stated in (2.2.1) states that the order of the data does not matter. In contrast, array exchangeability is a statement of indifference to any ordering of the entities or objects that the data measures relations between. In fact, exchangeable sequences can be viewed in this way too, the difference is that data is measured at individual entities or objects rather than at pairs.

This symmetry is illustrated in figure 2.1. The two networks on the left hand side are equivalent except for the labelling of their nodes. This relabelling results in the different adjacency matrix representations of the same network shown on the right hand side of the figure. Array exchangeability states that any probabilistic model of these arrays should assign them the same probability.

This symmetry weakens the hypothesis (2.2.1) by demanding invariance only under a subset of all permutations of $\mathbb{N}^2$—those of the form $(i, j) \mapsto (p(i), p(j))$—so we can no

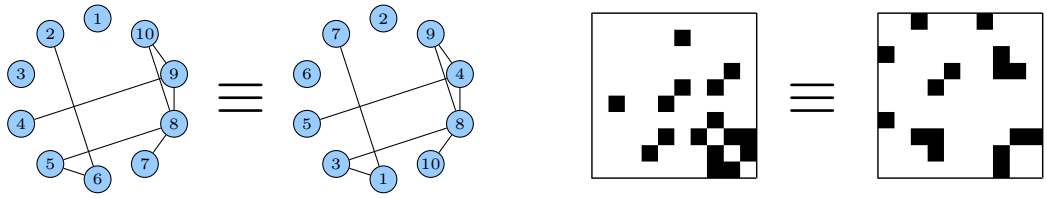


Figure 2.1: Left: Networks with equivalent structure but different node labels. Right: Corresponding adjacency matrix representations of these networks

longer expect de Finetti's theorem to hold. The relevant generalisation of the de Finetti theorem to this case is the following:

Theorem 2.2.1 (Aldous, Hoover)**.** *A random array (X_{ij}) is jointly exchangeable if and only if there is a random (measurable) function $F : [0,1]^3 \to \mathbf{X}$ such that the following holds: If $(U_i)_{i\in\mathbb{N}}$ and $(U_{\{ij\}})_{ij\in\mathbb{N}}$ are i.i.d. sequences of* Uniform$[0,1]$ *random variables, then*

$$(X_{ij}) \stackrel{d}{=} (F(U_i, U_j, U_{\{ij\}})) \qquad \text{for all } i, j \in \mathbb{N}\,. \tag{2.2.3}$$

Remark 2.2.2 (separately exchangeable arrays)**.** We say that an array (X_{ij}) is *separately exchangeable* when $(X_{ij}) \stackrel{\text{d}}{=} (X_{p(i)p'(j)})$ for every pair $p, p' \in \mathbb{S}_\infty$ of permutations. Such symmetry would be appropriate when modelling, e.g., a collaborative filtering task where the order of users and items are simultaneously irrelevant. A separately exchangeable array can be seen to correspond with an off-diagonal subarray of a jointly exchangeable array, and thus can be shown to have the representation $(X_{ij}) \stackrel{\text{d}}{=} (F(U_i, V_j, W_{ij}))$, where (U_i), (V_j) and $(W_{ij})_{i,j\in\mathbb{N}}$ are i.i.d. Uniform$[0,1]$ random variables.

2.2.3 The general case: d-arrays

Theorem 2.2.1 can in fact be stated in a more general setting than matrices, namely for random d-arrays, which are collections of random variables of the form $(X_{i_1\ldots i_d} : i_1, \ldots, i_d \in \mathbb{N})$. In particular, a sequence is a 1-array, a matrix a 2-array. A d-array can be interpreted as an encoding of a relation between d-tuples. In this general case, theorem 2.2.1 still holds, but the random function F in (2.2.3) is in general more complex: In addition to the collections (U_i) and $(U_{\{ij\}})$ of uniform variables, the representation requires all collections $(U_{\{i_j\}_{j\in I}})$ for any non-empty subset I of the set $\{1, \ldots, d\}$; e.g. $U_{\{i_1 i_3 i_4\}}$ for $d \geq 4$ and $I = \{1, 3, 4\}$. The representation (2.2.3) is then substituted by

$$F : [0,1]^{2^d-1} \to \mathbf{X} \qquad \text{and} \qquad (X_{i_1,\ldots,i_d}) \stackrel{d}{=} (F(U_{I_1}, \ldots, U_{I_{(2^d-1)}}))\,. \tag{2.2.4}$$

For $d = 1$, we recover de Finetti's theorem. In particular, if $\mathbf{X} = \mathbb{R}$, F is given by the inverse cumulative distribution function of the random measure guaranteed by de Finetti's theorem.

Since we do not explicitly consider the case $d > 2$ in our experiments in this chapter, we restrict our presentation throughout to the matrix-valued case for simplicity. We note, however, that the model and inference algorithms described in the following are applicable to general array-valued data. In chapter 3 we extend these representation results to arbitrary collections of arrays of arbitrary shape.

2.3 A nonparametric model of exchangeable arrays

To define a probabilistic model for array or graph valued data, we start with theorem 2.2.1: A distribution on exchangeable matrices can be specified by specifying a distribution on measurable functions $F : [0,1]^3 \to \mathbf{X}$. For simplicity we restrict attention to symmetric matrices $X_{ij} = X_{ji}$; in chapter 3 we present different arguments for which asymmetry is more elegantly handled but arrive at the same form of model. We decompose the function F into two functions $\Theta : [0,1]^2 \to \mathbf{W}$ and $H : [0,1] \times \mathbf{W} \to \mathbf{X}$ for a suitable space $\mathbf{W}$, such that

$$F(U_i, U_j, U_{\{ij\}}) = H(U_{\{ij\}}, \Theta(U_i, U_j)) \ . \tag{2.3.1}$$

Such a decomposition always exists—trivially, choose $\mathbf{W} = [0,1]^2$. Equivalently, there is a conditional probability $P[\,.\,|\Theta]$ on $\mathbf{X}$ such that

$$F(U_i, U_j, U_{\{ij\}}) \sim P[\,.\,|\Theta(U_i, U_j)] \ . \tag{2.3.2}$$

Thus, (2.3.1) can be interpreted as a decomposition into a variable Θ—the model parameter—which captures the structure of the underlying graph or array, and random noise represented by H or P, respectively.

To define a model, we assume Θ to be continuous, and to be distributed according to a Gaussian process. More precisely, we set $\mathbf{W} = \mathbb{R}$ and consider a zero-mean Gaussian process on $\mathbf{C}_{\mathbf{W}} := \mathbf{C}([0,1]^2, \mathbf{W})$, the space of continuous functions $[0,1]^2 \to \mathbf{W}$, with

kernel function $k : [0,1]^2 \times [0,1]^2 \to \mathbf{W}$. We assume the following generative model:

$$\begin{aligned} \Theta &\rightsquigarrow \mathrm{GP}(0,k) \\ U_1, U_2, \ldots &\overset{\text{iid}}{\rightsquigarrow} \mathrm{Uniform}[0,1] \\ X_{ij} &\rightsquigarrow P[\,.\,|\Theta(U_i,U_j)]\,. \end{aligned} \tag{2.3.3}$$

Additionally, the kernel may depend on parameters, which we denote by ψ. Since we have assumed X to be symmetric we can ignore the potentially unusual depedencies introduced by $U_{\{ij\}}$ e.g. we can restrict the model to the upper triangle of X[1].

The decomposition of F introduces a natural set of latent variables $W_{ij} := \Theta(U_i, U_j)$, conditioned on which the data is explained as $X_{ij} \sim P[\,.\,|W_{ij}]$. The parameter space of the model is the infinite-dimensional space $\mathbf{C}_{\mathbf{W}}$.

The choice of P depends on the type of data considered, two common cases are graphs and real-valued matrices. In either case, the model first generates a latent matrix $W = (W_{ij})$. Depending on the type of data, observations could then be generated as follows:

Observed data	Sample space	$P[X_{ij} \in \,.\,\|W_{ij}]$
Graph	$\mathbf{X} = \{0,1\}$	$\mathrm{Bernoulli}(\sigma(W_{ij}))$
Real matrix	$\mathbf{X} = \mathbb{R}$	$\mathrm{Normal}(W_{ij}, \sigma^2_{\mathbf{X}})$

where σ is the logistic function, and $\sigma^2_{\mathbf{X}}$ is a noise variance parameter.

The modelling assumptions we impose in addition to exchangeability are thus (i) that the function Θ is continuous—which implies measurability but is a stronger requirement—and (ii) that it is distributed according to a Gaussian process measure on $\mathbf{C}_{\mathbf{W}}$. Additionally, (iii) the specific choice of P may or may not introduce an additional assumption. In the case of graphs, the only assumptions are (i) and (ii), the Bernoulli likelihood can be shown to be general (e.g. Aldous, 2010). In the case of real-valued matrices, however, the model additionally assumes that the function H in (2.3.1) is of the form

$$H(U_{\{ij\}}, \Theta(U_i,U_j)) \overset{\text{d}}{=} \Theta(U_i,U_j) + \varepsilon_{\{ij\}} \qquad \text{where} \qquad \varepsilon_{\{ij\}} \sim \mathrm{Normal}(0,\sigma^2)\,. \tag{2.3.4}$$

The Gaussian process prior favours smooth functions, which may result in interpretable latent space embeddings. Inference in Gaussian processes is a well-understood problem, and the choice of a Gaussian prior allows us to leverage the full range of inference methods available for these models.

[1] The results of chapter 3 inspire the same form of model but also naturally avoid these dependencies.

The model described in equation (2.3.3) is illustrated for the case of graph data in figure 2.2. This demonstrates that Θ can be interpreted as a blurred adjacency matrix.

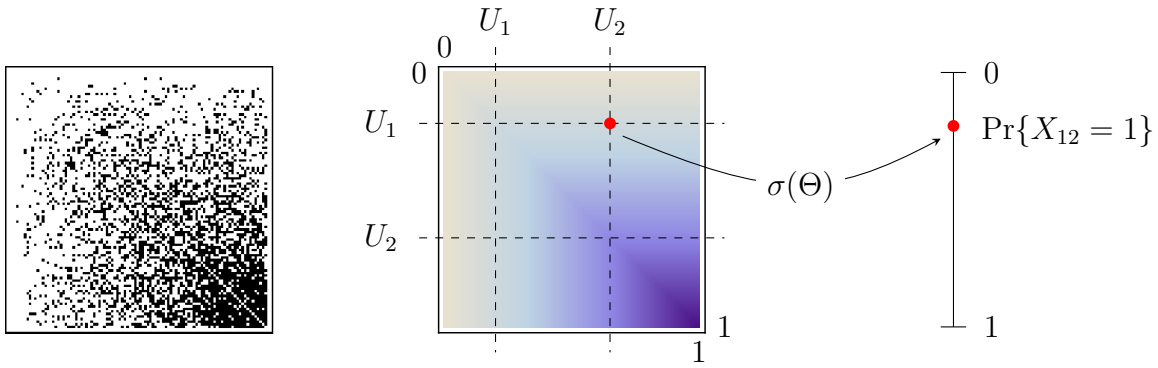


Figure 2.2: Illustration of a model for network data inspired by the Aldous–Hover representation theorem. The left shows a random sample of a binary network (represented by an adjacency matrix) generated by a model of the form given by equation (2.3.3).

Remark 2.3.1 (Dense vs. sparse data)**.** The methods described here address random matrices that are *dense*, i.e. as the size of a finite, observed $n \times n$ matrix increases, the number of non-zero entries (or the number of edges in a graph) grows as $O(n^2)$. Although the Aldous-Hoover theorem is sometimes invoked for data such as social networks, it should be kept in mind that network data is typically *sparse*, with $O(n)$ non-vanishing entries. Density is an immediate consequence of theorem 2.2.1: For graph data, for example, the asymptotic proportion of present edges is $p := \int \Theta(x, y)dxdy$, and the graph is hence either empty (for $p = 0$) or dense (since $O(pn^2) = O(n^2)$). Analogous representation theorems for sparse random graphs are to date an open problem in probability. There is however an emerging literature on representation theorems for sparse graphs including graphings (e.g. Lovász, 2012), thinning of graphon representations (Wolfe and Olhede, 2013) and constructing graphs from exchangeable measures on $\mathbb{R}^2_+$ (Caron and Fox, 2014).

2.4 Related work

Our model has some noteworthy relations to the Gaussian process latent variable model (GPLVM); a dimensionality-reduction technique (e.g. Lawrence, 2005). GPLVMs can be applied to relational data (Lawrence and Urtasun, 2009), but doing so makes the assumption that either the rows or the columns of the random matrix are independent.

In terms of our model, this corresponds to choosing kernels of the form $k_U \otimes \delta_V$, where $\otimes$ represents the Kronecker matrix product and δ represents an 'identity' kernel (i.e. the corresponding kernel matrix is the identity matrix). From this perspective, the application of our model to separately exchangeable real-valued matrices can be interpreted as a form of co-dimensionality reduction.

A related parametric model is the eigenmodel of Hoff (2007). This model, also justified by exchangeability arguments, approximates a matrix with a bilinear form, followed by some link function and distribution depending on the type of data. We show in proposition 2.4.1 that this can be represented in the form of our model by using a kernel function of the form $L_U \otimes L_V$ where L represents the dot product kernel (i.e. the kernel that gives rise to linear functions).

Available models for graph data include the parametric mixed membership stochastic blockmodel (MMSB; Airoldi et al. (2008)) and nonparametric infinite relational model (IRM; Kemp et al. (2006)), latent feature relational model (LFRM; Miller et al. (2009)), infinite latent attribute model (ILA; Palla et al. (2012)) and many others. Similar models also exist for real valued matrices, including the infinite hidden relational model (IHRM; Xu et al. (2006)) which is a counterpart to the IRM and binary matrix factorisation (BMF; Meeds et al. (2007)) which is a counterpart to the LFRM.

Roy and Teh (2009) present a nonparametric Bayesian model of relational data that approximates Θ by a piecewise constant function whose hierarchical structure is a Mondrian process. In contrast, we approximate Θ by a smooth function.

A recent development is the Sparse Matrix Variate Gaussian Process Blockmodel (SMGB; Yan et al. (2011)), and the extension to higher order arrays (InfTucker; Xu et al. (2012)). Although not motivated in terms of exchangeability, these Gaussian process models do not impose independence assumptions on either rows or columns, in contrast to the GPLVM. These models can be represented as placing a prior $\Theta \sim \mathrm{GP}(0, k_1 \otimes k_2)$ on the function Θ in (2.3.1). This prior is equivalent to (2.3.3) for some common choices of kernel k (e.g. squared exponential) but excludes others such as those that give rise to symmetric functions or additive kernels. Existing efficient algorithms for Gaussian processes of this form (e.g. Saatçi, 2011) perform calculations using a complete grid of data, which results in computational complexities that depend on the size of the array. Our work suggests that it may not be necessary to impose Kronecker product structure on the kernel, which allows for inference with improved scaling when data is sparsely observed. However, their most recent work (Zhe et al., 2013) has employed many other approximate inference strategies resulting in an algorithm that can be applied to data

much larger than that considered here.

2.4.1 A common perspective on the literature

The representations (2.3.1) and (2.3.2) allow for a common perspective on a range of models in the literature depending on how they construct (W_{ij}) or place a prior over Θ. The following proposition allows us to specify many models in both forms.

Proposition 2.4.1. *A matrix factorisation model defined as*

$$W_{ij} = U_i \Lambda V_j' \qquad \Lambda_{ij} \overset{iid}{\sim} \mathcal{N}(0,1) \tag{2.4.1}$$

is equivalent to

$$W_{ij} = \Theta\left(U_i, V_j\right) \qquad \Theta \sim \mathrm{GP}\left(0, L_U \otimes L_V\right) \tag{2.4.2}$$

where $L_U(U_{i_1}, U_{i_2}) = U_{i_1} U_{i_2}'$ *and similarly for* L_V*.*

Proof. For a finite collection of (W_{ij}), the Gaussian process model can be written as

$$(W_{ij}) \sim \mathcal{N}\left(0, K_U \otimes K_V\right) \tag{2.4.3}$$

where K is the kernel matrix corresponding to the kernel function L and $\otimes$ represents a Kronecker product. In the matrix factorisation, W_{ij} is a linear combination of Gaussian distributed random variables and hence is Gaussian distributed itself. Thus, we need only show that it has zero mean and the required covariance structure. This is trivial,

$$\begin{aligned}
\mathbb{E}(W_{ij} \,|\, U_i, V_j) = U_i \mathbb{E}(\Lambda) V_j' &= 0 && (2.4.4)\\
\mathrm{Cov}(W_{i_1 j_1}, W_{i_2 j_2} \,|\, U_{i_1}, U_{i_2}, V_{j_1}, V_{j_2}) &= \mathbb{E}(W_{i_1 j_1} W_{i_2 j_2} \,|\, U_{i_1}, U_{i_2}, V_{j_1}, V_{j_2}) \\
&= \sum_{k_1, l_1, k_2, l_2} u_{i_1 k_1} v_{j_1 l_1} u_{i_2 k_2} v_{j_2 l_2} \mathbb{E}(\lambda_{k_1, l_1} \lambda_{k_2 l_2}) \\
&= \Big(\sum_k u_{i_1 k} u_{i_2 k} \Big) \Big(\sum_l v_{j_1 l} v_{j_2 l} \Big) \\
&= U_{i_1} U_{i_2}' \times V_{j_1} V_{j_2}'. && (2.4.5)
\end{aligned}$$

□

This correspondence between matrix factorisation and priors on functions can be kernelized by mapping U_i, V_j through functions ϕ_U, ϕ_V respectively. Thus, we see that a Gaussian process model using a tensor product of general kernels is equivalent to kernelized matrix factorisation.

This correspondence allows us to succinctly summarise many of the models in the literature review in table 2.1 below (the model introduced here is referred to as the random function model).

	W_{ij}	k	$U_i, V_j \in$.
Random function model	$\phi(U_i, V_j)\Lambda$	$k_{U\times V}$	$\mathbb{R}^d$, $[0,1]$
SMGB, InfTucker	$\phi(U_i)'\Lambda\phi(V_j)$	$k_U \otimes k_V$	$\mathbb{R}^d$
GPLVM, Kernel PCA	$\phi(U_i)\Lambda$	$k_U \otimes \delta_V$	$\mathbb{R}^d$
Eigenmodel	$U_i\Lambda V_j'$	$L_U \otimes L_V$	$\mathbb{R}^d$
Linear relational GP	$U_i\Lambda V_j'$	$L_U \otimes L_V$	$\mathbb{R}^d$
PMF	$U_i V_j'$	0	$\mathbb{R}^d$
PCA, Factor analysis, etc.	$U_i\Lambda$	$L_U \otimes \delta_V$	$\mathbb{R}^d$
Latent distance	$-\lvert U_i - U_j\rvert$	0	$\mathbb{R}^d$
Mondrian process based	Decision tree	*	$[0,1]^d$
Latent class	$\Lambda_{U_iU_j}$	$\delta_{U\times U}$	$\{1,\ldots,d\}$
IRM	$\Lambda_{U_iU_j}$	$\delta_{U\times U}$	$\{1,\ldots,\infty\}$
IHRM	$\Lambda_{U_iV_j}$	$\delta_{U\times V}$	$\{1,\ldots,\infty\}$
BMF	$U_i\Lambda V_j'$	$L_U \otimes L_V$	$\{0,1\}^\infty$
LFRM	$U_i\Lambda U_j'$	$L_U \otimes L_U$	$\{0,1\}^\infty$
ILA	$\sum_d \mathbb{I}_{U_{id}}\mathbb{I}_{U_{jd}}\Lambda^{(d)}_{U_{id}U_{jd}}$	*	$\{0,\ldots,\infty\}^\infty$

Table 2.1: Summary of models classified by equation for W_{ij}, equivalent kernel k in GP representation and domain of latent variables $(U_i), (V_j)$. Λ is a matrix, ϕ is some mapping into a feature space and $\mathbb{I}_A$ is the indicator function of the event $A > 0$. k represents any kernel function, δ the identity kernel function and L the simple dot product kernel function.

Table 2.1 necessarily glosses over many details (e.g. choice of priors, inference methods), but it is hoped that it reveals the structural similarities between many models in the literature. Despite not often being viewed as a network model we have included principal component analysis (PCA) as a linear counterpart to the non-linear GPLVM.

This table reveals that the majority of models included here model Θ as linear or bilinear (potentially kernelized). The Mondrian process based model (Roy and Teh, 2009) is a notable exception since it models Θ as a piecewise constant random function which is fundamentally different from our approach of modelling Θ to be continuous. For simplicity we have refrained from writing down the kernel corresponding to the Mondrian process based model; it could be described as an indicator function of the equivalence class induced by the Mondrian process. We also see that the ILA model is significantly more flexible in its specification of Θ than other models using discrete latent variables.

Again, we refrain from writing down what would be a fairly complicated kernel function corresponding to this model.

We note that the nonparametric Gaussian process prior on Θ in the random function models means that, if supported by the data, the posterior distribution of Θ could approximate any of the functions specified by other models. This is of course true of the other nonparametric models; particular priors will influence the rate at which different forms of Θ can be inferred.

2.5 Posterior computation

In this section we describe Markov Chain Monte Carlo (MCMC) algorithms for generating approximate samples from the posterior distribution of the model parameters given a partially observed array. Most importantly, we describe a random subset-of-regressors approximation that scales to graphs with hundreds of nodes and tens of thousands of edge observations. Given the relatively straightforward nature of the proposed algorithms and approximations, we refer the reader to other papers whenever appropriate.

2.5.1 Latent space and kernel

The random function representation in theorem 2.2.1 is not restricted to the use of uniform distributions for the variables (U_i). The uniform distributions arise in the proof of the theorem as generic distributions with which one can encode the information of other random variables via a measurable function. Therefore, the proof remains unchanged if one replaces the uniform distributions with any isomorphic probability distribution i.e. any non-atomic probability measure on a Borel space. For the presentation in section 2.2, the domain $[0,1]^2$ of the pairs of uniforms provides an intuitive analogy with adjacency matrices. For the purposes of inference, normal distributions are more convenient, and we henceforth use $U_1, U_2, \ldots \overset{\text{iid}}{\sim} \mathcal{N}(0, I_r)$ for some integer r.

Since we focus on undirected graphical data, we require the symmetry condition $W_{ij} = W_{ji}$. This can be achieved by constructing the kernel function in the following way (e.g. Duvenaud, 2014)

$$k(\xi_1, \xi_2) = \frac{1}{2}\Big(\bar{k}(\xi_1, \xi_2) + \bar{k}(\xi_1, \bar{\xi}_2)\Big) + \sigma^2 I \qquad \text{(Symmetry + noise)} \tag{2.5.1}$$

$$\bar{k}(\xi_1, \xi_2) = s^2 \exp(-|\xi_1 - \xi_2|^2/(2\ell^2)) \qquad \text{(RBF kernel)} \tag{2.5.2}$$

where $\xi_k = (U_{i_k}, U_{j_k})$, $\bar{\xi}_k = (U_{j_k}, U_{i_k})$ and s, ℓ, σ represent a scale factor, length scale and

noise respectively (see e.g. chapter 4). We collectively denote the kernel parameters by ψ. For undirected data one can model only the upper triangular part of the adjacency matrix.

2.5.2 Sampling without approximating the model

In the simpler case of a real-valued array X, we construct an MCMC algorithm over the variables $((U_i), \psi, \sigma_X)$ by repeatedly slice sampling (Neal, 2003) from the conditional distributions

$$\psi_i \,|\, \psi_{-i}, \sigma_X, (U_i), X \qquad \sigma_X \,|\, \psi, (U_i), X \qquad \text{and} \qquad U_j \,|\, U_{-j}, \psi, \sigma_X, X \tag{2.5.3}$$

where σ_X is the noise variance parameter used when modelling real valued data introduced in section 2.3. Let $\mathrm{N} = |(U_i)|$ denote the number of rows in the observed array, let ξ be the set of all pairs (U_i, U_j) for all observed relations X_{ij}, let $\mathrm{O} = |\xi|$ denote the number of observed relations, and let K represent the $\mathrm{O} \times \mathrm{O}$ kernel matrix between all points in ξ. Changes to ψ affect every entry in the kernel matrix K and so, naively, the computation of the Gaussian likelihood of X takes $\mathcal{O}(\mathrm{O}^3)$ time. Likewise, changes to (U_i) also affect entries in K. Naively re-evaluating the Gaussian likelihood term again takes $\mathcal{O}(\mathrm{O}^3)$ time, and so, across all N rows, we expend $\mathcal{O}(\mathrm{O}^3\mathrm{N})$ time. Whether or not this can be improved by exploiting structure in the modifications, the cubic dependence on O seems unavoidable, and thus this naive algorithm is unusable for all but very small data sets.

2.5.3 A random subset-of-regressor approximation

In order to scale the method to much larger graphs, we applied a variation of a sparse approximation method known as Subsets-of-Regressors, or simply SoR (Silverman, 1985; Smola and Bartlett, 2001; Wahba et al., 1999) (See Quiñonero Candela and Rasmussen (2005) for an excellent survey of this and other sparse approximations). The SoR approximation replaces the infinite dimensional GP with a finite dimensional approximation. Our approach is to treat both the inputs and outputs of the GP as latent variables.

In particular, we introduce k Gaussian distributed pseudoinputs $\eta = (\eta_1, \ldots, \eta_k)$, and, for each $j = 1, \ldots, k$, define target values $T_j = \Theta(\eta_j)$. In other words, writing $K_{\eta\eta}$

for the kernel matrix formed from the pseudoinputs η, we have

$$(\eta_i) \overset{\text{iid}}{\sim} \mathcal{N}(0,1) \qquad \text{and} \qquad T \,|\, \eta \sim \mathcal{N}(0, K_{\eta\eta}). \tag{2.5.4}$$

The idea of the SoR approximation is to replace W_{ij} with the posterior mean conditioned on (η, T),

$$W = K_{\xi\eta} K_{\eta\eta}^{-1} T, \tag{2.5.5}$$

where $K_{\xi\eta}$ is the kernel matrix between the latent embeddings ξ and the pseudoinputs η. By considering random pseudoinputs, we construct an MCMC analogue of the techniques proposed in Titsias and Lawrence (2008).

Given this approximate model, the conditional distribution $T \,|\, (U_i), \eta, \psi, (\sigma_X), X$ is amenable to elliptical slice sampling (Murray et al., 2010), for both real valued and binary X. All other random parameters, including the (U_i) and (η_i), can again be sampled from their full conditional distributions using slice sampling. The sampling algorithms require that one computes expressions involving (2.5.5). As a result they cost at most $\mathcal{O}(k^3\mathrm{O})$.

2.6 Experiments

We evaluate the model on three different network data sets (see table 2.2). Two of these data sets—the high school and NIPS co-authorship data—have been extensively analysed in the literature. The third data set, a protein interactome, was previously noted by Hoff (2007) to be of interest since it exhibits both block structure and transitivity.

Data set	Recorded data	Vertices	Reference
High school	high school social network	90	e.g. Hoff (2007)
NIPS	subset of coauthorship network	234	e.g. Miller et al. (2009)
Protein	protein interactome	230	e.g. Hoff (2007)

Table 2.2: Summary of data sets in link prediction experiment.

We compare performance of our model on these data sets to three other models (see table 2.3), probabilistic matrix factorization (PMF) (Salakhutdinov and Mnih, 2008), Hoff's eigenmodel (Hoff, 2007), and the GPLVM (Lawrence, 2005). The models are

chosen for comparability, since they all embed nodes into a Euclidean latent space. Experiments for all three models were performed using reference implementations by the respective authors (see table 2.3 for additional algorithm details).[2]

Model	Method	Iterations [burn-in]	Parameters
PMF	stochastic gradient	1000	author defaults
Eigenmodel	MCMC	10000 [250]	author defaults
GPLVM	stochastic gradient	20 sweeps	author defaults
Random function model	MCMC	1000 [200]	(see below)

Table 2.3: Summary of algorithms and settings in link prediction experiment.

We use standard normal priors on the latent variables U and pseudo points η, and log normal priors for kernel parameters.

Slice sampling parameters are chosen to favour slice sampling acceptance after a reasonable number of iterations, as evaluated over a range of data sets. Specific values of prior and sampling parameters are summarised in the table on the right. To balance the computational demands of the different sampling steps, we sampled T 50 times per iteration whilst all other variables were sampled once per iteration.

	exp mean	std	width
length scale	1	0.5	0.5
scale factor	2	0.5	0.5
target noise	0.1	0.5	0.1
U	-	-	4
η	-	-	2

In all experiments, we partitioned the edge data into 5 equally sized partitions and performed 5-fold cross validation, predicting the links in one held out partition given the other 4. Where the models did not restrict their outputs to values between 0 and 1, we truncated any predictions lying outside this range. Table 2.4 reports average AUC (area under receiver operating characteristic) for the various models, with numbers for the top performing model set in bold. Significance of results is evaluated by means of a t-test with a p-value of 0.05; results for models not distinguishable from the top performing model in terms of this t-test are also set in bold.

The random function model outperforms the other models in *all* tests. We also note that in all experiments, a single latent dimension suffices to achieve better performance, even when the other models use additional latent dimensions.

[2]Implementations are available for PMF at `http://www.mit.edu/~rsalakhu/software.html`; for the eigenmodel at `http://cran.r-project.org/src/contrib/Descriptions/eigenmodel.html`; and for the GPLVM at `http://www.cs.man.ac.uk/~neill/collab/`.

AUC results

Data set	High school			NIPS			Protein		
d	1	2	3	1	2	3	1	2	3
PMF	0.747	0.792	0.792	0.729	0.789	0.820	0.787	0.810	0.841
Eigenmodel	0.742	0.806	0.806	0.789	0.818	0.845	0.805	0.866	0.882
GPLVM	0.744	0.775	0.782	0.888	0.876	0.883	0.877	0.883	0.873
RFM	**0.815**	**0.827**	**0.820**	**0.907**	**0.914**	**0.919**	**0.903**	**0.910**	**0.912**

Table 2.4: Average AUC for each method and dataset. d is the dimensionality of the latent variables used in each method. Results that are not distinguishable from the best performing method by a t-test at 5% significance are set in bold.

In general, a more general model of data will not necessarily improve empirical performance compared to simpler models. However, our results provide empirical evidence that the flexibility of a general nonparametric model can make an important difference for the problems considered here.

The posterior distribution of Θ favours functions defining random array distributions that explain the data well. In this sense, our model fits a probability distribution. The standard inference methods for GPLVM and PMF applied to relational data, in contrast, are designed to fit mean squared error, and should therefore be expected to show stronger performance under a mean squared error metric. As table 2.5 shows, this is indeed the case. The reported values are RMSE results, with significance evaluated as above.

RMSE results

Data set	High school			NIPS			Protein		
d	1	2	3	1	2	3	1	2	3
PMF	0.245	0.242	0.240	0.141	0.135	0.130	0.151	0.142	0.139
Eigenmodel	0.244	0.238	**0.236**	0.141	0.132	0.124	0.149	0.142	**0.138**
GPLVM	0.244	0.241	0.239	**0.112**	**0.109**	**0.106**	**0.139**	**0.137**	**0.138**
RFM	**0.239**	**0.234**	**0.235**	**0.114**	**0.111**	**0.110**	**0.138**	**0.136**	**0.136**

Table 2.5: Average RMSE for each method and dataset. d is the dimensionality of the latent variables used in each method. Results that are not distinguishable from the best performing method by a t-test at 5% significance are set in bold.

An arguably more suitable metric is comparison in terms of conditional edge probability i.e. $P(X_{\{ij\}} \,|\, W_{\{ij\}})$ for all i, j in the held out data. These cannot, however, be

Negative log conditional edge probability									
Data set	High school			NIPS			Protein		
d	1	2	3	1	2	3	1	2	3
Eigenmodel	220	210	**200**	88	81	75	96	92	86
RFM	**205**	**199**	**201**	**65**	**57**	**56**	**78**	**75**	**75**

Table 2.6: Average negative log conditional edge probability for the Eigenmodel and RFM for each dataset. The precise calculation implemented is $-\log(P(X_{\{ij\}} \,|\, W_{\{ij\}})) \times 1000\,/\,(\text{Number of held out edges})$. d is the dimensionality of the latent variables used in each method. Results that are not distinguishable from the best performing method by a t-test at 5% significance are set in bold.

computed in a meaningful manner for standard versions of models such as PMF and GPLVM, which assign a Gaussian likelihood to data. Table 2.6 hence reports only comparisons to the eigenmodel.

2.6.1 Inferred representing functions

We now discuss what can be learned about a network by inspecting the posterior for the representing function Θ. Figure 2.3 shows a visualisation of the protein interactome (left), its adjacency matrix (middle) and approximate maximum a posteriori (MAP) estimate of Θ (right). The adjacency matrix has been sorted by the values of the approximate MAP (U_i).

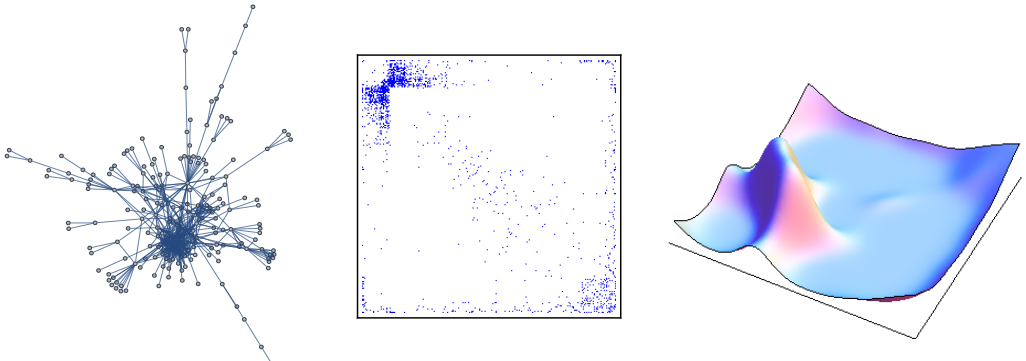


Figure 2.3: Protein interactome data. *Left:* Interactome network. *Middle:* Sorted adjacency matrix. The network exhibits stochastic equivalence (visible as block structure in the matrix) and homophily (concentration of points around the diagonal). *Right:* Maximum a posteriori estimate of the function Θ, corresponding to the function in figure 2.2 (middle).

The sorted adjacency matrix and MAP estimate of Θ reveal many qualitatively different structures present in this network. The bottom right of the adjacency matrix reveals a densely connected block of nodes; this is highlighted and visualised in figure 2.4. In contrast, figure 2.5 highlights many nodes which display patterns of transitivity although the overall connectivity is sparse. A pair of hub nodes are also revealed as demonstrated in figure 2.6.

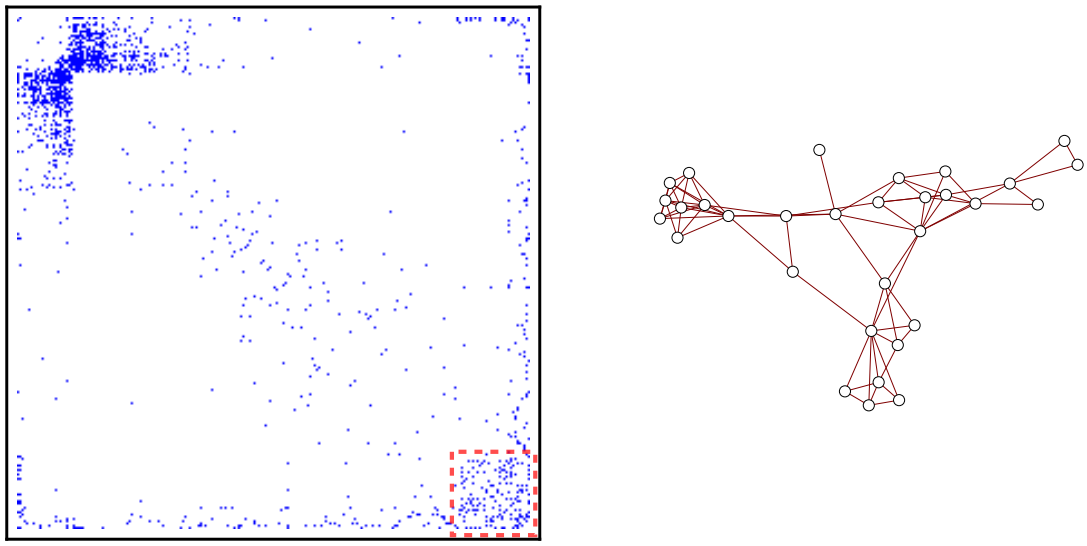


Figure 2.4: A densely connected group of nodes in the protein interactome.

Finally, the prominent feature at the top left of the adjacency matrix is visualised in figure 2.7. The sorted adjacency matrix has revealed three blocks of nodes which we will refer to as the left, right and top blocks according to the right hand side of figure 2.7. Ignoring the top block, the left and right blocks are nearly bipartite. The top block is almost a clique and is much more strongly connected to the right block than the left. The degree distribution of the left and right blocks is also of interest; the variation is larger than expected by a block model (i.e. assuming that the probability of a link between different blocks is constant). A more detailed analysis is currently ongoing, but the structure found by this analysis can be shown to correspond with reality; for example, the proteins on the left are almost all ribosomal subunits.

Similar but less complicated structures are observed in the NIPS and Highschool datasets. Figure 2.8 shows unsorted (left) and sorted (right) adjacency matrices for the NIPS coauthorship network. The map embedding of the U_i has identified clear block structures of densely connected groups of authors. However, there are many authors (in the middle of the adjacency matrix) for which there is no block structure, only

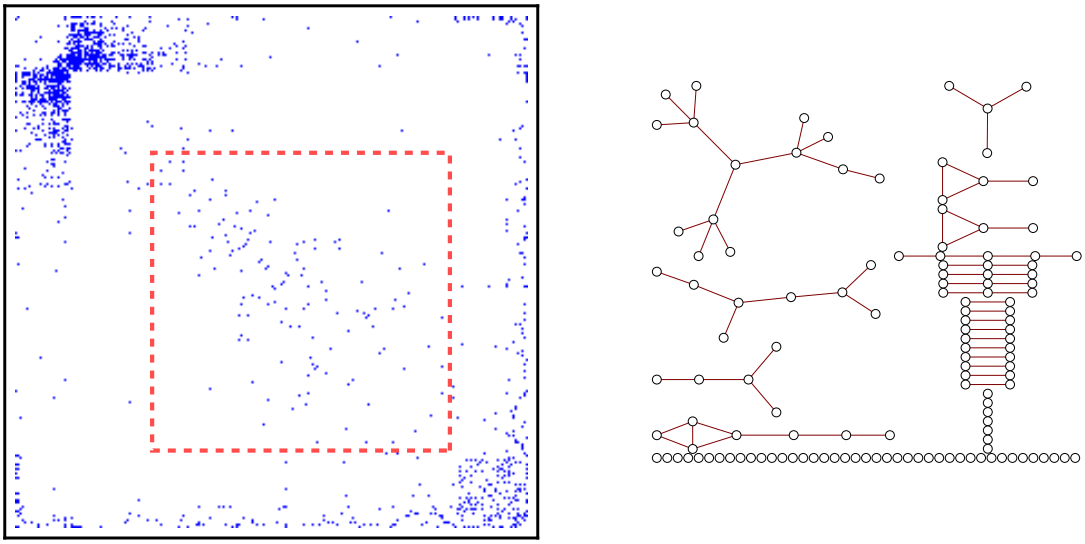


Figure 2.5: Sparsely connected nodes in the protein interactome with some transitivity (triangles).

transitivity (coauthors of coauthors are likely to be coauthors). Similarly, the sorted high school social network adjacency matrix in figure 2.9 mostly shows patterns of transitivity with most links near the diagonal.

2.7 Discussion and conclusions

There has been a tremendous amount of research into modelling matrices, arrays, graphs and relational data, but nonparametric Bayesian modelling of such data has only recently been explored. In most modelling circumstances, the assumption of exchangeability amongst data objects is natural and fundamental to the model. In this case, the representation results (Aldous, 1981; Hoover, 1979; Kallenberg, 1992) precisely map out the scope of possible probabilistic models for exchangeable arrays: Any such model can be interpreted as a prior on random measurable functions on a suitable space.

Nonparametric Bayesian statistics provides a number of possible priors on random functions, but the Gaussian process and its modifications are the only well-studied model for almost surely continuous functions. For this choice of prior, our work provides a general and simple modelling approach that can be motivated directly by the relevant representation results. The model results in both interpretable representations for networks, such as a visualisation of a protein interactome, and has competitive predictive performance on benchmark data.

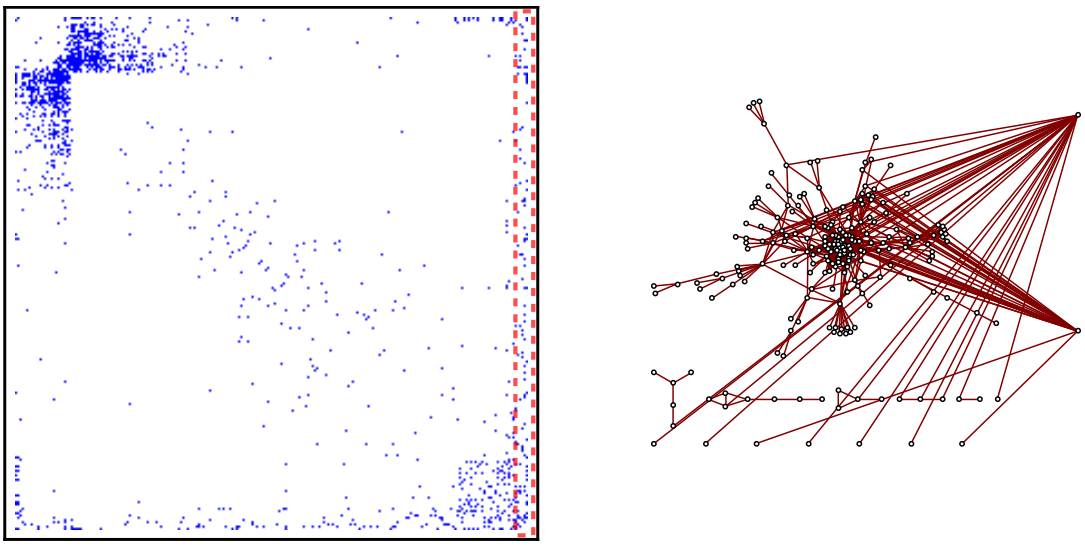


Figure 2.6: Two hub nodes in the protein interactome.

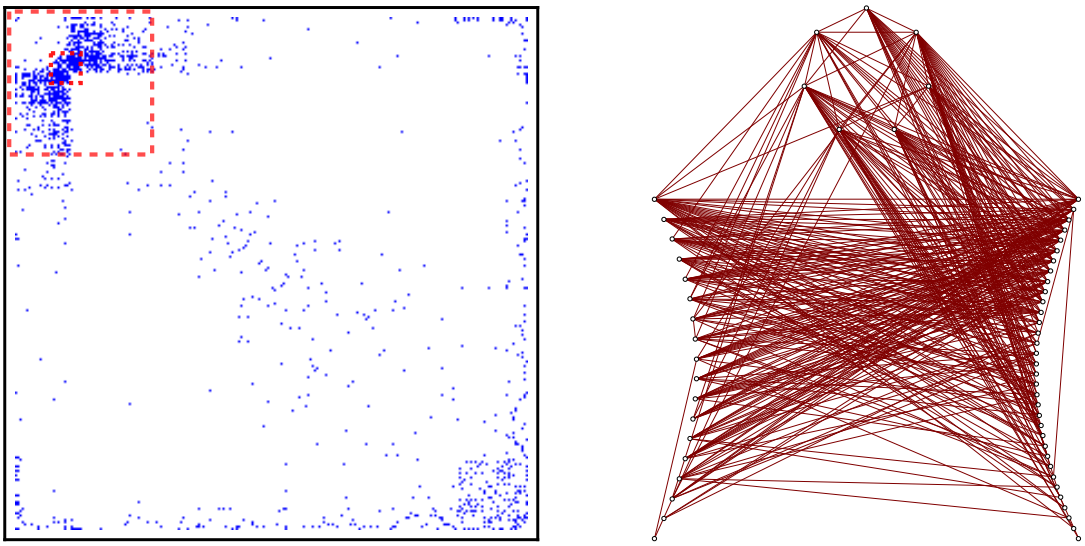


Figure 2.7: Three groups of nodes in the protein interactome with an interesting connectivity pattern.

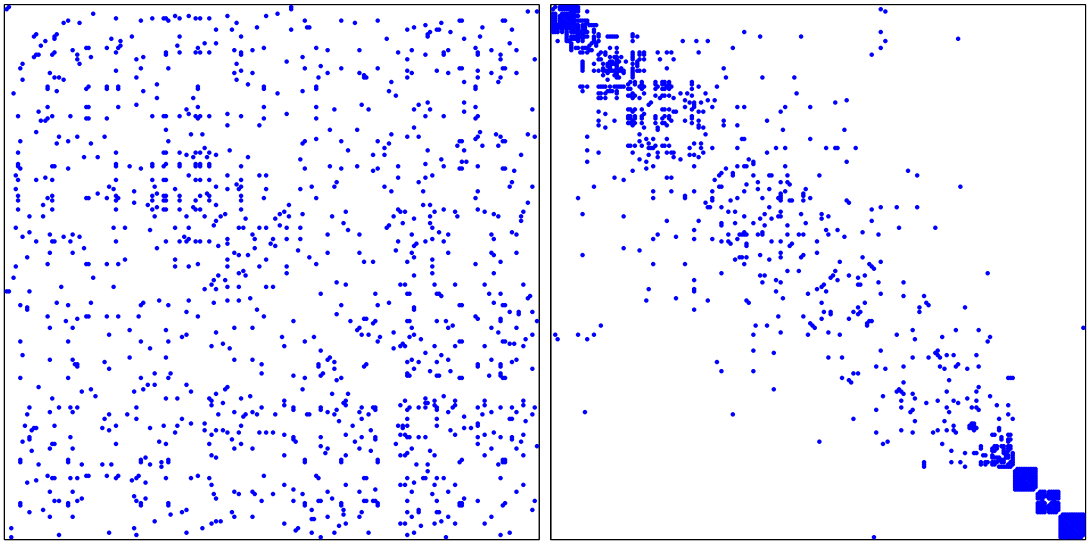


Figure 2.8: NIPS coauthorship data. *Left:* Unsorted adjacency matrix. *Middle:* Sorted adjacency matrix.

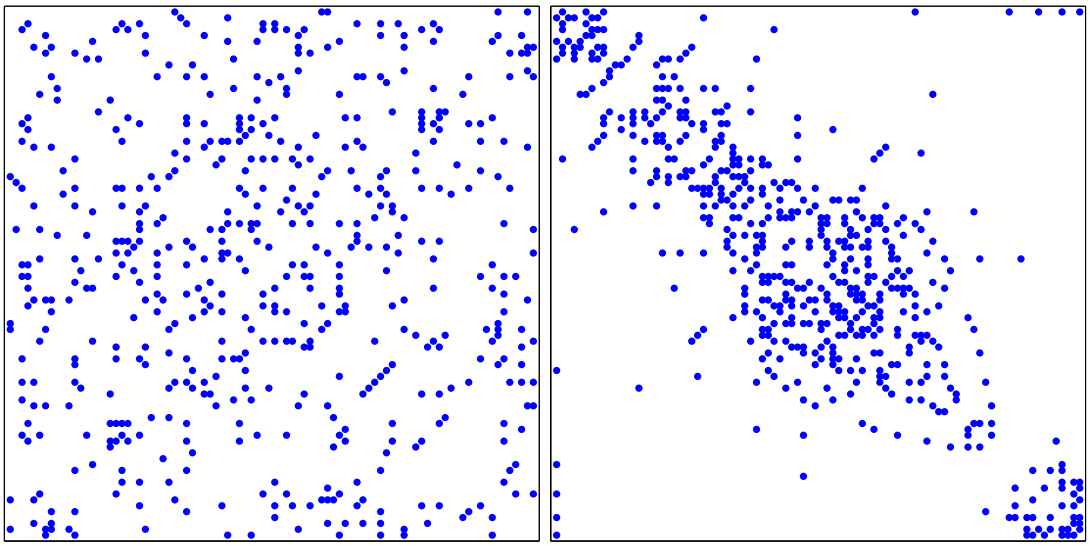


Figure 2.9: Highschool social network. *Left:* Unsorted adjacency matrix. *Middle:* Sorted adjacency matrix.

Chapter 3

Representing probability distributions of exchangeable databases

In chapter 2 we demonstrated how to characterise probability distributions over objects such as networks or user–item rating matrices (see remark 2.2.2). The key properties that allowed us to characterise these distributions were symmetries in the data, or invariances to the way they are stored. For networks, we assumed that the ordering of the nodes contained no information. For user–item data we assumed that the order of both users and items were irrelevant to the meaning of the data.

In this chapter we extend these ideas of exchangeability to any data that can be stored in a relational database i.e. arbitrary relational data. We demonstrate that extensions of the Aldous–Hoover theorem to n-arrays due to Kallenberg (1999) can be extended to the case of exchangeable databases. These new theorems reveal a natural parameter space for data stored in relational databases elucidating how to construct appropriate statistical models for such data.

This chapter is largely based on collaborations with Zoubin Ghahramani, Peter Orbanz and Daniel Roy. In particular, preliminary versions of the technical results presented in this chapter appeared at the Bayesian Nonparametrics Workshop 2012 and a NIPS workshop (Lloyd et al., 2013) but without proof.

3.1 Introduction

Relational databases are an extremely common data structure so it is natural to want to perform statistical tasks with such data e.g. predicting unobserved data or identifying latent structure. In particular, network data is rarely encountered in isolation e.g. in a social network one will often have access to information about each user. To perform a statistical analysis of such data we will typically need to specify a probabilistic model of the data, but it is not immediately clear what an appropriate parameter space for such a model is. The choice of parameter space is important because it indicates the targets of statistical inference and determines where we can share statistical strength between different aspects of the data.

We demonstrate that the weak assumption of an appropriate form of exchangeability can provide a natural parameter space. This form of exchangeability is appropriate when the order of the objects underlying a relational database (e.g. users and movies in a database of ratings data) is arbitrary or unimportant. For example, the left hand side of figure 3.1 shows the same network but with differently labeled nodes. If the labelling is unimportant, then any probabilistic model of such data should assign them the same probability.

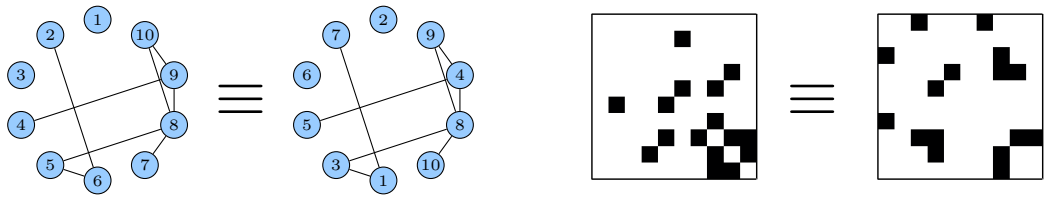


Figure 3.1: Left: Networks with equivalent structure but different node labels. Right: Corresponding adjacency matrix representations of these networks

Relational data are typically stored in arrays; the right hand side of figure 3.1 shows the corresponding adjacency matrix representations of the networks on the left. We demonstrate that exchangeability of the objects underlying a relational database can be expressed in terms of array exchangeability. Prior work on array exchangeability, both theoretical (e.g. Aldous, 1981, 2010; Austin, 2012; Choi and Wolfe, 2013; Diaconis and Janson, 2007; Hoover, 1982, 1979; Kallenberg, 1999; Wolfe and Olhede, 2013) and applied (e.g. Hoff, 2007; Lloyd et al., 2012; Roy and Teh, 2009), has focused on single exchangaeble arrays. We show that the representation theorems for single arrays can be used to derive representations for collections of exchangeable arrays i.e. exchangeable

databases or exchangeable relational data.

3.2 Exchangeable databases

We abstractly define a database following the entity-relationship formalism (e.g. Ullman and Widom, 2002) where the values of attributes are the result of evaluating functions (relations) over a collection of entities / objects.

Definition 3.2.1 (types, signatures, relation)**.** Fix a finite set T of *types*. Define a *signature* to be a finite sequence $s \in T^d$ of types. Define a *relation* r *of signature* $s \in T^d$ *with values in a space* S to be a function from $\mathbb{N}^d$ to S.

We may encode a relation r with signature $s \in T^d$ as an array $X^r := (X^r_k)_{k \in \mathbb{N}^d}$ given by

$$X^r_k = r(k_1, \ldots, k_d), \qquad \text{for } k = (k_1, \ldots, k_d) \in \mathbb{N}^d. \tag{3.2.1}$$

Example 3.2.1. Let $T = \{$*users,movies*$\}$. A relation r of signature $(\mathit{users}, \mathit{movies})$ taking values in $\{1, 2, 3, 4, 5\}$ might record movie ratings. This could be stored in an array, with rows corresponding to some enumeration of *users*, and columns corresponding to some enumeration of *movies*. A relation r' of signature $(\mathit{users}, \mathit{users})$ taking values in $\{0, 1\}$ might store the symmetric friendship relations in a social network.

Definition 3.2.2 (database)**.** Define a *database* to be a collection of R relations $r^1, \ldots, r^R$ of signature $s^1, \ldots, s^R \in T^{d_1}, \ldots, T^{d_R}$, taking values in spaces $S^1, \ldots, S^R$, respectively.

We may encode a database as a collection of arrays $(X^{r^j})_{j=1}^R$, where X^{r^j} encodes the relation r^j. We will often refer to the collection of arrays $(X^{r^j})_{j=1}^R$ as if it were the database itself.

Permuting the ordering of objects within a database can result in a permutation of the indices of several of the arrays encoding its relations. For each type $t \in T$, let $p_t \in \mathbb{S}_\infty$ be a permutation of $\mathbb{N}$. Write $p = (p_t; t \in T) \in \mathbb{S}_\infty^T$ for a collection of such permutations. Given a signature $s \in T^d$, define p^s to be the map from $\mathbb{N}^d$ to $\mathbb{N}^d$ such that

$$p^s(k) := (p_{s_1}(k_1), \ldots, p_{s_d}(k_d)), \qquad \text{for } k \in \mathbb{N}^d. \tag{3.2.2}$$

In other words, p^s maps a sequence $k_1, \ldots, k_d$ of indices (indexing objects of type $s_1, \ldots, s_d$, respectively) to the sequence where each index is permuted by the permutation corresponding to its type.

If X^r is the encoding of a relation r with signature $s \in T^d$, then the permuted relation $r \circ p$ is represented by the array $X^{r \circ p}$ given by

$$X_k^{r \circ p} = X_{p^s(k)}^r, \qquad \text{for } k \in \mathbb{N}^d. \tag{3.2.3}$$

Definition 3.2.3 (exchangeable database)**.** We say that a random database $(X^{r^j})_{j=1}^R$ is *exchangeable* when it has the same distribution as $(X^{r^j \circ p})_{j=1}^R$ for every $p \in \mathbb{S}_\infty^T$.

3.2.1 A simplified representation theorem

The following result characterises the distribution of any exchangeable database to arbitrary accuracy.

Corollary 3.2.1 (simple functional representation for exchangeable databases)**.** *Let $(X^{r^j})_{j=1}^R$ be an exchangeable random database. Then there exists a sequence of random measurable functions $F^{j,1}, F^{j,2}, \ldots$ for every $j = 1, \ldots, R$ and collection of i.i.d. Uniform$[0,1]$ random variables $(U_i^t)_{i \in \mathbb{N}, t \in T}$ such that the random databases $(X^{r^j,n})_{j=1}^R$ converge in distribution to $(X^{r^j})_{j=1 \ldots R}$ on all finite subarrays as $n \to \infty$, where the sequence of arrays $X^{r^j,1}, X^{r^j,2}, \ldots$ for $j = 1, \ldots, R$ are given by*

$$X_k^{r^j,n} := F^{j,n}(U_{k_1}^{s_1^j}, \ldots, U_{k_{d_j}}^{s_{d_j}^j}), \qquad \textit{for } k \in \mathbb{N}^{d_j}. \tag{3.2.4}$$

This is a corollary of theorem 3.3.3, presented later in this chapter, which states that the distributions of finite subarrays are mutually absolutely continuous and the associated Radon–Nikodym derivatives converge uniformly to 1 as $n \to \infty$. We also present an almost-sure representational result which requires some heavy notation which we build up in subsequent sections.

To demonstrate corollary 3.2.1, we present two special cases applicable to modelling a network with side information for each node and a social network with associated user-item data.

Corollary 3.2.2. *Consider an exchangeable database with one object type, one unary relationship, and one binary relationship; denote the binary relationship by the array $X = (X_{ij})_{i,j \in \mathbb{N}}$ and the unary relationship with the sequence $C = (C_i)_{i \in \mathbb{N}}$. Then there*

exists a sequence of pairs of random measurable functions $(F^n, G^n)_{n\in\mathbb{N}}$ *and a collection of i.i.d. Uniform*$[0,1]$ *random variables* $(U_i)_{i\in\mathbb{N}}$ *such that if we define the arrays* $X^1, X^2, \ldots$ *and sequences* $C^1, C^2, \ldots$ *by*

$$X^n_{i,j} := F^n(U_i, U_j), \qquad \text{for } i,j,n \in \mathbb{N}, \tag{3.2.5}$$

$$C^n_i := G^n(U_i), \qquad \text{for } i,n \in \mathbb{N}, \tag{3.2.6}$$

then (X^n, C^n) *converges in distribution to* (X, C) *as* $n \to \infty$ *on all finite subarrays.*

Corollary 3.2.3. *Consider an exchangeable database with two object types, one binary relation between two objects of the first type and a binary relation between objects of different types; denote the first relation by the array* $X = (X_{ij})_{i,j\in\mathbb{N}}$ *and the second by* $Y = (Y_{ik})_{i,k\in\mathbb{N}}$*. Then there exists a sequence of pairs of random measurable functions* $(F^n, G^n)_{n\in\mathbb{N}}$ *and a collection of i.i.d. Uniform*$[0,1]$ *random variables* $(U_i)_{i\in\mathbb{N}}, (V_i)_{i\in\mathbb{N}}$ *such that if we define the arrays* $X^1, X^2, \ldots$ *and* $Y^1, Y^2, \ldots$ *by*

$$X^n_{ij} := F^n(U_i, U_j), \qquad \text{for } i,j,n \in \mathbb{N}, \tag{3.2.7}$$

$$Y^n_{ik} := G^n(U_i, V_k), \qquad \text{for } i,k,n \in \mathbb{N}, \tag{3.2.8}$$

then (X^n, Y^n) *converges in distribution to* (X, Y) *as* $n \to \infty$ *on all finite subarrays.*

Remark 3.2.4 (random functions)**.** When we refer to a random function we mean a deterministic function which takes an additional source of randomness as input. For example $F(\text{arguments}) := f(U, \text{arguments})$ where f is deterministic and U is uniformly distributed would be an example of a random function. This is made precise in the statements of the main results.

3.2.2 Interpretation and examples

Corollary 3.2.1 states that the joint distribution of an exchangeable database can be arbitrarily well approximated by a collection of random measurable functions and uniform random variables. This functional form provides a set of parameters to be estimated that are naturally hierarchical. The functions $(F^{j,n})$ capture properties of entire relations whilst the (U^t_i) represent particular objects underlying the relational data.

Example : Exchangeable networks

Consider modelling a single binary relation which indicates whether or not two nodes in a network are connected or not. This data is typically represented in the form of an adjacency matrix (X_{ij}) where $X_{ij} = 1$ if and only if node i is connected to node j. Theorem 3.3.3 states that if the distribution of X is exchangeable then it can be arbitrarily well approximated by

$$(F(U_i, U_j)) \tag{3.2.9}$$

where F is a random measurable function and (U_i) are i.i.d. Uniform$[0, 1]$ random variables.This special case was used previously by Hoff (2007); Lloyd et al. (2012); Roy and Teh (2009) and in chapter 2 to inspire probabilistic models of networks of the form

$$(U_i) \overset{\text{iid}}{\rightsquigarrow} \text{e.g. Gaussian} \tag{3.2.10}$$

$$F \rightsquigarrow \text{e.g. Gaussian process, bilinear function} \ldots \tag{3.2.11}$$

$$W_{ij} \leftarrow F(U_i, U_j) \tag{3.2.12}$$

$$X_{ij} \rightsquigarrow \text{Bernoulli}(\sigma(W_{ij})). \tag{3.2.13}$$

This is illustrated in figure 3.2; in this case F can be interpreted as a blurred adjacency matrix.

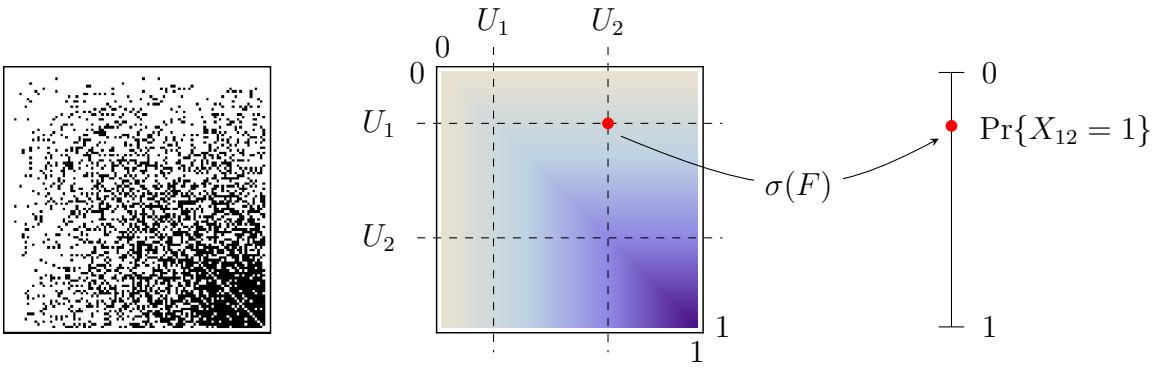


Figure 3.2: Illustration of a model for network data inspired by the Aldous–Hover representation theorem. The left shows a random sample of a binary network (represented by an adjacency matrix) generated by a model of the form given by equation (3.2.13).

Example : A simple database

Consider the database shown on the left hand side of figure 3.3. There are two objects, students and courses, and three relations, the unary relation 'age' acting on students, the binary relation 'friends' acting on pairs of students and the binary relation 'grade' that acts on students and courses. Sample data encoded in arrays is shown at the bottom of this figure.

Exchangeability of this database means that the entries of the leftmost table, the rows and columns of the second table and the rows of the third table can be arbitrarily permuted without changing the distribution of the database when viewed as a random variable. Similarly the columns of the rightmost table may be independently arbitrarily permuted.

The functional form resulting from the application of theorem 3.3.3 or corollary 3.2.1 to this data structure is shown on the right hand side of figure 3.3. The two objects are represented by i.i.d. random variables, (U_i) for students, (V_i) for courses and the three relations are represented by three random functions F, G and H whose inputs are the random variables representing the objects the relations act upon.

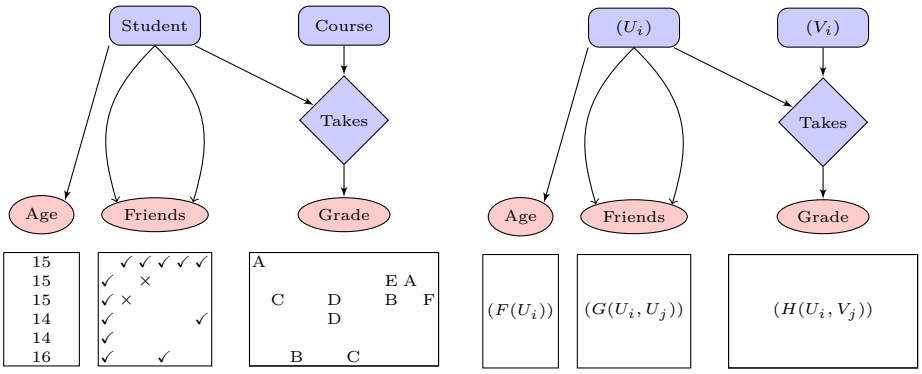


Figure 3.3: Left: A pictorial representation of a relational database. Right: The functional representation of the distribution of data of this form guaranteed to be an arbitrarily good approximation by theorem 3.3.3

3.3 Representation theorems for exchangeable databases

We now state and prove the main technical results of this chapter.

3.3.1 Background: π-exchangeability

We first summarise material from Kallenberg (1999). Let π be a partition of the set $\{1, \ldots, d\}$ and let $\pi_i := (I \in \pi : i \in I)$ i.e. this is the partition element which i is in. Let $p = (p^{\pi_i} : i = 1, \ldots, d)$ be a collection of permutations of $\mathbb{N}$ i.e. a list of permutations which are equal on the partition elements. We say that an array X is π-exchangeable if

$$(X_k) \stackrel{d}{=} (X_{p(k)}) \tag{3.3.1}$$

for all collections p of permutations of $\mathbb{N}^d$ of the form given above. This matches the symmetry of an exchangeable database defined in definition 3.2.3 and equation (3.2.3) if X represents a relation with signature $(\pi_i : i = 1, \ldots, d)$.

We say that a π-exchangeable array X is *simple* if it admits a functional representation of the form

$$X_k = f(U, U^{\pi_1}_{k_1}, \ldots, U^{\pi_d}_{k_d}) \tag{3.3.2}$$

where f is a measurable function and $U, (U^{\pi_j}_{k_j})$ are i.i.d. Uniform$[0, 1]$ random variables.

Proposition 3.3.1 (π-exchangeability representation theorem)**.** *Let X be a π-exchangeable array. Then there exist some simple π-exchangeable arrays X_1 , X_2 , ... such that $\chi_m X_n$ and $\chi_m X$ are mutually absolutely continuous for all $m, n \in \mathbb{N}$ and the associated Radon–Nikodym derivatives tend uniformly to 1 as $n \to \infty$ for fixed m where χ_m is the array subset operation such that $\chi_m Y := (Y_k : k \in \{1, \ldots, m\}^d)$.*

3.3.2 Simple exchangeable database representation theorem

We generalise the concept of a simple π-exchangeable array by saying that an exchangeable database $(X^{r_j})_{j=1}^R$ is simple if it admits a functional representation of the form

$$X^{r^j}_k = f^{r^j}(U, U^{s^j_1}_{k_1}, \ldots, U^{s^j_{d_j}}_{k_{d_j}}) \ \forall \ k \in \mathbb{N}^{d_j} \ \forall \ j \in 1, \ldots, R. \tag{3.3.3}$$

We first state a simple but useful lemma. It can be proven by writing down the definitions of Radon–Nikodym derivatives and uniform convergence.

Lemma 3.3.2. *Suppose that two sequences of random variables (X^n) and (Y^n) are such that X^n and Y^n are mutually absolutely continuous for all n, and the associated Radon–Nikodym derivatives converge uniformly to 1. Then for any measurable f, $f(X^n)$ and $f(Y^n)$ are mutually absolutely continuous for all n and the associated Radon–Nikodym derivatives converge uniformly to 1.*

We also build up some notation for a particular exchangeable database $(X^{r^j})_{j=1}^R$.Assume w.l.o.g. that the types are ordered (e.g. type 1, type 2,...) and the types in each signature are sorted according to this ordering (e.g. $s = (1, 1, 2, 3, 3, \dots)$). Define the multiplicity of a type t in signature s by the number of times the type appears in the signature, and denote this by m_t^s. Define the maximum multiplicity of a type t in a database as $m_t = \max_j m_t^{s^j}$. Let $d = \sum_{t \in T} m_t$ be the sum of the maximum multiplicities. Let π be the partition of $\{1, \dots, d\}$ consisting of consecutive blocks of integers with sizes equal to $\{m_1, \dots, m_T\}$. Let $\rho_j : \mathbb{N}^d \to \mathbb{N}^{d_j}$ be the projection that keeps the first $m_1^{s^j}$ dimensions, ignores the next $m_1 - m_1^{s^j}$, keeps the next $m_2^{s^j}$ et cetera. Similarly, let $\tilde{\rho}_j : \mathbb{N}^d \to \mathbb{N}^d$ be analogous to ρ_j except that the indices removed by ρ_j are now set to the value 1. Again similarly, for a signature s let I_s be the subset of $\{1, \dots, d\}$ that contains the first m_1^s integers and then not the next $m_1 - m_1^s$ integers, then the next m_2^s integers, and then not the next $m_2 - m_2^s$ integers and so on. Let $\mathbb{I}_s$ be the indicator function of the set I_s within $\{1, \dots, d\}$. We can now state and prove the first main result of this chapter.

Theorem 3.3.3 (Simple exchangeable database representation theorem)**.** *Let $(X^{r^j})_{j=1}^R$ be an exchangeable database. Then there exist some simple exchangeable databases $(X^{r^j,1})_{j=1}^R$, $(X^{r^j,2})_{j=1}^R$, ... such that $(\chi_m X^{r^j,n})_{j=1}^R$ and $(\chi_m X^{r^j})_{j=1}^R$ are mutually absolutely continuous for all $m, n \in \mathbb{N}$ and the associated Radon–Nikodym derivatives converge uniformly to 1 as $n \to \infty$ for fixed m.*

Proof. We first embed all of the arrays representing the exchangeable database into one larger array:

$$R_k = (X^{r^j}_{\rho_j(k)} : j = 1, \dots, R) \quad \forall\, k \in \mathbb{N}^d. \tag{3.3.4}$$

By the definition of an exchangeable database, this array is π exchangeable where π is as defined in the preceding text. By proposition 3.3.1 there exists a sequence of measurable functions (f^n) and collection of independent uniform random variables U and (U_i^t) such that $(\chi_m R_k)$ and $(\chi_m f^n(U, U_{k_1}^{\pi_1}, \dots, U_{k_d}^{\pi_d}))$ are mutually absolutely continuous and the associated Radon–Nikodym derivates converge uniformly to 1 as $n \to \infty$ for all m. By using a projection in lemma 3.3.2 we have the same convergence for the following subarrays (jointly over j)

$$f^{n,j}(U, U^{\pi_1}_{\tilde{\rho}_j(k_1)}, \dots, U^{\pi_d}_{\tilde{\rho}_j(k_d)}) \to X^{r^j}_{\rho_j(k)} \tag{3.3.5}$$

and by exchangeability we also have

$$f^{n,j}(U, U^{\pi_1}_{\tilde{\rho}_j(k_1+1)}, \ldots, U^{\pi_d}_{\tilde{\rho}_j(k_d+1)}) \to X^{r^j}_{\rho_j(k)}. \tag{3.3.6}$$

By the Borel isomorphism theorem there is a bimeasurable function h and a uniformly distributed random variable V such that

$$V = h(U, U^1_1, \ldots, U^T_1) \text{ a.s.} \tag{3.3.7}$$

Let $V^t_k = U^t_{k+1}$ and

$$g^{n,j}(V, V^{\pi_i}_{k_i} : i \in I_{s^j}) := f^{n,j}(U, U^{\pi_1}_{\tilde{\rho}_j(k_1+1)}, \ldots, U^{\pi_d}_{\tilde{\rho}_j(k_d+1)}). \tag{3.3.8}$$

g is measurable by composition of measurable f and h. Finally set

$$X^{r_j,n}_k := g^{n,j}(V, V^{\pi_i}_{k_i} : i \in I_{s^j}) \tag{3.3.9}$$

and we are done. □

3.3.3 Almost sure exchangeable database representation theorem

To prove an almost sure representation theorem for exchangeable databases we will require yet more notation; we follow Kallenberg except where we think changes may make it easier to understand. There is only so much we have been able to do to reduce the notational burden so we provide corollaries of the theorem after its statement and proof to demonstrate that behind the notation are some fairly intuitive concepts.

We define for each $k \in \mathbb{N}^d$ and $I \subset \{1, \ldots, d\}$ a set $k_I \subset \mathbb{N}$ by

$$k_I = \{k_i : i \in I\}, \quad k \in \mathbb{N}^d \tag{3.3.10}$$

i.e. k_I only includes the indices in I and is a set.

We further define $k_{\pi I}$ by

$$k_{\pi I} = (k_{I \cap J} : J \in \pi) \tag{3.3.11}$$

N.B. we have replaced Kallenberg's function notation with a list since this may be more natural to the readers of this work. Let $\mathbb{N}^{\leq d}$ be the collection of all subsets of the naturals $K \subset \mathbb{N}$ with cardinality $|K| \leq d$.

Recall that for a signature s we define I_s to be the subset of $\{1, \ldots, d\}$ that contains the first m_1^s integers and then not the next $m_1 - m_1^s$ integers, then the next m_2^s integers, and then not the next $m_2 - m_2^s$ integers and so on.

We refer to indexed collections of i.i.d. Uniform$[0,1]$ random variables as U-arrays.

Proposition 3.3.4 (Kallenberg (1999))**.** *Let X be a random d-array in a Borel space S. Then X is π-exchangeable iff there exists a measurable function $f : [0,1]^{2^d} \to S$ and a U-array ξ with index set $\prod_{J \in \pi} \mathbb{N}^{\leq |J|}$ such that*

$$X_k = f(\xi_{k_{\pi I}} : I \subset \{1, \ldots, d\}) \text{ a.s.}, \quad k \in \mathbb{N}^d. \tag{3.3.12}$$

We now extend this proposition to the case of an exchangeable database. Again we state a useful lemma.

Lemma 3.3.5. *Let $X_{ij} = f(U_i, V_{ij})$ a.s. $i, j \in \mathbb{N}$ where f is measurable and V_{ij} are i.i.d. Suppose further that $X_{ij} = X_i$ a.s. for some collection of random variables $X_i \, \forall \, i, j \in \mathbb{N}$. Then there exists a measurable function g such that $X_i = g(U_i)$ a.s.*

Proof. For each i and conditioned on U_i, X_{ij} are i.i.d. Then by the conditional strong law of large numbers

$$\frac{1}{n} \sum_{j \leq n} f(U_i, V_{ij}) \to \mathbb{E}(f(U_i, V_{ij}) \,|\, U_i) \text{ a.s.} \tag{3.3.13}$$

whence

$$\frac{1}{n} \sum_{j \leq n} f(U_i, V_{ij}) = X_i = \mathbb{E}(f(U_i, V_{ij}) \,|\, U_i) =: g(U_i) \text{ a.s.} \tag{3.3.14}$$

□

Theorem 3.3.6 (almost sure functional representation for exchangeable databases)**.** *A database $(X^{r^j})_{j=1}^R$ is exchangeable iff there exist measurable functions $f^j : [0,1]^{2^{d_j}} \to S^j$ and a U-array ξ with index set $\prod_{J \in \pi} \mathbb{N}^{\leq |J|}$ such that*

$$X^{r^j}_{\rho_j(k)} = f^j(\xi_{k_{\pi I}} : I \subset I_{s^j}) \text{ a.s.}, \quad k \in \mathbb{N}^d, \ j \in \{1, \ldots, R\}. \tag{3.3.15}$$

Proof. It is simple to verify that this construction produces databases with the required symmetry properties so we need only show that such a construction is guaranteed to exist by exchangeability. We proceed as before by embedding all of the arrays into one

larger array

$$R_k = (X^{r^j}_{\rho_j(k)} : j = 1, \ldots, R) \quad \forall\, k \in \mathbb{N}^d \tag{3.3.16}$$

which is π exchangeable and thus by proposition 3.3.4 there exists a measurable g and U-array ξ with index set $\prod_{J\in\pi} \mathbb{N}^{\leq|J|}$ such that

$$R_k = g(\xi_{k_{\pi I}} : I \subset \{1, \ldots, d\}) \text{ a.s.}, \quad k \in \mathbb{N}^d. \tag{3.3.17}$$

We then repeatedly apply lemma 3.3.5 to remove the redundancy of this representation. For example, for some $i \notin I_{s^j}$ we have that there exists some measurable h^j such that

$$h^j(\xi_{k_{\pi I}} : I \subset \{1, \ldots, d\}\backslash i) = X^{r_j}_{\rho_j(k)} \text{ a.s.} \tag{3.3.18}$$

After repeated application we will find that there exists measureable f^j such that

$$f^j(\xi_{k_{\pi I}} : I \subset I_{s^j}) = X^{r_j}_{\rho_j(k)} \text{ a.s.} \tag{3.3.19}$$

and we are done. □

3.3.4 Examples

Corollary 3.3.7. *Consider an exchangeable database with one object type, one unary relationship, and one binary relationship; denote the binary relationship by the array $X = (X_{ij})_{i,j\in\mathbb{N}}$ and the unary relationship with the sequence $C = (C_i)_{i\in\mathbb{N}}$. Then there exist measurable functions (f, g) and a collection of i.i.d. Uniform$[0, 1]$ random variables $U, (U_i)_{i\in\mathbb{N}}, (U_{\{ij\}})_{i,j\in\mathbb{N}}$ such that*

$$X_{ij} = f(U, U_i, U_j, U_{\{ij\}}), \qquad \text{for } i, j \in \mathbb{N}, \tag{3.3.20}$$

$$C_i = g(U, U_i), \qquad \text{for } i \in \mathbb{N}, \tag{3.3.21}$$

almost surely.

This is the Aldous–Hoover representation for a jointly exchangeable array and the de Finetti representation of an exchangeable sequence at the same time, with coupled uniform random variables.

Corollary 3.3.8. *Consider an exchangeable database with two object types, one binary relation between two objects of the first type and a binary relation between objects of*

different types; denote the first relation by the array $X = (X_{ij})_{i,j\in\mathbb{N}}$ *and the second by* $Y = (Y_{ik})_{i,k\in\mathbb{N}}$*. Then there exists a pair of measurable functions* (f, g) *and a collection of i.i.d. Uniform*$[0,1]$ *random variables* $U, (U_i)_{i\in\mathbb{N}}, (V_i)_{i\in\mathbb{N}}, (U_{\{ij\}})_{i,j\in\mathbb{N}}, (W_{ij})_{i,j\in\mathbb{N}}$ *such that*

$$X_{ij} = f(U, U_i, U_j, U_{\{ij\}}), \qquad \text{for } i, j \in \mathbb{N}, \tag{3.3.22}$$

$$Y_{ik} = g(U, U_i, V_k, W_{ik}), \qquad \text{for } i, k \in \mathbb{N}, \tag{3.3.23}$$

almost surely.

This is the Aldous–Hoover representation for a jointly exchangeable array and the Aldous–Hoover representation of a separately exchangeable array at the same time, but with coupled uniform random variables due to the shared objects underlying the data.

Corollary 3.3.9. *Consider an exchangeable database with two object types, one binary relation between two objects of the first type and a ternary relation between two objects of the first type and one of the second type; denote the first relation by the array* $X = (X_{ij})_{i,j\in\mathbb{N}}$ *and the second by* $Y = (Y_{ijk})_{i,j,k\in\mathbb{N}}$*. Then there exists a pair of measurable functions* (F, G) *and a collection of i.i.d. Uniform*$[0,1]$ *random variables* $U, (U_i)_{i\in\mathbb{N}}, (V_i)_{i\in\mathbb{N}}, (U_{\{ij\}})_{i,j\in\mathbb{N}}, (W_{ij})_{i,j\in\mathbb{N}}, (Z_{\{ij\}k})_{i,j,k\in\mathbb{N}}$ *such that*

$$X_{ij} = F(U, U_i, U_j, U_{\{ij\}}), \qquad \text{for } i, j \in \mathbb{N}, \tag{3.3.24}$$

$$Y_{ijk} = G(U, U_i, U_j, V_k, U_{\{ij\}}, W_{ik}, W_{jk}, Z_{\{ij\}k}), \qquad \text{for } i, j, k \in \mathbb{N}, \tag{3.3.25}$$

almost surely.

This is the Aldous–Hoover representation for a jointly exchangeable array and the Kallenberg representation of a π-exchangeable array, where $\pi = \{\{1, 2\}, 3\}$, at the same time, but again with shared uniform random variables.

3.4 A generic statistical model template

In analogy to the work of Hoff (2007); Lloyd et al. (2012); Roy and Teh (2009) on exchangeable arrays, theorem 3.3.3 naturally inspires a generic generative model of exchangeable databases. Each object of type t in the database is associated with an i.i.d. sample, U_i^t, from some distribution $\mathcal{U}$ e.g. Uniform, Gaussian. For each relation r^j we sample a random function F^j from some distribution $\mathcal{F}^j$, e.g. Gaussian process, random

(bi/tri/...)linear functions. We denote the evaluation of these functions at the corresponding values of (U_i^t) by W^j. W^j can then be passed through a link function and distribution $L^j(\cdot)$ to model the observed value of the relation r^j.

$$(U_i^t) \overset{\text{iid}}{\rightsquigarrow} \mathcal{U} \tag{3.4.1}$$
$$F^j \rightsquigarrow \mathcal{F}^j \tag{3.4.2}$$
$$W_k^j \leftarrow F^j(U_{k_1}^{s_1^j}, \cdots, U_{k_{d_j}}^{s_{d_j}^j}) \tag{3.4.3}$$
$$X_k^{r_j} \rightsquigarrow L^j(W_k^j) \qquad \text{independently across } j \text{ and } k. \tag{3.4.4}$$

Remark 3.4.1 (dependence between functions)**.** In general, the functions F^j may be dependent. However, the representation results presented in this chapter provide no guidance on the form of this dependence; stronger assumptions than exchangeability would have to be made. In practice one may model the functions to be independent a priori at the risk of wasting statistical strength.

3.4.1 Deriving appropriate forms for conditional models

Consider a social network with side information on each user. The natural generative model for such data inspired by the simple database representation result is

$$U_i \overset{\text{iid}}{\rightsquigarrow} \mathcal{U} \tag{3.4.5}$$
$$F \rightsquigarrow \mathcal{F} \tag{3.4.6}$$
$$G \rightsquigarrow \mathcal{G} \tag{3.4.7}$$
$$X_{ij} \rightsquigarrow L^X(F(U_i, U_j)) \tag{3.4.8}$$
$$C_i \rightsquigarrow L^C(G(U_i)) \tag{3.4.9}$$

but in the situation where C is fully observed and the task is to predict missing entries of X it may be unnecessary to learn a model of C. Indeed, if C is assumed to be noiselessly observed, then we have $C_i = L^C(G(U_i))$ where $L^C(G(.))$ is a deterministic function. We can then choose to model X as follows

$$F'(U_i, U_j, C_i, C_j) = F'(U_i, U_j, L^C(G(U_i)), L^C(G(U_j))) = F(U_i, U_j) \tag{3.4.10}$$
$$X_{ij} \rightsquigarrow L^X(F'(U_i, U_j, C_i, C_j)). \tag{3.4.11}$$

Furthermore we could choose F' to not depend on U_i, U_j resulting in the model

$$X_{ij} \rightsquigarrow L^X(F''(C_i, C_j)) \tag{3.4.12}$$

which is simply a regression model.

Indeed, the same logic can be applied to exchangeable sequences to derive more standard regression models. Suppose that (X_i, Y_i) is an exchangeable sequence. By de Finetti's theorem we can model this as

$$U_i \rightsquigarrow \mathcal{U} \tag{3.4.13}$$
$$F \rightsquigarrow \mathcal{F} \tag{3.4.14}$$
$$G \rightsquigarrow \mathcal{G} \tag{3.4.15}$$
$$X_i \rightsquigarrow L^X(F(U_i)) \tag{3.4.16}$$
$$Y_i \rightsquigarrow L^Y(G(U_i)) \tag{3.4.17}$$

and by similar arguments to those above, if we were to assume that X was noiselessly observed we could model Y as

$$Y_i \rightsquigarrow L^Y(G(X_i)) \tag{3.4.18}$$

which is a standard regression model, or

$$Y_i \rightsquigarrow L^Y(G(U_i, X_i)) \tag{3.4.19}$$

which combines latent variable modelling and regression. An example of this hybrid model structure can be found in Wang and Neal (2012) but is surprisingly uncommon.

Now suppose we perfectly observe a binary relation; what can this tell us about modelling a related unary relation? We now demonstrate how we could (almost) reconstruct an example model presented in Friedman et al. (1999). They present a simple genetic model where one's maternal chromosome depends on one's mother's maternal and paternal chromosomes and similarly for one's paternal chromosome. In this example we have two binary relations, mother-of and father-of and two unary relations, maternal and paternal chromosomes. We write this as $X_{ij} = 1 \iff$ person i is the mother of person j and $Y_{ij} = 1 \iff$ person i is the father of person j and the array elements take the value 0 otherwise. Further let M_i and P_i be the maternal and paternal chromosomes

of person i respectively. A potential form of a model for this data is

$$U_i \rightsquigarrow \mathcal{U} \tag{3.4.20}$$
$$F \rightsquigarrow \mathcal{F} \tag{3.4.21}$$
$$G \rightsquigarrow \mathcal{G} \tag{3.4.22}$$
$$H \rightsquigarrow \mathcal{H} \tag{3.4.23}$$
$$J \rightsquigarrow \mathcal{J} \tag{3.4.24}$$
$$X_{ij} \rightsquigarrow L^X(F(U_i, U_j)) \tag{3.4.25}$$
$$Y_{ij} \rightsquigarrow L^Y(G(U_i, U_j)) \tag{3.4.26}$$
$$M_i \rightsquigarrow L^M(H(U_i)) \tag{3.4.27}$$
$$P_i \rightsquigarrow L^P(J(U_i)) \tag{3.4.28}$$

Applying similar arguments to before we can replace U_i by $U_i' := (U_i, V_i, M_i, P_i)$ where we have split the latent variable U_i into U_i and V_i for convenience. We could then define

$$L^X(F(U_i', U_j')) = \begin{cases} 1 & \text{if } (M_i, P_i) = U_j \\ 0 & \text{otherwise} \end{cases} \tag{3.4.29}$$

$$L^Y(G(U_i', U_j')) = \begin{cases} 1 & \text{if } (M_i, P_i) = V_j \\ 0 & \text{otherwise} \end{cases} \tag{3.4.30}$$

Loosely speaking, if we performed inference in this model we would then find that in the posterior $U_i = (M_j, P_j)$ where person j is the mother of person i and similarly for V_i. This would then allow the functions H and J to specify that the maternal chromosome of person i depends probabilistically on the chromosomes of their mother and similarly for their paternal chromosome, recovering the structure of the probabilistic relational model.

However, the alert reader will have noticed that the functions $L^X(F(.,.))$ and $L^Y(G(.,.))$ defined above are almost everywhere zero. Indeed the event $U_i = (M_j, P_j)$ could have measure zero and therefore cannot be said to be inferred without properly defining a suitable limiting process of valid inferences (see e.g. Jaynes (2003) for many good examples where casual handling of measure zero events can lead to 'paradoxes'). The functions $L^X(F(.,.))$ and $L^Y(G(.,.))$ are almost everywhere zero exactly because the relations mother-of and father-of define sparse graphs with in-degrees of at most one. As remarked in chapter 2, any not almost everywhere zero representing function will

imply density when modelling a graph. We repeat that the construction of general representation results for sparse graphs is an area of emerging research and refer the reader to the sparse literature (Caron and Fox, 2014; Lovász, 2012; Wolfe and Olhede, 2013).

As before with the regression example, it may be wise to use models which use both observed and hidden variables within the representing functions. Examples are again rare but this has been applied in e.g. the movie recommendation context (e.g. Menon, 2011).

3.4.2 Higher order dependencies

So far we have only considered probabilistic models inspired by the simple array versions of exchangeability theorems. Do the almost sure representation theorems, with their more detailed latent variable representations, tell us anything useful? Consider the following dataset: we have a social network X_{ij} and a ternary relation Y_{ijk} that records if both person i and person j have been to location k. The simple array inspired representation of this data is of the form

$$X_{ij} \;\leftsquigarrow\; L^X(F(U_i, U_j)) \tag{3.4.31}$$

$$Y_{ijk} \;\leftsquigarrow\; L^Y(G(U_i, U_j, V_k)) \tag{3.4.32}$$

and in contrast the almost sure result would suggest a more elaborate model of the form

$$X_{ij} \;\leftsquigarrow\; L^X(F(U_i, U_j, U_{\{ij\}})) \tag{3.4.33}$$

$$Y_{ijk} \;\leftsquigarrow\; L^Y(G(U_i, U_j, U_{\{ij\}}, V_k, W_{ik}, W_{jk}, Z_{\{ij\}k})). \tag{3.4.34}$$

Now suppose that X included information on the type of relations in the social network, and that people i and j were in a romantic relationship. This knowledge would make it more likely for this couple to visit e.g. restaurants typically frequented by couples. In the almost sure representation this information could easily be transferred from X to the modelling of Y via the shared latent variable $U_{\{ij\}}$. In the simple array representation this knowledge would somehow have to be encoded in the latent variables U_i and U_j meaning that the problem of sharing this information is at least as hard as producing an effective generative model of which pairs of people are in a relationship[1].

We are however unaware of any standard probabilistic models making use of this type of dependence. This is most probably due to the potentially large computational

[1]Presumably a problem that has been well studied by online dating services.

barrier to storing and inferring the values of latent variables for each pair of objects. There may however be use in this type of model on small datasets where one can afford relatively high computational costs to perform a more detailed analysis.

3.4.3 A brief word about longitudinal data

It is worth mentioning how the modelling paradigms presented here can be extended to longitudinal data (i.e. time varying). For example, consider longitudinal measurements of a social network X_{ij}^t. Let $Y_{ij} := (X_{ij}^t : t \in \mathcal{T})$ where $\mathcal{T}$ represents some period of time. Then Y_{ij} can be assumed to be exchangeable meaning that we can represent its distribution as

$$Y_{ij} = G(U_i, U_j) \tag{3.4.35}$$

which in turn means

$$X_{ij}^t = G^t(U_i, U_j) \tag{3.4.36}$$

or more conventionally

$$X_{ij}^t = F(U_i, U_j, t) \tag{3.4.37}$$

meaning that we can represent the distribution of the time evolving social network with fixed latent variables for each node but a time varying representing function. The form of the time varying function is completely general, but it could be constrained further by assumptions of continuity or Markov exchangeability for example. It is more typical however to assume that the latent variables also evolve through time (this still produces an exchangeable distribution) with the representing function potentially static. Examples of models of this form include (e.g. Adams et al., 2010; Durante and Dunson, 2014). It may be advantageous however to have fixed latent variables when one is jointly modelling both time evolving and static data.

3.4.4 Prior work using models of this form

In section 2.4 it was demonstrated that many models of single 2-arrays fit the form of the generic model presented above. In particular there are models that assume F is linear (e.g. Hoff, 2007; Meeds et al., 2007; Miller et al., 2009; Salakhutdinov and Hinton,

2008; Yu and Chu, 2008), that F is Gaussian process distributed (e.g. Lawrence and Urtasun, 2009; Lloyd et al., 2012; Yan et al., 2011) and other non-linear forms for F both parametric (e.g. Hoff et al., 2002) and nonparametric (e.g. Roy and Teh, 2009). In addition to this there has been a line of work that uses increasingly more expressive forms of the distribution $\mathcal{U}$ (e.g. Hoffman et al., 1998; Kemp et al., 2006; Meeds et al., 2007; Miller et al., 2009; Nowicki and Snijders, 2001; Palla et al., 2012; Wang and Wong, 1987; Xu et al., 2006).

Many, but not all, of these models have been extended to model d-arrays. A summary of models using linear forms of F is given in Kolda and Bader (2009); non-linear models include Xu et al. (2012).

For full databases, the literature is limited to clustering / block / latent class models (Kemp et al., 2006; Xu et al., 2006) and models using linear forms for the F^r (e.g. Acar et al., 2011, 2012, 2013; Andersen et al., 2013; Ermis et al., 2012; Gao et al., 2011; Jimeng et al., 2009; Lippert et al., 2008; Nickel, 2011; Shangguan et al., 2012; Singh and Gordon, 2008a,b, 2012; Yin et al., 2013).

We should certainly expect to see new research using some of the more advanced forms proposed for networks being applied to higher order arrays or full databases. However, rather than proposing yet more specific models it would be very interesting to see automated model building techniques applied to these domains, potentially combined with model forms inspired by probabilistic relational models.

3.5 Discussion

We have demonstrated how the concept of exchangeability can be applied to databases and used to derive a natural parameter space for statistical models of such data. Identifying a parameter space is the first step in many statistical analyses, allowing either frequentist estimation of the parameters or Bayesian prior specification. This concept is well established for exchangeable sequences where de Finetti's theorem applies. For exchangeable arrays, the relevant representation theorems were presented by Aldous (1981) and Hoover (1979) over 30 years ago but it is only recently that these results are being used to inspire probabilistic models (Hoff, 2007; Lloyd et al., 2012; Roy and Teh, 2009) and estimation procedures with frequentist guarantees (Choi and Wolfe, 2013; Kallenberg, 1999; Wolfe and Olhede, 2013). We hope that this work will continue and be extended to the analysis of exchangeable databases.

Chapter 4

Compositionally constructed kernels for Gaussian process models

In the introduction we demonstrated that Gaussian processes can be used to express probability distributions over functions. The properties of these distributions very much depended on the choice of kernel; different kernels where shown to yield probability distributions over smooth, linear and periodic functions. In this chapter we address the problem of selecting which kernel to use in a Gaussian process model. Our approach is to define an open ended space of potential kernel functions via a generative grammar and then perform model selection combined with a simple search strategy to explore the space of models.

This chapter is based upon joint work with David Duvenaud, Roger Grosse, Joshua Tenenbaum and Zoubin Ghahramani (Duvenaud et al., 2013). The initial ideas were formed during discussions between Roger, David and myself. I performed the majority of experiments; all were involved with the writing.

4.1 Introduction

Kernel-based nonparametric models, such as support vector machines and Gaussian processes (GPs), have been one of the dominant paradigms for supervised machine learning over the last 20 years. These methods depend on defining a kernel function, $k(x, x')$, which specifies how similar or correlated outputs y and y' are expected to be at two inputs x and x'. By defining the measure of similarity between inputs, the kernel determines the pattern of inductive generalisation.

However, to apply existing kernel learning algorithms, the user must typically specify

the parametric form of the kernel, and this can require considerable expertise and / or several iterations of trial and error.

To make kernel learning more generally applicable we propose an automatic procedure to select an appropriate parametric form of a kernel. In particular, we formulate a space of kernel structures defined compositionally in terms of sums and products of a small number of base kernels. This produces an expressive modelling language which concisely captures many widely used techniques for constructing kernels. We focus on Gaussian process regression, where the kernel specifies a covariance function, since the Bayesian framework provides natural methods by which to select between models. Borrowing discrete search techniques which have proved successful in equation discovery (Todorovski and Dzeroski, 1997) and unsupervised learning (Grosse et al., 2012), we automatically search over this space of kernel structures using model evidence as the search criterion.

We found that our kernel search algorithm is able to automatically recover known structures from synthetic data as well as plausible structures for a variety of real-world datasets. On a variety of time series datasets, the learned kernels yield decompositions of the unknown function into interpretable components that enable accurate extrapolation beyond the range of the observations. Furthermore, the automatically discovered kernels outperform a variety of widely used kernel classes and kernel combination methods on supervised prediction tasks.

While we focus on Gaussian process regression, we believe our kernel search method can be extended to other supervised learning frameworks such as classification or ordinal regression, or to other kinds of kernel architectures such as kernel SVMs. We hope that the algorithm developed in this chapter and the refinements made in chapter 5 will help replace the current and often opaque art of kernel engineering with a more transparent science of automated kernel construction.

4.2 Expressing high level patterns with kernels

Gaussian process regression models use a kernel to define the covariance between any two function values: $\mathrm{Cov}(y, y') = k(x, x')$. The kernel specifies which structures are likely under the GP prior, which in turn determines the generalisation properties of the model. In this section, we review the ways in which kernel families[1] can be composed

to express diverse probability distributions over functions.

There has been significant work on constructing kernels for use with Gaussian processes (e.g. chapter 4 of Rasmussen and Williams, 2006). Commonly used kernel families include the squared exponential (SE)[2], periodic (Per), linear (Lin), and rational quadratic (RQ) which are defined as follows:

$$\mathrm{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) \tag{4.2.1}$$

$$\mathrm{Per}(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi(x-x')/p)}{\ell^2}\right) \tag{4.2.2}$$

$$\mathrm{Lin}(x, x') = \sigma_b^2 + \sigma_v^2(x-\ell)(x'-\ell) \tag{4.2.3}$$

$$\mathrm{RQ}(x, x') = \sigma^2\left(1 + \frac{(x-x')^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{4.2.4}$$

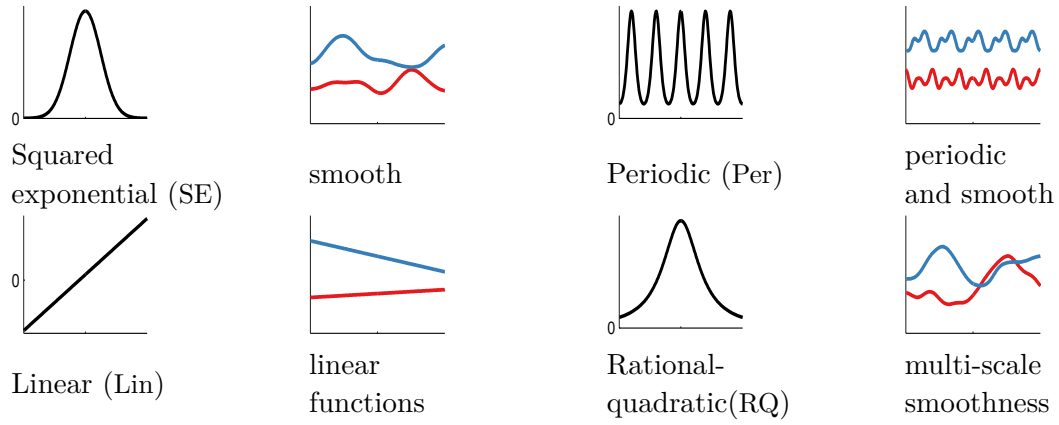


Figure 4.1: First and third columns: base kernels $k(\cdot, 0)$. Second and fourth columns: samples from a GP from each kernel. The x-axis has the same range on all plots.

Composing Kernels Positive semidefinite kernels (i.e. those which define valid covariance functions) are closed under addition and multiplication[3]. We can therefore use these two operations to create new kernels from old.

[1]When unclear from context, we use 'kernel family' to refer to the parametric forms of the kernel functions. A kernel is a kernel family with all of the parameters specified.

[2]This kernel family is also known as the exponentiated quadratic and radial basis function. I prefer the term exponentiated quadratic and recommend others to use it in future, but old habits die hard.

[3]And other operations (e.g. Rasmussen and Williams, 2006)

In particular, kernels over multidimensional inputs can be constructed by summation and multiplication of kernels acting on single dimensions of the input. We therefore define our base kernel families to be one dimensional where we indicate the dimension the kernel applies to using subscripts e.g. SE_2 represents an SE kernel over the second dimension of the input x.

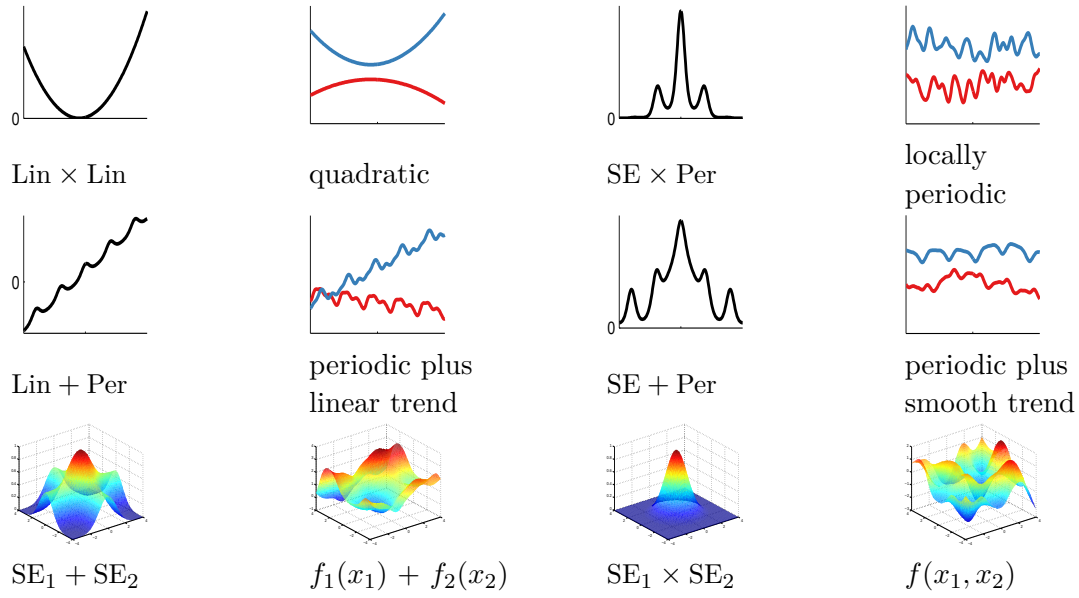


Figure 4.2: Examples of structures expressible by composite kernels. First and third columns: composite kernels $k(\cdot, 0)$. Plots have the same meaning as in figure 4.1.

Summation Suppose functions f_1 and f_2 are samples from independent GP distributions, $f_1 \sim \mathrm{GP}(\mu_1, k_1)$, $f_2 \sim \mathrm{GP}(\mu_2, k_2)$. Then $f := f_1 + f_2 \sim \mathrm{GP}(\mu_1 + \mu_2, k_1 + k_2)$. This allows us to create kernels that encode for distributions over superpositions of independent functions with different properties.

In a single dimension, sums of functions / kernels can express superposition of different processes, possibly operating at different scales. In multiple dimensions, summing kernels gives additive structure over different dimensions, similar to generalized additive models (Hastie and Tibshirani, 1990). These two kinds of structure are demonstrated in rows 2 and 3 of figure 4.2, respectively.

Multiplication Multiplying kernels allows us to account for interactions between different input dimensions or different notions of similarity. For example, for multidimensional data, the multiplicative kernel $\mathrm{SE}_1 \times \mathrm{SE}_3$ encodes for a distribution over functions

that vary smoothly with input dimensions 1 and 3 but is not constrained to be additive. In one dimension, multiplying a kernel by SE removes any long range correlations in the distribution it encodes for. For example, Per corresponds to globally periodic structure, whereas Per × SE corresponds to locally periodic structure, as shown in row 1 of figure 4.2. This is made more precise in chapter 5.

Many architectures for learning complex functions, such as convolutional neural networks (e.g. LeCun et al., 1989) or sum-product networks (Poon and Domingos, 2011), include units which compute AND-like and OR-like operations. The composite kernels defined here can be viewed similarly. A sum of kernels can be understood as an OR-like operation: two points are considered similar if either kernel has a high value. Similarly, multiplying kernels is an AND-like operation, since two points are considered similar only if both kernels have high values. Since we are applying these operations to the similarity functions rather than the regression functions themselves, compositions of even a few base kernels are able to capture complex relationships in data which do not have a simple parametric form.

Example expressions In addition to the examples given in figure 4.2, many common motifs of supervised learning can be captured using sums and products of one-dimensional base kernels:

Bayesian linear regression	Lin
Bayesian polynomial regression	$\text{Lin} \times \text{Lin} \times \ldots$
Periodic function decomposition	$\text{Per} + \text{Per} + \ldots$
Additive smoothing	$\sum_{d=1}^{D} \text{SE}_d$
Automatic relevance determination	$\prod_{d=1}^{D} \text{SE}_d$
Linear regression with correlated residuals	Lin + SE

4.3 Searching over kernel families

We have demonstrated that we can construct a wide variety of kernel structures compositionally by adding and multiplying a small number of base kernels. In particular, we consider four base kernel families: SE, Per, Lin, and RQ. Any algebraic expression combining these kernels using the operations + and × defines a kernel family.

Our search procedure begins by proposing all base kernel families applied to all input dimensions. We propose the following search operators over the algebra of kernel families we have just defined:

(1) Any subexpression $\mathcal{S}$ can be replaced with $\mathcal{S}+\mathcal{B}$, where $\mathcal{B}$ is any base kernel family.

(2) Any subexpression $\mathcal{S}$ can be replaced with $\mathcal{S} \times \mathcal{B}$, where $\mathcal{B}$ is any base kernel family.

(3) Any base kernel $\mathcal{B}$ may be replaced with any other base kernel family $\mathcal{B}'$.

These operators can generate all possible algebraic expressions. To see this, observe that if we restricted the $+$ and $\times$ rules only to apply to base kernel families, we would obtain a context-free grammar (CFG) which generates the set of algebraic expressions. However, the more general versions of these rules allow more flexibility in the search procedure, which is useful because the CFG derivation may not be the most straightforward way to arrive at a kernel family.

Our algorithm searches over this space using a greedy search: at each stage, we choose the highest scoring kernel and expand it by applying all possible operators. Our search operators are motivated by strategies researchers often use to construct kernels or other modularly constructed probabilistic models. In particular,

- One can look for patterns, e.g. periodicity, in the residuals of a model, and then extend the model to capture this pattern. This corresponds to applying rule (1).

- One can start with a pattern, e.g. linearity, which is assumed to hold globally, but find that it only holds locally. This corresponds to applying rule (2) to obtain patterns like those shown in the top right of figure 4.2.

- One can add features incrementally, analogous to algorithms like boosting, backfitting, or forward selection. This corresponds to applying rules (1) or (2) to dimensions not yet included in the model.

This potentially suboptimal search strategy was chosen for its simplicity and previously demonstrated empirical performance (Grosse et al., 2012). We discuss its shortcomings and potential improvements in the discussion in section 4.8.

Scoring kernel families Choosing kernel families within the greedy search requires a criterion for evaluating them. We choose model evidence as our criterion, since it balances the fit and complexity of a model and implements what is known as "Bayesian Occam's Razor" (e.g. MacKay, 2003; Rasmussen and Ghahramani, 2001). Conditioned on kernel parameters, the model evidence of a GP can be computed analytically (e.g. Rasmussen and Williams, 2006). However, to evaluate the model evidence of an entire kernel family we must integrate over kernel parameters (after having also defined a prior

distribution over them). We approximate this intractable integral with the Bayesian information criterion (Schwarz, 1978) after first optimising to find the maximum-marginal-likelihood kernel parameters by conjugate gradient descent. For each optimised model, M, the Bayesian Information Criterion (BIC) is calculated as follows:

$$\mathrm{BIC}(M) = -2 \log p(D \mid M) + |M| \log n \tag{4.3.1}$$

where $|M|$ is the number of kernel parameters, $p(D|M)$ is the marginal likelihood of the data, D, and n is the number of data points. We note however that using the Bayesian information criterion is not theoretically justified in this context; but attempts to use a theoretically justified approximation to the marginal likelihood uncover interesting fundamental problems which we discuss in section 4.8.

Unfortunately, optimising over parameters is not a convex optimisation problem, and the space can have many local optima. For example, in data with periodic structure, integer multiples of the true period (i.e. harmonics) are often local optima. To alleviate this difficulty, we take advantage of our search procedure to hot start the optimisation problem: all of the parameters which were part of the previous kernel are initialised to their previous values. All parameters are then optimised using conjugate gradients, randomly restarting the newly introduced parameters. This procedure is not guaranteed to find the global optimum, but it is similar to the commonly used heuristic of iteratively modelling residuals.

4.4 Related Work

4.4.1 Nonparametric regression in high dimensions

Nonparametric regression methods such as smoothing splines, locally weighted regression, and GP regression are popular because they are capable of learning arbitrary smooth functions of the data. Unfortunately, they suffer from the curse of dimensionality: it is very difficult for the basic versions of these methods to generalise well in more than a few dimensions. Applying nonparametric methods in high-dimensional spaces can require imposing additional structure on the model.

One such structure is additivity. Generalised additive models (GAM) assume the regression function is a transformed sum of functions defined on the individual dimensions: $\mathbb{E}[f(\mathbf{x})] = g^{-1}(\sum_{d=1}^{D} f_d(x_d))$. These models have a limited compositional form, but one which is interpretable and often generalises better than an analogous but un-

constrained model. In our grammar, we can capture analogous structure through sums of base kernels along different dimensions.

It is possible to add more flexibility to additive models by considering higher-order interactions between different dimensions. Additive Gaussian processes (Duvenaud et al., 2011) are a GP model whose kernel implicitly sums over all possible products of one-dimensional base kernels; this large summation is efficiently computed at the expense of tying together various parameters of the base kernels. Plate (1999) constructs a GP with a composite kernel, summing an SE kernel along each dimension, with an SE-ARD kernel (i.e. a product of SE over all dimensions). Both of these models can be expressed in our grammar.

A closely related procedure is smoothing-splines ANOVA (Gu, 2002; Wahba, 1990, 2004). This model is a linear combination of smoothing splines along each dimension, all pairs of dimensions, and possibly higher-order combinations. Since the number of terms to consider grows exponentially in the order, in practice, only terms of first and second order are usually considered.

Semiparametric regression (e.g. Ruppert et al., 2003) attempts to combine interpretability with flexibility by building a composite model out of an interpretable, parametric part (such as linear regression) and a 'catch-all' nonparametric part (such as a GP with a SE kernel). In our approach, this can be represented as a sum of SE and Lin.

4.4.2 Kernel learning

There is a large body of work attempting to construct composite kernels defined as a weighted sum of base kernels (e.g. Bach, 2009; Christoudias et al., 2009). While these approaches find the optimal solution in polynomial time, speed comes at a cost: the component kernels, as well as their parameters, are typically specified in advance.

Another approach to kernel learning is to learn an embedding of the data points such that the mapping between the embedding locations and the regression function is simple (e.g. linear or smooth). Lawrence (2005) learns an embedding of the data into a low-dimensional space, and constructs a fixed kernel structure over that space. This model is typically used in unsupervised tasks and requires an expensive integration or optimisation over potential embeddings when generalising to test points. Salakhutdinov and Hinton (2008) use a deep neural network to learn an embedding; this is a flexible approach to kernel learning but relies upon finding structure in the input density, p(x). Instead we focus on domains where most of the interesting structure is in the regression function f(x).

Sparse spectrum GPs (Lázaro-Gredilla and Quiñonero-Candela, 2010) approximate the spectral density of a stationary kernel function using delta functions; this corresponds to kernels of the form $\sum \cos$. Similarly, Wilson and Adams (2013) introduce spectral mixture kernels which approximate the spectral density using a scale-location mixture of Gaussian distributions corresponding to kernels of the form $\sum \mathrm{SE} \times \cos$. Both demonstrate, using Bochner's theorem (Bochner et al., 1959), that these kernels can approximate any stationary covariance function. In chapter 5 we introduce a new version of the periodic kernel for which $\cos(x - x')$ is a limiting form and hence we also attain this completeness property.

Rasmussen and Williams (2006) devote 4 pages to manually constructing a composite kernel to model a time series of carbon dioxide concentrations. When applied to this dataset our procedure chose a model similar to the one they constructed by hand. Other examples of papers whose main contribution is to manually construct and fit a composite GP kernel are Klenske et al. (2013) and Lloyd (2013).

Bing et al. (2010); Diosan et al. (2007) and Kronberger and Kommenda (2013) learn composite kernels for support vector machines, relevance vector machines and Gaussian processes, using genetic search algorithms. We go beyond these works by demonstrating the interpretability of the structure implied by composite kernels, and how such structure allows for extrapolation (these claims are more thoroughly justified in chapter 5).

4.4.3 Structure discovery

There have been several attempts to uncover the structural form of a dataset by searching over a grammar of structures. For example, Schmidt and Lipson (2009), Todorovski and Dzeroski (1997) and Washio et al. (1999) attempt to learn parametric forms of equations to describe time series, or relations between quantities. Since we learn expressions describing the covariance structure rather than the functions themselves, we are able to capture patterns which do not have a simple parametric form.

Kemp and Tenenbaum (2008) learned the structural form of a graph used to model human similarity judgments. Examples of graphs included planes, trees, and cylinders. Some of their discrete graph structures have continuous analogues in our own space; e.g. $\mathrm{SE}_1 \times \mathrm{SE}_2$ and $\mathrm{SE}_1 \times \mathrm{Per}_2$ can be seen as mapping the data to a plane and a cylinder, respectively (e.g. Duvenaud, 2014).

Grosse et al. (2012) performed a greedy search over a compositional model class for unsupervised learning, using a grammar and a search procedure which parallel our own. This model class contained a large number of existing unsupervised models as special

cases and was able to choose appropriate models automatically from data. Our work is tackling a similar problem, but in a supervised setting.

4.5 Pattern discovery in time series

To investigate our method's ability to discover patterns, we ran the kernel search on several time-series to a depth of 10. As discussed in section 4.2, a GP whose kernel is a sum of kernels can be viewed as a sum of functions drawn from independent component GPs. In particular, all kernels in our search space can be equivalently written as sums of products of base kernels by applying distributivity of multiplication over addition. For example,

$$\mathrm{SE} \times (\mathrm{RQ} + \mathrm{Lin}) = \mathrm{SE} \times \mathrm{RQ} + \mathrm{SE} \times \mathrm{Lin}. \tag{4.5.1}$$

This allows us to visualise the learned models as a sum of functions for which we now present the relevant formulae.

Posterior decomposition We can analytically decompose a GP posterior distribution over additive components using the following identity: The conditional distribution of a Gaussian vector $\boldsymbol{f}_1$ conditioned on its sum with another Gaussian vector $\boldsymbol{f} = \boldsymbol{f}_1 + \boldsymbol{f}_2$ where $\boldsymbol{f}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{K}_1)$ and $\boldsymbol{f}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \mathbf{K}_2)$ is given by

$$\begin{aligned} \boldsymbol{f}_1 | \boldsymbol{f} \sim \mathcal{N}\Big(&\boldsymbol{\mu}_1 + {\mathbf{K}_1}^{\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\left(\boldsymbol{f} - \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\right), \\ &\mathbf{K}_1 - {\mathbf{K}_1}^{\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\mathbf{K}_1\Big). \end{aligned}$$

Mauna Loa atmospheric $\mathbf{CO_2}$ measurements Using our method, we analysed records of carbon dioxide levels recorded at the Mauna Loa observatory. Since this dataset was analysed in detail by Rasmussen and Williams (2006), we can compare the kernel chosen by our method to a kernel constructed by human experts.

Figure 4.3 shows the posterior mean and variance on this dataset as the search depth increases. While the data can be smoothly interpolated by a single base kernel model, the extrapolations improve dramatically as the increased search depth allows more structure to be included.

Figure 4.4 shows the final model chosen by our method, together with its decomposition into additive components. The final model exhibits both plausible extrapolation

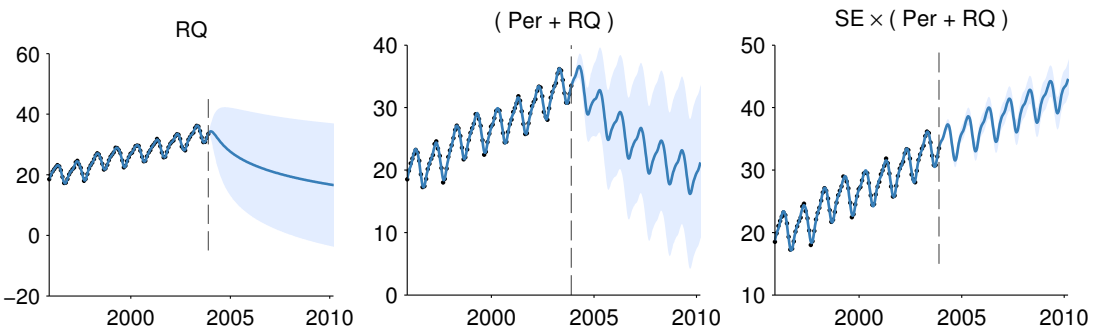


Figure 4.3: Posterior mean and variance for different depths of kernel search. The dashed line marks the extent of the dataset. In the first column, the function is only modelled as a locally smooth function, and the extrapolation is poor. Next, a periodic component is added, and the extrapolation improves. At depth 3, the kernel can capture most of the relevant structure, and is able to extrapolate reasonably.

and interpretable components: a long-term trend, annual periodicity and medium-term correlated deviations; the same components chosen by Rasmussen and Williams (2006). We also plot the residuals, observing that there is little obvious structure left in the data. On this example, our search procedure is able to automate this construction where previously two GP experts devoted 4 pages of text and analysis[4] to the development of a composite kernel.

Airline passenger data Figure 4.6 shows the decomposition produced by applying our method to monthly totals of international airline passengers (e.g. Box et al., 1976). We observe similar components to the previous dataset: a long term trend, annual periodicity and medium-term deviations. In addition, the composite kernel captures the near-linearity of the long-term trend, and the linearly growing amplitude of the annual oscillations.

Solar irradiance Data Finally, we analysed annual solar irradiation data from 1610 to 2011 (Lean et al., 1995). The posterior and residuals of the learned kernel are shown in figure 4.5. None of the models in our search space are capable of parsimoniously representing the lack of variation from 1645 to 1715. Despite this, our approach fails gracefully: the learned kernel still captures the periodic structure, and the quickly growing posterior variance demonstrates that the model is uncertain about long term structure. In chapter 5 we demonstrate that an expanded space of kernel families is able to

[4]Admittedly this was intended as reference material

accurately model this phenomenon.

4.6 Validation on synthetic data

True Kernel	D	SNR = 10	SNR = 1	SNR = 0.1
$\mathrm{SE} + \mathrm{RQ}$	1	SE	$\mathrm{SE} \times \mathrm{Per}$	SE
$\mathrm{Lin} \times \mathrm{Per}$	1	$\mathrm{Lin} \times \mathrm{Per}$	$\mathrm{Lin} \times \mathrm{Per}$	SE
$\mathrm{SE}_1 + \mathrm{RQ}_2$	2	$\mathrm{SE}_1 + \mathrm{SE}_2$	$\mathrm{Lin}_1 + \mathrm{SE}_2$	Lin_1
$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{Per}_1 + \mathrm{SE}_3$	3	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{Per}_1 + \mathrm{SE}_3$	$\mathrm{SE}_2 \times \mathrm{Per}_1 + \mathrm{SE}_3$	-
$\mathrm{SE}_1 \times \mathrm{SE}_2$	4	$\mathrm{SE}_1 \times \mathrm{SE}_2$	$\mathrm{Lin}_1 \times \mathrm{SE}_2$	Lin_2
$\mathrm{SE}_1 \times \mathrm{SE}_2 + \mathrm{SE}_2 \times \mathrm{SE}_3$	4	$\mathrm{SE}_1 \times \mathrm{SE}_2 + \mathrm{SE}_2 \times \mathrm{SE}_3$	$\mathrm{SE}_1 + \mathrm{SE}_2 \times \mathrm{SE}_3$	SE_1
$(\mathrm{SE}_1 + \mathrm{SE}_2) \times (\mathrm{SE}_3 + \mathrm{SE}_4)$	4	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \ldots$ $(\mathrm{SE}_3 \times \mathrm{Lin}_3 \times \mathrm{Lin}_1 + \mathrm{SE}_4)$	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \ldots$ $\mathrm{SE}_3 \times \mathrm{SE}_4$	-

Table 4.1: Kernels chosen by our method on synthetic data generated using known kernel structures. D denotes the dimension of the functions being modelled. SNR indicates the signal-to-noise ratio. Dashes - indicate no structure.

We validated our method's ability to recover known structure on a set of synthetic datasets. For several composite kernel expressions, we constructed synthetic data by first sampling 300 points uniformly at random, then sampling function values at those points from a GP distribution. We then added i.i.d. Gaussian noise to the functions, at various signal-to-noise ratios (SNR).

Table 4.1 lists the true kernels we used to generate the data. Subscripts indicate which dimension each kernel was applied to. Subsequent columns show the dimensionality D of the input space, and the kernels chosen by our search for different SNRs. Dashes - indicate that no kernel had a higher marginal likelihood than modelling the data as i.i.d. Gaussian noise.

For the highest SNR, the method finds all relevant structure in all but one test. The reported additional linear structure is explainable by the fact that functions sampled from SE kernels with long length scales occasionally have near-linear trends. As the noise increases, our method generally backs off to simpler structures.

4.7 Quantitative evaluation

In addition to the qualitative evaluation above, we investigated quantitatively how our method performs on both extrapolation and interpolation tasks.

4.7.1 Extrapolation

We compared the extrapolation capabilities of our model against standard baselines[5]. Dividing the airline dataset into contiguous training and test sets, we computed the predictive mean-squared-error (MSE) of each method. We varied the size of the training set from the first 10% to the first 90% of the data.

Figure 4.7 shows the learning curves of linear regression, a variety of fixed kernel family GP models, and our method. GP models with only SE and Per kernels did not capture the long-term trends, since the best parameter values in terms of GP marginal likelihood only capture short term correlations. Linear regression approximately captured the long-term trend, but quickly plateaued in predictive performance. The more richly structured GP models (SE + Per and SE × Per) eventually captured more structure and performed better, but the full structures discovered by our search outperformed the other approaches in terms of predictive performance for all data amounts. We perform a more extensive version of this experiment in chapter 5.

4.7.2 High-dimensional prediction

To evaluate the predictive accuracy of our method in a high-dimensional setting, we extended the comparison of Duvenaud et al. (2011) to include our method. We performed 10 fold cross validation on 5 datasets[6] comparing 5 methods in terms of mean squared error and predictive likelihood. Our kernel search procedure was run up to depth 10, using the SE and RQ base kernel families[7].

The comparison included three methods with fixed kernel families: Additive GPs, Generalized Additive Models (GAM), and a GP with a standard SE kernel using Automatic Relevance Determination (GP SE-ARD). Also included was the related kernel-search method of Hierarchical Kernel Learning (HKL).

Results are presented in tables 4.2 and 4.3. Our method outperformed the next-best method in each test, although not substantially.

All GP kernel parameter estimation was performed by automated calls to the GPML toolbox[8]; Python code to perform all experiments is available on github[9].

[5] In one dimension, the predictive means of all baseline methods in table 4.2 are identical to that of a GP with an SE kernel.

[6] The data sets had dimensionalities ranging from 4 to 13, and the number of data points ranged from 150 to 450.

[7] We did not believe a priori to find any periodicity or perfect linearity in these data sets.

[8] Available at `www.gaussianprocess.org/gpml/code/`

[9] `github.com/jamesrobertlloyd/gp-structure-search`

Method	bach	concrete	puma	servo	housing
Linear Regression	1.031	0.404	0.641	0.523	0.289
GP GAM	1.259	0.149	0.598	0.281	0.161
HKL	**0.199**	0.147	0.346	0.199	0.151
GP Squared-exp	**0.045**	0.157	**0.317**	**0.126**	**0.092**
GP Additive	**0.045**	**0.089**	**0.316**	**0.110**	0.102
Kernel search	**0.044**	**0.087**	**0.315**	**0.102**	**0.082**

Table 4.2: Comparison of multidimensional regression performance in terms of cross validated mean squared error. Bold results are not significantly different from the best-performing method in each experiment, in a paired t-test with a p-value of 5%.

Method	bach	concrete	puma	servo	housing
Linear Regression	2.430	1.403	1.881	1.678	1.052
GP GAM	1.708	0.467	1.195	0.800	0.457
GP Squared-exp	**−0.131**	0.398	**0.843**	0.429	0.207
GP Additive	**−0.131**	**0.114**	**0.841**	**0.309**	0.194
Kernel search	**−0.141**	**0.065**	**0.840**	**0.265**	**0.059**

Table 4.3: Comparison of multidimensional regression performance in terms of cross validated negative log likelihood. Bold results are not significantly different from the best-performing method in each experiment, in a paired t-test with a p-value of 5%.

4.8 Conclusion and discussion

Towards the goal of automating the choice of kernel family, we introduced a space of composite kernels defined compositionally as sums and products of a small number of base kernels. The set of models in this space includes many standard regression models. We proposed a search procedure for this space of kernels which parallels human model building strategies.

We found that the learned structures are often capable of accurate extrapolation in complex time-series datasets, and are competitive with widely used kernel classes and kernel combination methods on a variety of prediction tasks. The learned kernels often yield decompositions of a signal into diverse and interpretable components, enabling visual model criticism of complex nonparametric models. We believe that a data-driven approach to choosing kernel structures automatically can help make nonparametric regression and classification methods accessible to non-experts.

However, there are a number of improvements that could be made to the procedure

described in this chapter. Some of those improvements are implemented in the work described in chapter 5 but we now discuss two other shortcomings and discuss potential future work to overcome them.

4.8.1 Estimating marginal likelihoods

In section 4.3 it was stated that we used the Bayesian information criterion (BIC) (Schwarz, 1978) to approximate the model evidence of a kernel family. The model evidence is a central quantity in this work since it allows us to balance model complexity and model fit in order to find a parsimonious explanation of a data set. However, the technical conditions required for a completely sound usage of BIC are not met by the Gaussian process models considered in this chapter. We now discuss more principled alternatives and the difficulties that remain before they can be applied.

In an exact and fully Bayesian approach we would put priors over the kernel parameters and compute the model evidence of the kernel families with all the parameters integrated out. Since the relevant equations of Bayes' rule are analytically intractable in this case we turn to approximate inference techniques. The majority of approximate inference methods taking a fully Bayesian approach to Gaussian process regression have focused on producing approximate samples from the posterior distribution (e.g. Lloyd et al., 2012; Murray and Adams, 2010; Snoek et al., 2012) but have ignored the estimation of the model evidence. Variational Bayes and expectation propagation methods naturally produce an estimate or lower bound of model evidence and have been applied to Gaussian processes (e.g. Hensman et al., 2013; Rasmussen and Williams, 2006). However, to the best of my knowledge, all existing techniques condition on kernel parameters which are then optimised which means we are still faced with the problem of appropriately penalising our optimisation of a growing number of kernel parameters. While there is a growing literature on generic and automatic methods for approximating model evidences (e.g. Feroz et al., 2013; Gerrish and Blei; Skilling, 2006) I am unaware of successful applications of these techniques to Gaussian processes with unknown kernel parameters.

A simple method that we attempted to utilise was the Laplace approximation to the model evidence (e.g. Bishop, 2006). This method approximates the log-likelihood function as a quadratic with centre at a mode of the likelihood and curvature at the mode matched to that of the true log-likelihood. To be able to calculate this approximation one must be sufficiently close to a mode of the likelihood in order to be able to produce a numerical estimate of the curvature that is strictly negative definite (i.e. defining an

appropriate turning point of the likelihood).

We found that the conjugate gradients solver used within the kernel search algorithm could become confused by numerical noise and would terminate before it reached a mode making the Laplace approximation impossible. The situation of not quite being at the mode however is not just a numerical annoyance, it is becoming increasingly common. As data sets become increasingly large and computation and time constraints of algorithms become increasingly important it will often be irrational to run optimisers to completion since it may be more beneficial to spend computation on investigating other models (e.g. Swersky et al., 2014).

A potential way forward is to imagine a more general version of the Laplace approximation. The Laplace approximation attempts to approximate the log-likelihood with a quadratic; this can be achieved without exactly having found the mode simply by performing regression using a quadratic regression function. This could be viewed as a simple version of Bayesian quadrature (e.g. Ghahramani and Rasmussen, 2002; O'Hagan, 1991) with potentially attractive computation time. In fact, it has recently been shown by Hennig and Kiefel (2012) that aspects of certain quasi-Newton optimisation methods can be interpreted as approximate inference within a probabilistic model. In particular, the running estimate of curvature produced by the BFGS algorithm could be used to produce a quadratic approximation to the optimisation surface. It would be profitable to study the effectiveness of this curvature estimate as a tool for model evidence approximation in comparison to the revised methods proposed by Hennig and Kiefel (2012).

4.8.2 Model selection rather than model averaging

In this chapter we have demonstrated that selecting the model with the highest model evidence encountered along the search was sufficient to achieve good empirical performance on the moderate dimensional datasets considered. However, much like BIC guided linear model selection will fail on high dimensional problems (e.g. Hastie et al., 2009), so will the kernel search as currently described. Staying within the Bayesian paradigm it is conceptually simple to alleviate this problem by instead computing a Bayes model average (e.g. Hoeting et al., 1999) over kernel families. The natural optimisation objective would then change from optimising the model evidence of a single kernel family to that of the sum of model evidences of all kernel families visited by the search, appropriately weighted by a prior over kernel families. Setting the prior over kernel families could be a very interesting exercise if attempting to adapt the prior to match the patterns found in

'typical' data sets. Optimising the weighted sum of model evidences would also require a new search algorithm to be effective; we are currently investigating if the freeze-thaw algorithm of Swersky et al. (2014) can be effectively applied to this domain.

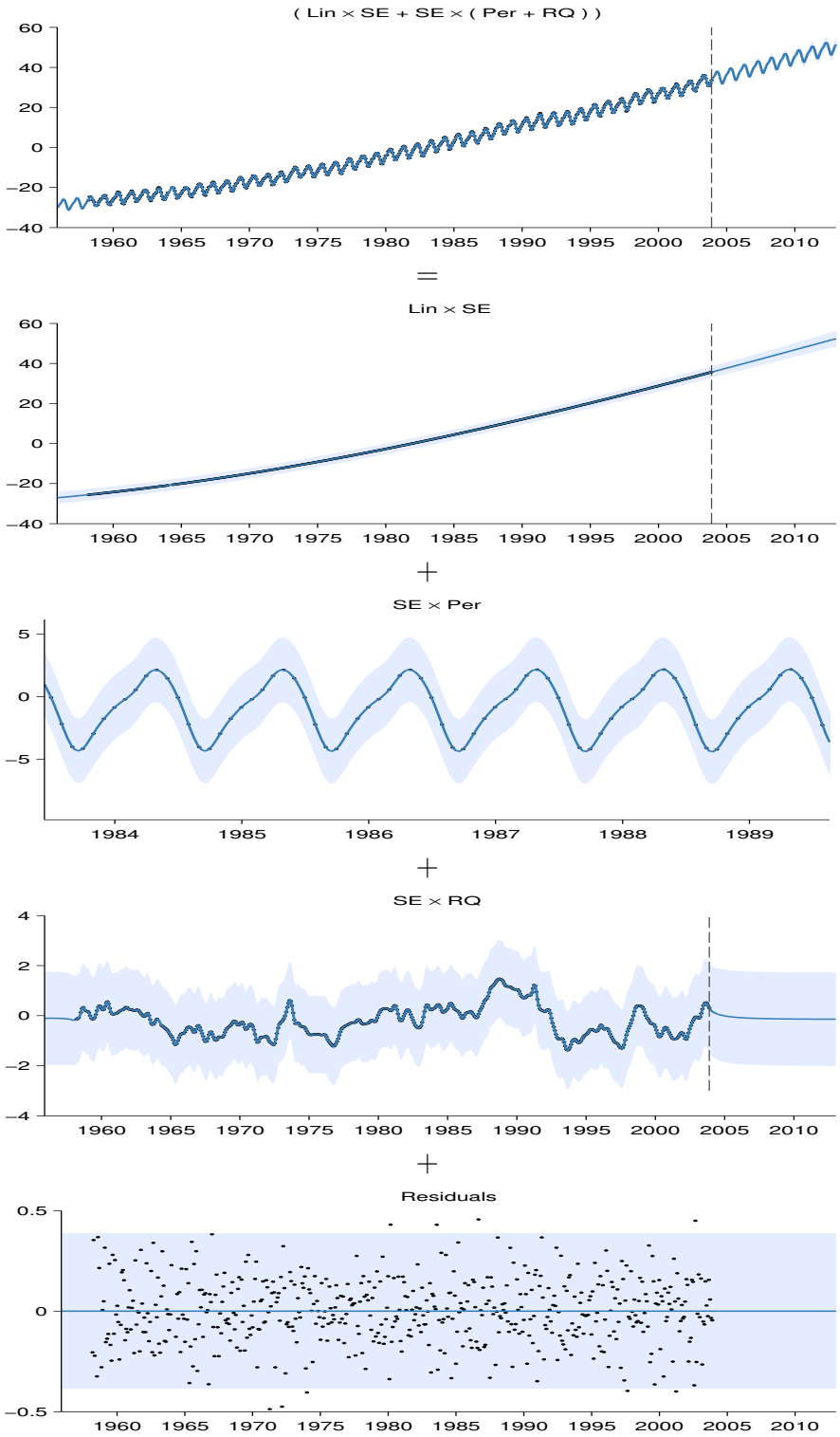


Figure 4.4: First row: The posterior on the Mauna Loa dataset, after a search of depth 10. Subsequent rows show the automatic decomposition of the time series. The decompositions shows long-term, yearly periodic, medium-term anomaly components, and residuals, respectively. In the third row, the scale has been changed in order to clearly show the yearly periodic structure.

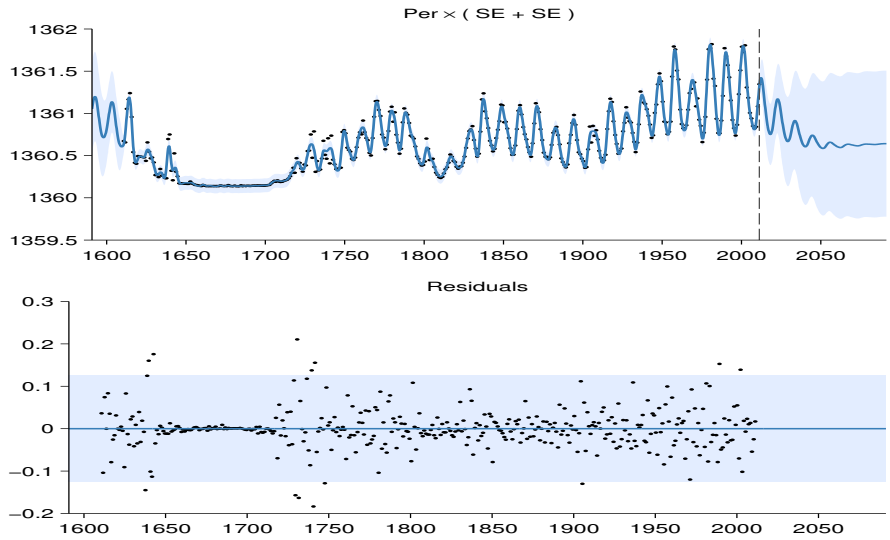


Figure 4.5: Full posterior and residuals on the solar irradiance dataset.

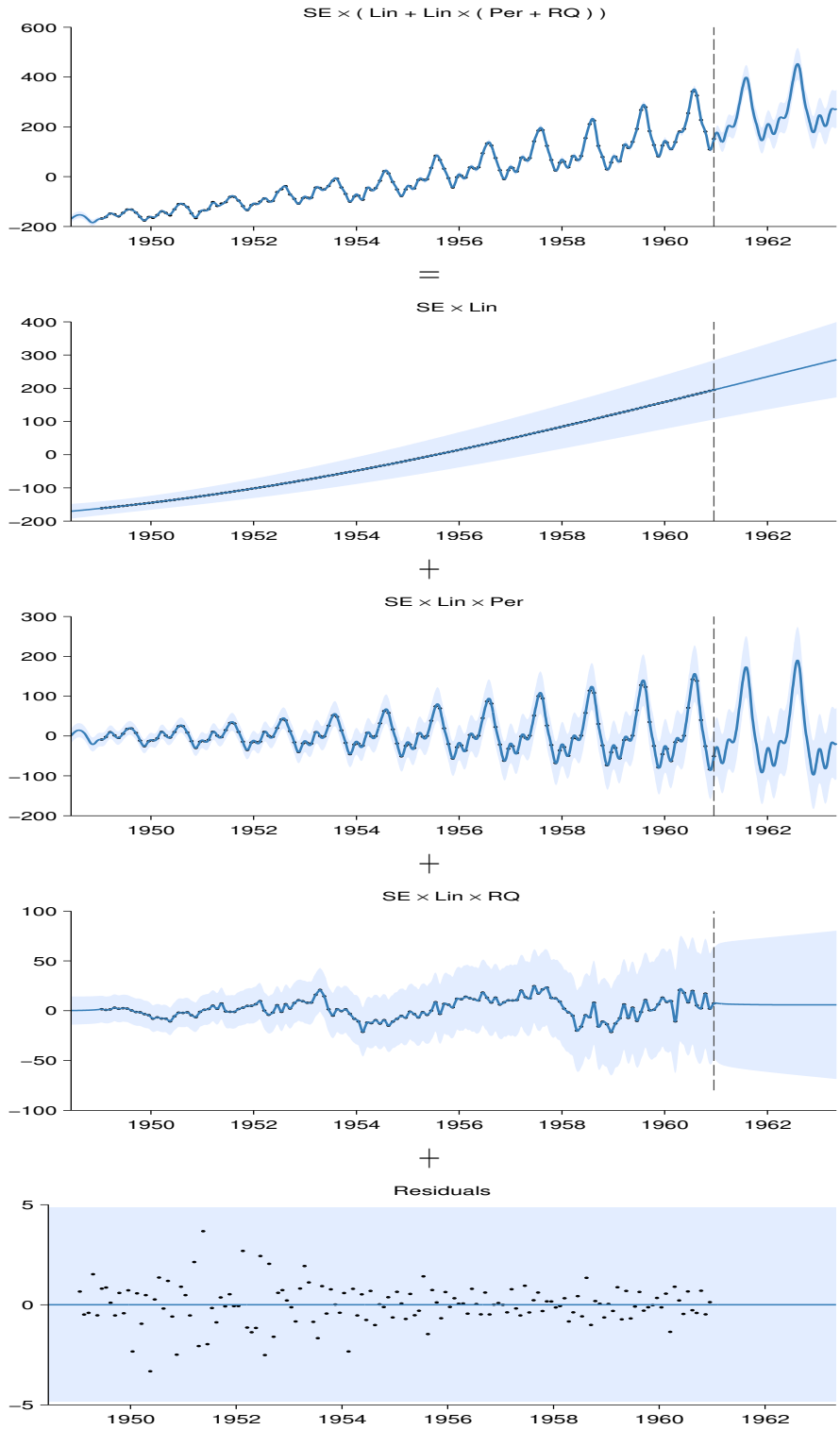


Figure 4.6: First row: The airline dataset and posterior after a search of depth 10. Subsequent rows: Additive decomposition of posterior into long-term smooth trend, yearly variation, and short-term deviations. Due to the linear kernel, the marginal variance grows over time, making this a heteroscedastic model.

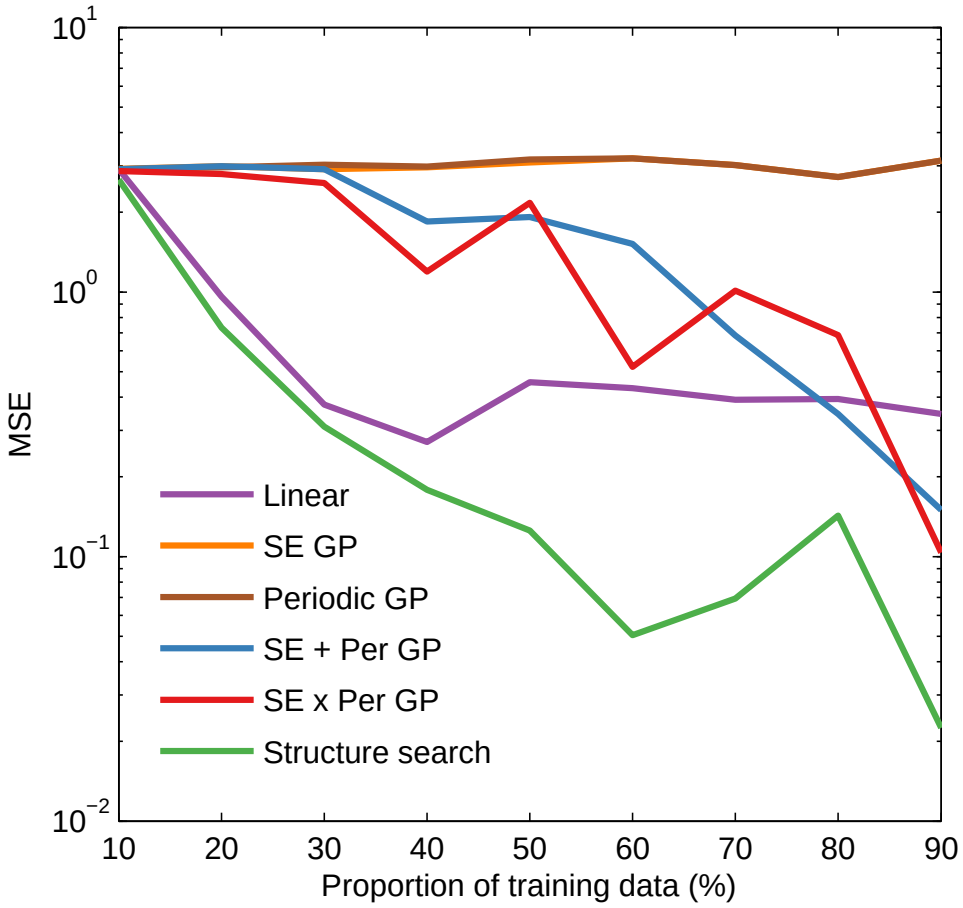


Figure 4.7: Extrapolation performance on the airline dataset. We plot test-set MSE as a function of the fraction of the dataset used for training.

Chapter 5

Natural-language description of Gaussian process models

In chapter 4 we demonstrated that the models produced by the Gaussian process kernel search could yield interpretable decompositions of regression models when applied to time series. We now justify this claim of interpretability by demonstrating that the models produced by the kernel search can be automatically interpreted. In particular we show that the grammar generated language of kernel families neatly maps onto natural-language sentences describing the properties of the functions these kernels encode for.

This chapter is based upon joint work with David Duvenaud, Roger Grosse, Joshua Tenenbaum and Zoubin Ghahramani (Lloyd et al., 2014). I performed all experiments; all were involved with the writing.

5.1 Introduction

The kernel search procedure introduced in chapter 4 can be viewed as an approximately Bayesian agent performing an exploratory data analysis. Given no information about a data set other than the data itself the kernel search explores a large space of potential statistical models, reporting those which appear to be the most parsimonious description of the data. Viewed either as an autonomous agent or as a tool, this type of system has the potential to significantly aid those without expert machine learning or statistical knowledge to perform exploratory analyses of novel data sets.

However, while graphical descriptions can convey some of the patterns encoded by a statistical model, much of the detail of the properties of the statistical model are expressed by the forms of the kernels. However, there are very few people even who can

interpret the meaning of Gaussian processes with novel kernel functions. It is therefore natural to ask to what extent any kernel produced by the kernel search algorithm can be explained in relatively simple terms.

In this chapter we demonstrate that the compositional structure of the kernels produced by the search algorithm neatly map onto compositionally generated natural-language descriptions of the types of functions they encode for. In attempting to produce an algorithm that can describe arbitrary kernel expressions we discover that some kernel expressions are much less easy to intuitively describe than others. In response we revise the language of kernel functions which in turn leads to more interpretable models being produced by the search.

Combining the kernel search with automatic descriptions, predictions and model criticism (see chapter 7) we introduce a system for modelling time-series data which we call the Automatic Bayesian Covariance Discovery (ABCD) system. A flow chart for the operation of the system is shown in figure 5.1.

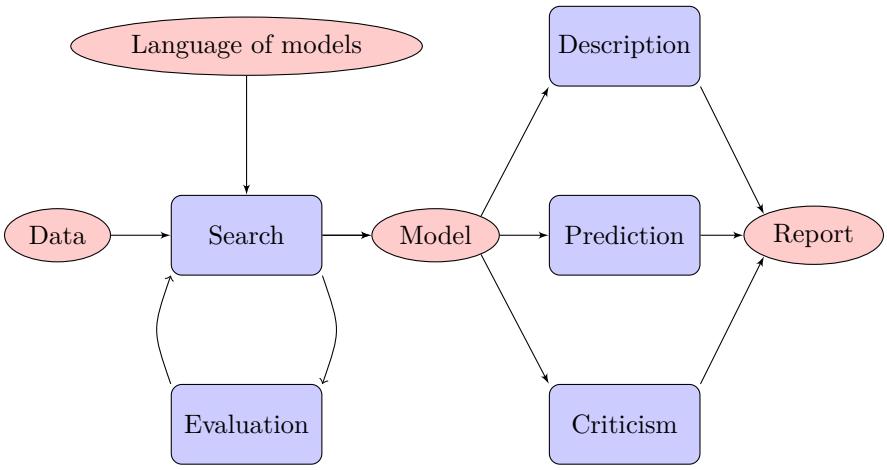


Figure 5.1: Flow chart of the ABCD system. An open-ended language of Gaussian process models is defined by a compositional grammar. The space is searched greedily, using marginal likelihood and the Bayesian Information Criterion (BIC) to evaluate models. The compositional structure of the language allows us to develop a method for automatically translating components of the model into natural-language descriptions of patterns in the data. These descriptions are then combined with predictions from the model and model criticism to form a report describing the data.

5.2 Improvements and additions to the language of kernels

5.2.1 Improvements to interpretability

We now demonstrate that some of the kernel expressions that can be reached by the search procedure described in chapter 5 do not lend themselves to automatic description in natural language. Furthermore, at their worst they can produce posterior distributions over functions that are highly uninterpretable or unusual. We find that removing or modifying these kernels makes automatic description possible. Furthermore, removing and modifying these kernels has the potentially unexpected benefit of improving the interpretability of the resulting statistical models.

Removal of the rational quadratic kernel The squared exponential (SE) kernel encodes for probability distributions over smooth functions; the rational quadratic (RQ) kernel also encodes for smooth functions and can be expressed as a scale mixture of infinitely many SE kernels. Kernels such as SE×RQ and RQ×RQ also encode for yet more variations on smooth functions. It seems unlikely that there can be simple descriptions of all of these different types of smoothness (there are only so many synonyms) without resorting to equations or graphs of the kernels; neither of these are easy to interpret, even by experts.

This is one of the reasons why we remove the rational quadratic kernel from our set of base kernels with which we form the language of kernel families. As we will see in section 5.4 this will allow us to only have to describe one type of smoothness making the task of automatic description into natural language much simpler. In addition to this, we avoid the occasionally pathological behaviour of the rational quadratic to capture multiple scales of variation in a single function. The left of figure 5.2 shows the posterior of a component involving a rational quadratic kernel produced by the procedure of chapter 4 on the Mauna Loa data set. This component has captured both a medium term trend and short term variation. This is both visually unappealing and difficult to describe simply. In contrast, the right of figure 5.2 shows two of the components produced by ABCD on the same data set which clearly separate the medium term trend and short term deviations.

An even more extreme pathological behaviour can be seen in figure 5.3 where a kernel of the form $\mathrm{RQ} \times \mathrm{Per}$ has explained a data set which ABCD would more naturally de-

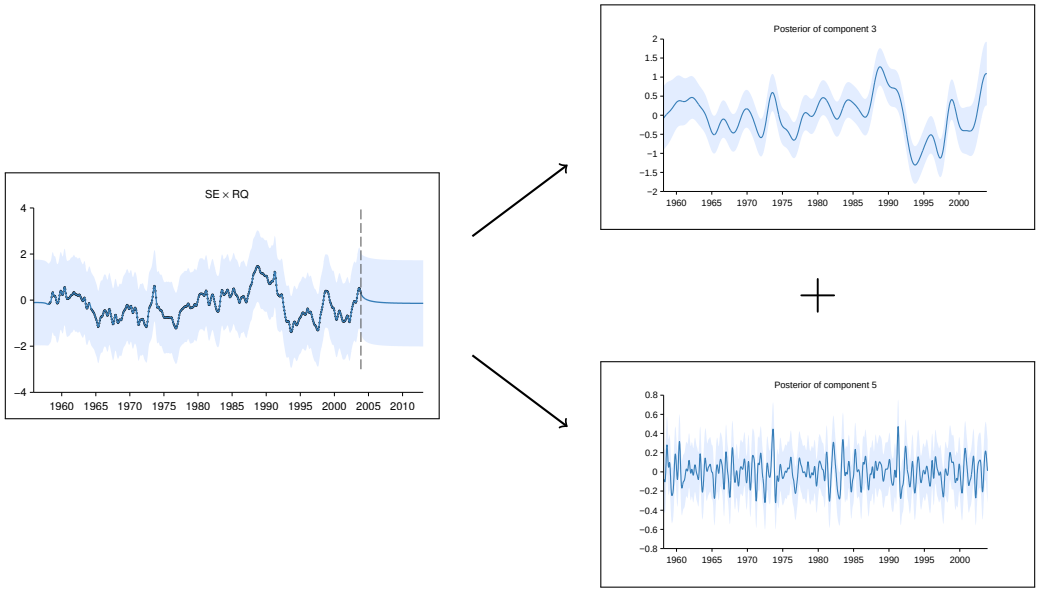


Figure 5.2: Left: Posterior of rational quadratic component of model for Mauna Loa data from chapter 4. Right: Posterior of two components found by ABCD - the different scales of variation have been separated.

scribed as a sum of a periodic component and uncorrelated noise. The rational quadratic has introduced multiple lengthscales into the function allowing it to make a reasonable extrapolation of the periodicity in the data but also fit every data point perfectly. For these reasons we exclude the rational quadratic kernel from our language.

At this point one may reasonably ask what the definition of 'more natural' is. I am using these phrases in the sense of attempting to produce a language of statistical models that match my own subjective biases. By finding these pathological examples we are finding consequences of the grammar of kernels that do not match our subjective beliefs about the types of functions we expect to see. While subjectivity is often less desirable than some form of objectivity when beginning a scientific investigation, it is probably better than biasing an analysis towards mathematically convenient forms that go against our subjective biases unless they can be proven to have good properties. We discuss these philosophical points at greater length in section 5.8.

Subtraction of unnecessary constants The typical definition of the periodic (Per) kernel (e.g. Rasmussen and Williams, 2006) used by Duvenaud et al. (2013), Kronberger and Kommenda (2013) and in chapter 4[1] is always greater than zero. This is not nec-

[1]and almost every other application of Gaussian processes to periodic data

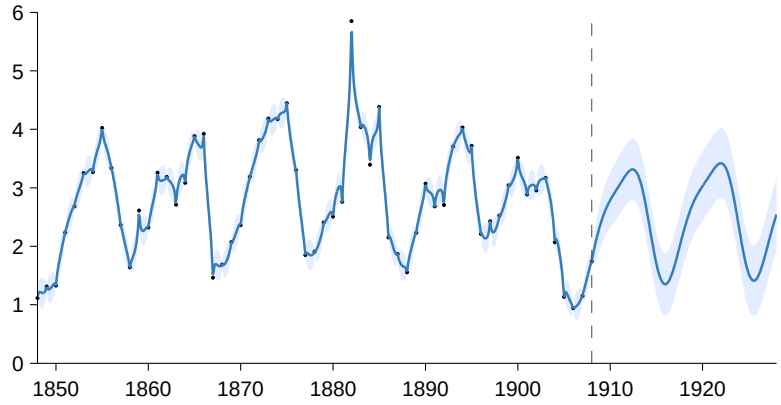


Figure 5.3: Fit of a kernel of the form $\mathrm{RQ} \times \mathrm{Per}$ to mink fur sales data. While this kernel produces good extrapolations of the average periodicity, it simulataneously 'joins the dots', fitting all data points almost exactly. It would be difficult to parsimoniously describe the properties of this function in words.

essary for the kernel to be positive semidefinite; we can subtract a constant from this kernel. Similarly, the linear kernel used by Duvenaud et al. (2013) and in chapter 4 contains a constant term that can be subtracted.

Without these constants subtracted we observe two main problems. First, descriptions of products of kernels would become convoluted e.g. $(\mathrm{Per} + \mathrm{C}) \times (\mathrm{Lin} + \mathrm{C}) = \mathrm{C} + \mathrm{Per} + \mathrm{Lin} + \mathrm{Per} \times \mathrm{Lin}$ is a sum of four qualitatively different functions (see section 5.4 for justification that each of these kernels encodes for a distinct type of function). Second, the constant functions can result in anti-correlation between components in the posterior, resulting in unnecessarily inflated credible intervals for each component which is shown in figure 5.4.

5.2.2 Properties of the new periodic kernel

The newly defined zero-mean periodic kernel is given by

$$\mathrm{ZMPer}(x, x') = \sigma^2 \frac{\exp\left(\frac{\cos\frac{2\pi(x-x')}{p}}{\ell^2}\right) - I_0\left(\frac{1}{\ell^2}\right)}{\exp\left(\frac{1}{\ell^2}\right) - I_0\left(\frac{1}{\ell^2}\right)} \tag{5.2.1}$$

where I_0 is the modified Bessel function of the first kind of order zero. It is simply a carefully chosen linear transformation of the more common periodic kernel defined in

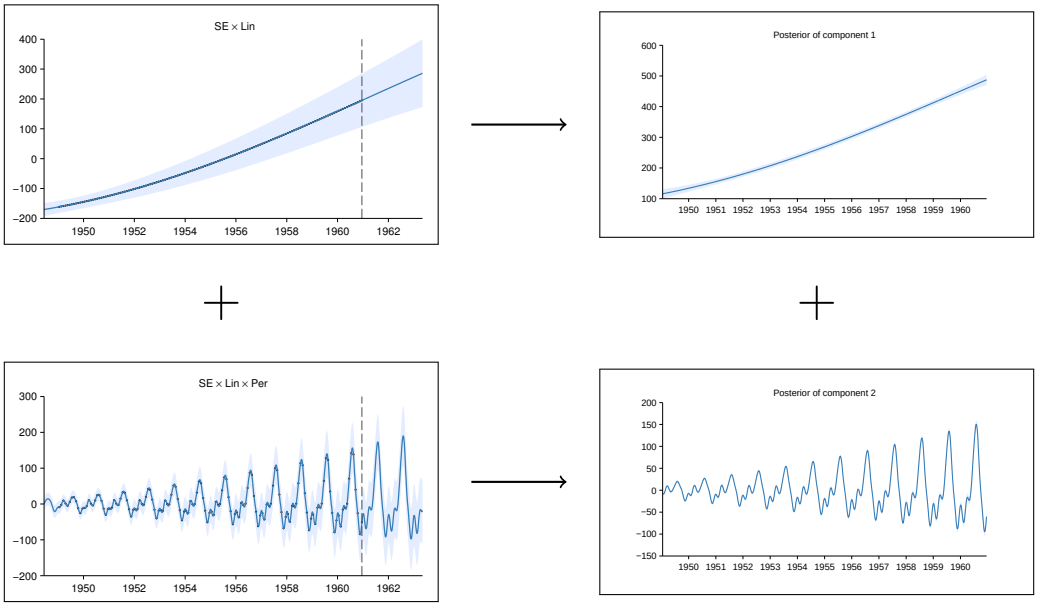


Figure 5.4: Left: Posterior of first two components for the airline passenger data from Duvenaud et al. (2013) and chapter 4. Right: Posterior of first two components found by ABCD - removing the constants from Lin and Per has removed the inflated credible intervals due to anti-correlation in the posterior.

equation 4.2.2. A simple application of l'Hôpital's rule shows that

$$\mathrm{ZMPer}(x, x') \to \sigma^2 \cos\left(\frac{2\pi(x - x')}{p}\right) \quad \text{as} \quad \ell \to \infty. \tag{5.2.2}$$

This limiting form is written as the cosine kernel (cos). It was recently demonstrated (Wilson and Adams, 2013) that any stationary kernel can be arbitrarily well approximated by kernels with expressions of the form

$$\sum \mathrm{SE} \times \cos \tag{5.2.3}$$

by appealing to Bochner's theorem (Bochner et al., 1959). By using this new periodic kernel our language of kernels also attains this completeness property. Note that the infinite lengthscale (ℓ) limit of the more common definition of the periodic kernel is a constant.

This version of the periodic kernel is now implemented in the GPML toolbox[2] and is available at `gaussianprocess.org/gpml/code/matlab/cov/covPeriodicNoDC.m`

[2] `http://www.gaussianprocess.org/gpml/code/matlab/doc/`

5.2.3 Addition of the changepoint and changewindow operators

In addition to these changes to the base kernels we also introduce changepoints into the language of kernels since they are a common phenomenon in time series (e.g. figure 5.10). We construct changepoints by assuming that the function we are attempting to model transitions between two different functions with the transition controlled by a sigmoidal function σ:

$$f(x) = \sigma(x)f_1(x) + (1 - \sigma(x))f_2(x). \tag{5.2.4}$$

If we assume that $f_1 \sim \mathrm{GP}(0, k_1)$ and $f_2 \sim \mathrm{GP}(0, k_2)$ independently then $f \sim \mathrm{GP}(0, k)$ where

$$k(x, x') = \sigma(x)\sigma(x')k_1(x, x') + (1 - \sigma(x))(1 - \sigma(x'))k_2(x, x'). \tag{5.2.5}$$

We therefore define the changepoint operator as

$$\mathrm{CP}(k_1, k_2) = k_1 \times \boldsymbol{\sigma} + k_2 \times \bar{\boldsymbol{\sigma}} \tag{5.2.6}$$

where $\boldsymbol{\sigma} = \sigma(x)\sigma(x')$ and $\bar{\boldsymbol{\sigma}} = (1 - \sigma(x))(1 - \sigma(x'))$. Not written are parameters of the sigmoid σ that control its location and steepness.

We define changewindows $\mathrm{CW}(\cdot, \cdot)$ similarly by replacing $\sigma(x)$ with a product of two sigmoids to create a 'hat' function. Examples of draws from Gaussian processes with kernels constructed using the changepoint operator are shown in figure 5.5.

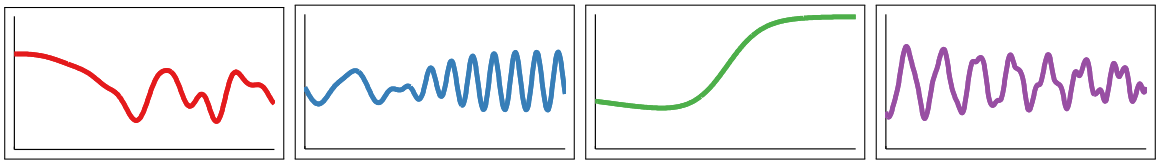


Figure 5.5: Examples of random samples from Gaussian processes with kernels constructed using the changepoint operator.

5.2.4 Including heteroscedasticity

In the language of kernels introduced in chapter 4 homoscedastic Gaussian noise was assumed for all models. Gaussian noise is also a Gaussian process so we can introduce

the noise model into the modelling language by introducing the white noise kernel

$$\mathrm{WN}(x, x') = \sigma^2 \delta_{x,x'} \tag{5.2.7}$$

where $\delta_{x,x'}$ is the Kronecker delta function. We use this as a base kernel and replace the model likelihood with a delta function.

Introducing the white noise kernel as a base kernel allows for the modelling language to express some forms of heteroscedasticity. In particular, a WN×Lin kernel encodes for uncorrelated Gaussian noise with linearly varying standard deviation. The changepoint and changewindow operators also allow for heteroscedasticity when the noise model is included in the kernel.

5.3 An updated language of and search over regression models

Since we have an updated set of base kernels and operators we review them.

5.3.1 Base kernels

For scalar-valued inputs, the white noise (WN), constant (C), linear (Lin), squared exponential (SE), and zero mean periodic kernels (ZMPer) are defined as follows:

$$\mathrm{WN}(x, x') = \sigma^2 \delta_{x,x'} \tag{5.3.1}$$

$$\mathrm{C}(x, x') = \sigma^2 \tag{5.3.2}$$

$$\mathrm{Lin}(x, x') = \sigma^2 (x - \ell)(x' - \ell) \tag{5.3.3}$$

$$\mathrm{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) \tag{5.3.4}$$

$$\mathrm{ZMPer}(x, x') = \sigma^2 \frac{\exp\left(\frac{\cos\frac{2\pi(x-x')}{p}}{\ell^2}\right) - I_0\left(\frac{1}{\ell^2}\right)}{\exp\left(\frac{1}{\ell^2}\right) - I_0\left(\frac{1}{\ell^2}\right)} \tag{5.3.5}$$

where $\delta_{x,x'}$ is the Kronecker delta function, I_0 is the modified Bessel function of the first kind of order zero and other symbols are parameters of the kernel functions.

5.3.2 A larger language of regression models

Table 5.1 lists common regression models that can be expressed by our language. In this chapter we restrict to one dimensional functions (in particular, time series); the revised modelling language naturally extends into multiple dimensions in the manner described in chapter 4.

Regression model	Kernel
GP smoothing	$\mathrm{SE} + \mathrm{WN}$
Linear regression	$\mathrm{C} + \mathrm{Lin} + \mathrm{WN}$
Multiple kernel learning	$\sum \mathrm{SE} + \mathrm{WN}$
Trend, cyclical, irregular	$\sum \mathrm{SE} + \sum \mathrm{ZMPer} + \mathrm{WN}$
Fourier decomposition*	$\mathrm{C} + \sum \cos + \mathrm{WN}$
Sparse spectrum GPs*	$\sum \cos + \mathrm{WN}$
Spectral mixture*	$\sum \mathrm{SE} \times \cos + \mathrm{WN}$
Changepoints*	e.g. $\mathrm{CP}(\mathrm{SE}, \mathrm{SE}) + \mathrm{WN}$
Heteroscedasticity*	e.g. $\mathrm{SE} + \mathrm{Lin} \times \mathrm{WN}$

Table 5.1: Common regression models expressible in our language. cos is a special case of ZMPer. * indicates a model that could not be expressed by the language defined in chapter 4.

5.3.3 Search operators

The search operators introduced in chapter 4 are as follows:

$$\mathcal{S} \rightarrow \mathcal{S} + \mathcal{B} \tag{5.3.6}$$

$$\mathcal{S} \rightarrow \mathcal{S} \times \mathcal{B} \tag{5.3.7}$$

$$\mathcal{B} \rightarrow \mathcal{B}' \tag{5.3.8}$$

where $\mathcal{S}$ represents any kernel subexpression and $\mathcal{B}$ is any base kernel i.e. the search operators represent addition, multiplication and replacement.

To accommodate changepoint/window operators we introduce the following addi-

tional operators

$$\mathcal{S} \rightarrow \mathrm{CP}(\mathcal{S}, \mathcal{S}) \tag{5.3.9}$$
$$\mathcal{S} \rightarrow \mathrm{CW}(\mathcal{S}, \mathcal{S}) \tag{5.3.10}$$
$$\mathcal{S} \rightarrow \mathrm{CW}(\mathcal{S}, \mathrm{C}) \tag{5.3.11}$$
$$\mathcal{S} \rightarrow \mathrm{CW}(\mathrm{C}, \mathcal{S}) \tag{5.3.12}$$

where C is the constant kernel. The last two operators result in a kernel only applying outside or within a certain region.

Based on experience with typical paths followed by the search algorithm we introduced the following operators

$$\mathcal{S} \rightarrow \mathcal{S} \times (\mathcal{B} + \mathrm{C}) \tag{5.3.13}$$
$$\mathcal{S} \rightarrow \mathcal{B} \tag{5.3.14}$$
$$\mathcal{S} + \mathcal{S}' \rightarrow \mathcal{S} \tag{5.3.15}$$
$$\mathcal{S} \times \mathcal{S}' \rightarrow \mathcal{S} \tag{5.3.16}$$

where $\mathcal{S}'$ represents any other kernel expression. Their introduction is currently not rigorously justified.

5.4 Automatic description of regression models

Overview In this section, we describe how ABCD generates natural-language descriptions of the models found by the search procedure. There are two main features of our language of GP models that allow description to be performed automatically.

First, the sometimes complicated kernel expressions can be simplified into a sum of products. A sum of kernels corresponds to a sum of functions so each product can be described separately. Second, each kernel in a product modifies the resulting model in a consistent way. Therefore, we can choose one kernel to be described as a noun, with all others described using adjectives or modifiers.

Sum of products normal form We convert each kernel expression into a standard, simplified form. We do this by first distributing all products of sums into a sum of products. Next, we apply several simplifications to the kernel expression: The product of two SE kernels is another SE with different parameters. Multiplying WN by any

stationary kernel (C, WN, SE, or ZMPer) gives another WN kernel. Multiplying any kernel by C only changes the parameters of the original kernel.

After applying these rules, the kernel can as be written as a sum of terms of the form:

$$K \prod_m \mathrm{Lin}^{(m)} \prod_n \boldsymbol{\sigma}^{(n)},$$

where K, if present, is one of WN, C, SE, $\prod_k \mathrm{ZMPer}^{(k)}$ or $\mathrm{SE} \prod_k \mathrm{ZMPer}^{(k)}$ and $\prod_i k^{(i)}$ denotes a product of kernels, each with different parameters.

Sums of kernels are sums of functions Formally, if $f_1(x) \sim \mathrm{GP}(0, k_1)$ and independently $f_2(x) \sim \mathrm{GP}(0, k_2)$ then $f_1(x) + f_2(x) \sim \mathrm{GP}(0, k_1 + k_2)$. This lets us describe each product of kernels separately just as we plotted separate additive components in chapter 4

Each kernel in a product modifies a model in a consistent way This allows us to describe the contribution of each kernel as a modifier of a noun phrase. These descriptions are summarised in table 5.2 and justified below:

- **Multiplication by SE** removes long range correlations from a model since $\mathrm{SE}(x, x')$ decreases monotonically to 0 as $|x - x'|$ increases. This will convert any global correlation structure into local correlation only.

- **Multiplication by Lin** is equivalent to multiplying the function being modelled by a linear function. If $f(x) \sim \mathrm{GP}(0, k)$, then $xf(x) \sim \mathrm{GP}(0, k \times \mathrm{Lin})$. This causes the standard deviation of the model to vary linearly without affecting the correlation.

- **Multiplication by $\boldsymbol{\sigma}$** is equivalent to multiplying the function being modelled by a sigmoid which means that the function goes to zero before or after some point.

- **Multiplication by ZMPer** modifies the correlation structure in the same way as multiplying the function by an independent periodic function. Formally, if $f_1(x) \sim \mathrm{GP}(0, k_1)$ and $f_2(x) \sim \mathrm{GP}(0, k_2)$ then

$$\mathrm{Cov}\left[f_1(x)f_2(x), f_1(x')f_2(x')\right] = k_1(x, x')k_2(x, x').$$

We note however that since this multiplication affects only the correlation structure, the description given in table 5.2 is somewhat loose.

Kernel	Postmodifier phrase
SE	whose shape changes smoothly
ZMPer	modulated by a periodic function
Lin	with linearly varying amplitude
$\prod_k \mathrm{Lin}^{(k)}$	with polynomially varying amplitude
$\prod_k \boldsymbol{\sigma}^{(k)}$	which applies until / from [changepoint]

Table 5.2: Postmodifier descriptions of each kernel

Constructing a complete description of a product of kernels We choose one kernel to act as a noun which is then described by the functions it encodes for when unmodified (see table 5.3). Modifiers corresponding to the other kernels in the product are then appended to this description, forming a noun phrase of the form:

$$\text{Determiner} + \text{Premodifiers} + \text{Noun} + \text{Postmodifiers}$$

As an example, a kernel of the form $\mathrm{ZMPer} \times \mathrm{Lin} \times \boldsymbol{\sigma}$ could be described as a

$$\underbrace{\mathrm{ZMPer}}_{\text{periodic function}} \times \underbrace{\mathrm{Lin}}_{\text{with linearly varying amplitude}} \times \underbrace{\boldsymbol{\sigma}}_{\text{which applies until 1700.}}$$

where ZMPer has been selected as the head noun.

Kernel	Noun phrase
WN	uncorrelated noise
C	constant
SE	smooth function
ZMPer	periodic function
Lin	linear function
$\prod_k \mathrm{Lin}^{(k)}$	polynomial

Table 5.3: Noun phrase descriptions of each kernel

Refinements to the descriptions There are a number of ways in which the descriptions of the kernels can be made more interpretable and informative:

- Which kernel is chosen as the head noun can change the interpretability of a description.
- Descriptions can change qualitatively according to kernel parameters e.g. 'a rapidly varying smooth function'.
- Descriptions can include kernel parameters e.g. 'modulated by a periodic function with a period of [period]'.
- Descriptions can include extra information calculated from data e.g. 'a linearly increasing function'.
- Some kernels can be described as premodifiers e.g. 'an approximately periodic function'.

The reports in appendix A and section 5.5 include some of these refinements. For example, the head noun is chosen according to the following ordering:

$$\text{ZMPer} > \text{WN}, \text{SE}, \text{C} > \prod_m \text{Lin}^{(m)} > \prod_n \boldsymbol{\sigma}^{(n)}$$

i.e. ZMPer is always chosen as the head noun when present. The parameters and design choices of these refinements have been chosen by our best judgement, but learning these parameters objectively from expert statisticians would be an interesting area for future study.

Ordering additive components The reports generated by ABCD attempt to present the most interesting or important features of a data set first. As a heuristic, we order components by always adding next the component which most reduces the 10-fold cross-validated mean absolute error.

5.4.1 Worked example

Suppose we start with a kernel of the form

$$\text{SE} \times (\text{WN} \times \text{Lin} + \text{CP}(\text{C}, \text{ZMPer})).$$

This is converted to a sum of products:

$$\text{SE} \times \text{WN} \times \text{Lin} + \text{SE} \times \text{C} \times \boldsymbol{\sigma} + \text{SE} \times \text{ZMPer} \times \bar{\boldsymbol{\sigma}}.$$

which is simplified to

$$\text{WN} \times \text{Lin} + \text{SE} \times \boldsymbol{\sigma} + \text{SE} \times \text{ZMPer} \times \bar{\boldsymbol{\sigma}}.$$

To describe the first component, the head noun description for WN, 'uncorrelated noise', is concatenated with a modifier for Lin, 'with linearly increasing amplitude'. The second component is described as 'A smooth function with a lengthscale of [lengthscale] [units]', corresponding to the SE, 'which applies until [changepoint]', which corresponds to the $\boldsymbol{\sigma}$. Finally, the third component is described as 'An approximately periodic function with a period of [period] [units] which applies from [changepoint]'.

5.5 Example descriptions of time series

We demonstrate the ability of our procedure to discover and describe a variety of patterns on two time series. An example of a full automatically-generated report is provided in appendix A.

5.5.1 Summarizing 400 Years of Solar Activity

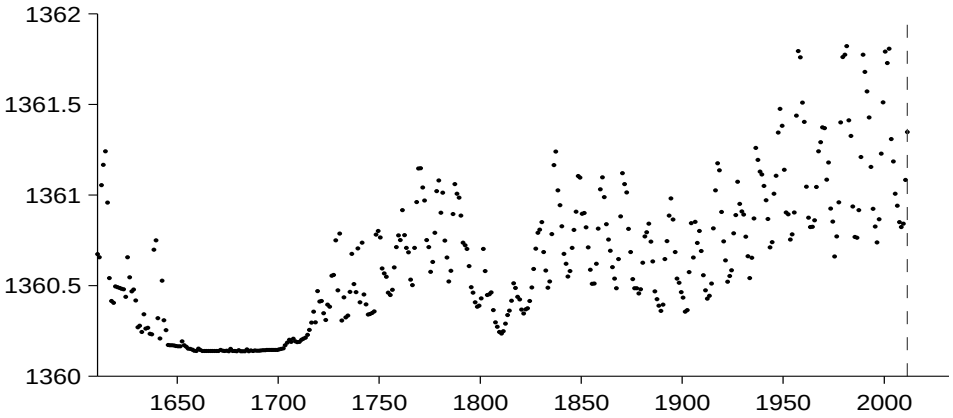


Figure 5.6: Solar irradiance data.

We show excerpts from the report automatically generated on annual solar irradiation data from 1610 to 2011 (figure 5.6). This time series has two pertinent features: a roughly 11-year cycle of solar activity, and a period lasting from 1645 to 1715 with much smaller variance than the rest of the dataset. This flat region is known as the Maunder

minimum, a period in which sunspots were extremely rare (Lean et al., 1995). ABCD clearly identifies these two features, as discussed below.

The structure search algorithm has identified eight additive components in the data. The first 4 additive components explain 92.3% of the variation in the data as shown by the coefficient of determination (R^2) values in table 1. The first 6 additive components explain 99.7% of the variation in the data. After the first 5 components the cross validated mean absolute error (MAE) does not decrease by more than 0.1%. This suggests that subsequent terms are modelling very short term trends, uncorrelated noise or are artefacts of the model or search procedure. Short summaries of the additive components are as follows:

- A constant.
- A constant. This function applies from 1643 until 1716.
- A smooth function. This function applies until 1643 and from 1716 onwards.
- An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards.

Figure 5.7: Automatically generated descriptions of the components discovered by ABCD on the solar irradiance data set. The dataset has been decomposed into diverse structures with simple descriptions.

Figure 5.7 shows the natural-language summaries of the top four components chosen by ABCD. From these short summaries, we can see that our system has identified the Maunder minimum (second component) and 11-year solar cycle (fourth component). These components are visualised in figures 5.8 and 5.10, respectively. The third component corresponds to long-term trends, as visualised in figure 5.9.

5.5.2 Finding heteroscedasticity in air traffic data

Next, we present the analysis generated by our procedure on international airline passenger data (figure 5.11). The model constructed by ABCD has four components: Lin + SE × ZMPer × Lin + SE + WN × Lin, with descriptions given in figure 5.12.

The second component (figure 5.13) is accurately described as approximately (SE) periodic (ZMPer) with linearly increasing amplitude (Lin). By multiplying a white noise kernel by a linear kernel, the model is able to express heteroscedasticity (figure 5.14).

5.5.3 Comparison to equation learning

We now compare the descriptions generated by ABCD to parametric functions produced by an equation learning system. We show equations produced by Eureqa (Nutonian,

This component is constant. This component applies from 1643 until 1716.

This component explains 37.4% of the residual variance; this increases the total variance explained from 0.0% to 37.4%. The addition of this component reduces the cross validated MAE by 31.97% from 0.33 to 0.23.

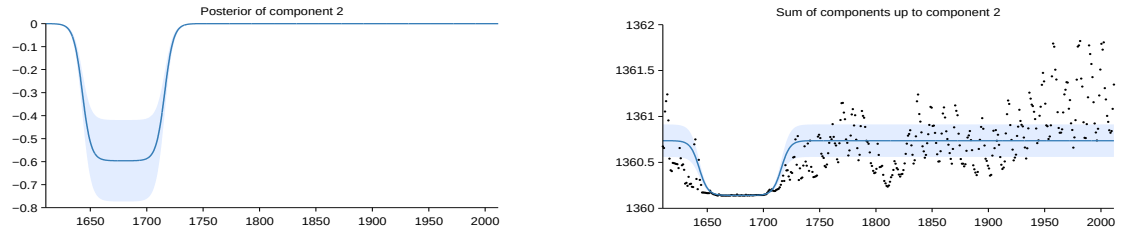


Figure 4: Pointwise posterior of component 2 (left) and the posterior of the cumulative sum of components with data (right)

Figure 5.8: One of the learned components corresponds to the Maunder minimum.

This component is a smooth function with a typical lengthscale of 23.1 years. This component applies until 1643 and from 1716 onwards.

This component explains 56.6% of the residual variance; this increases the total variance explained from 37.4% to 72.8%. The addition of this component reduces the cross validated MAE by 21.08% from 0.23 to 0.18.

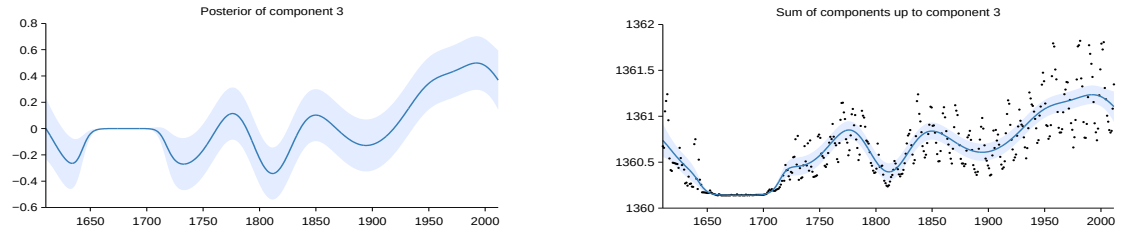


Figure 6: Pointwise posterior of component 3 (left) and the posterior of the cumulative sum of components with data (right)

Figure 5.9: Characterising the medium-term smoothness of solar activity levels. By allowing other components to explain the periodicity, noise, and the Maunder minimum, ABCD can isolate the part of the signal best explained by a slowly-varying trend.

2011) for the data sets shown above, using the default mean absolute error performance metric.

This component is approximately periodic with a period of 10.8 years. Across periods the shape of this function varies smoothly with a typical lengthscale of 36.9 years. The shape of this function within each period is very smooth and resembles a sinusoid. This component applies until 1643 and from 1716 onwards.

This component explains 71.5% of the residual variance; this increases the total variance explained from 72.8% to 92.3%. The addition of this component reduces the cross validated MAE by 16.82% from 0.18 to 0.15.

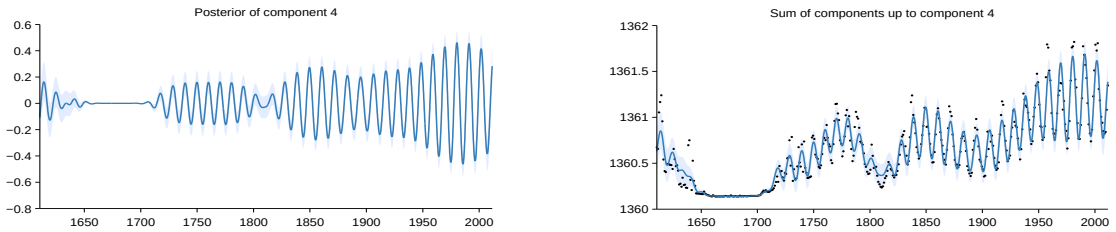


Figure 8: Pointwise posterior of component 4 (left) and the posterior of the cumulative sum of components with data (right)

Figure 5.10: Extract from an automatically-generated report describing the model components discovered by ABCD. This part of the report isolates and describes the approximately 11-year sunspot cycle, also noting its disappearance during the 16th century, a time known as the Maunder minimum (Lean et al., 1995).

The learned function for the solar irradiance data is

$$\text{Irradiance}(t) = 1361 + \alpha \sin(\beta + \gamma t) \sin(\delta + \epsilon t^2 - \zeta t)$$

where t is time and constants are replaced with symbols for brevity. This equation captures the constant offset of the data, and models the long-term trend with a product of sinusoids, but fails to capture the solar cycle or the Maunder minimum.

The learned function for the airline passenger data is

$$\text{Passengers}(t) = \alpha t + \beta \cos(\gamma - \delta t)\text{logistic}(\epsilon t - \zeta) - \eta$$

which captures the approximately linear trend, and the periodic component with approximately linearly (logistic) increasing amplitude. However, the annual cycle is heavily approximated by a sinusoid and the model does not capture heteroscedasticity.

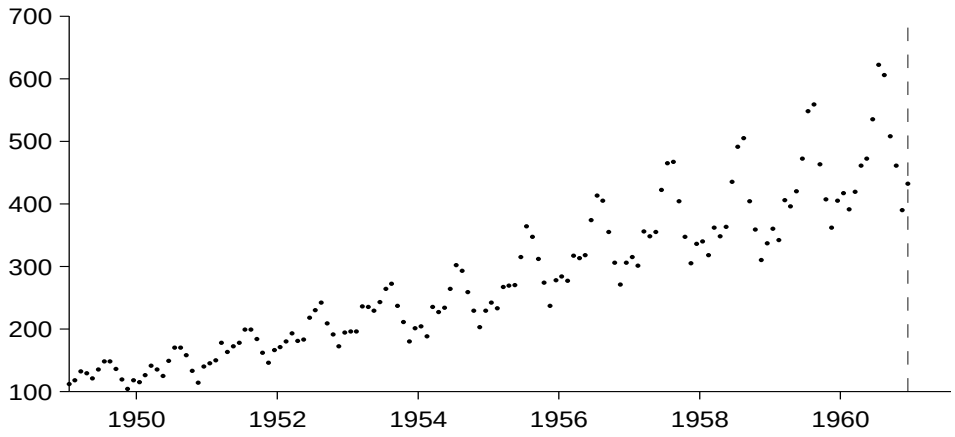


Figure 5.11: International airline passenger monthly volume (e.g. Box et al., 1976).

5.6 Related work

Section 4.4 describes work most closely related to the construction of Gaussian process models with complex kernels. In this section we review work relating to systems producing natural language output and the automation of data analysis.

Natural-language output To the best of our knowledge, our procedure is the first example of automatic description of nonparametric statistical models. However, systems with natural language output have been built in the areas of scene interpretation (Karpathy and Fei-Fei, 2014), video interpretation (Barbu and Lay, 2012) and automated theorem proving (Ganesalingam and Gowers, 2013).

Automatic machine learning Our work joins a growing literature attempting to automate various aspects of machine learning. Thornton et al. (2013) present AutoWEKA, a system which automates model selection in the data analysis software WEKA. Similarly the Google prediction API (Green et al., 2011) provides an API which hides the model selection and optimisation aspects of making predictions given training data. In a more restricted but no less important context, Snoek et al. (2012) demonstrated how Bayesian optimisation techniques could be used to tune the parameters of machine learning methods automatically, improving upon state of the art performance in a number of domains merely by improved parameter tuning. Our work moves this line of research forward by attempting to automatically explain a data set as well as building a model with good predictive performance.

The structure search algorithm has identified four additive components in the data. The first 2 additive components explain 98.5% of the variation in the data as shown by the coefficient of determination (R^2) values in table 1. The first 3 additive components explain 99.8% of the variation in the data. After the first 3 components the cross validated mean absolute error (MAE) does not decrease by more than 0.1%. This suggests that subsequent terms are modelling very short term trends, uncorrelated noise or are artefacts of the model or search procedure. Short summaries of the additive components are as follows:

- A linearly increasing function.
- An approximately periodic function with a period of 1.0 years and with linearly increasing amplitude.
- A smooth function.
- Uncorrelated noise with linearly increasing standard deviation.

#	R^2 (%)	ΔR^2 (%)	Residual R^2 (%)	Cross validated MAE	Reduction in MAE (%)
-	-	-	-	280.30	-
1	85.4	85.4	85.4	34.03	87.9
2	98.5	13.2	89.9	12.44	63.4
3	99.8	1.3	85.1	9.10	26.8
4	100.0	0.2	100.0	9.10	0.0

Figure 5.12: Short descriptions and summary statistics for the four components of the airline model.

5.7 Predictive Accuracy

In addition to our demonstration of the interpretability of ABCD, we compared the predictive accuracy of various model-building algorithms at interpolating and extrapolating time-series. ABCD outperforms the other methods on average.

Data sets We evaluate the performance of the algorithms listed below on 13 real time-series from various domains from the time series data library (Hyndman); plots of the data can be found in appendix B.

Algorithms We compare ABCD to equation learning using Eureqa (Nutonian, 2011) and six other regression algorithms: linear regression, GP regression with a single SE kernel (squared exponential), a Bayesian variant of multiple kernel learning (MKL) (e.g. Bach et al., 2004), change point modeling (e.g. Fox and Dunson, 2012; Garnett et al., 2010; Saatçi et al., 2010), spectral mixture kernels (Wilson and Adams, 2013) (spectral kernels) and trend-cyclical-irregular models (e.g. Lind et al., 2006).

The spans of the languages of kernels of the language used by ABCD and the kernel

2.2 Component 2 : An approximately periodic function with a period of 1.0 years and with linearly increasing amplitude

This component is approximately periodic with a period of 1.0 years and varying amplitude. Across periods the shape of this function varies very smoothly. The amplitude of the function increases linearly. The shape of this function within each period has a typical lengthscale of 6.0 weeks.

This component explains 89.9% of the residual variance; this increases the total variance explained from 85.4% to 98.5%. The addition of this component reduces the cross validated MAE by 63.45% from 34.03 to 12.44.

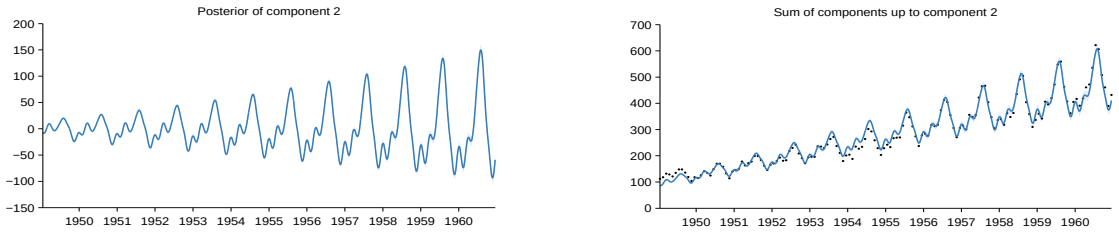


Figure 4: Pointwise posterior of component 2 (left) and the posterior of the cumulative sum of components with data (right)

Figure 5.13: Capturing non-stationary periodicity in the airline data.

search introduced in chapter 4 are very similar; the changes are close to being only a change of basis. Consequently their predictive accuracy is nearly identical so we only include ABCD in the results for brevity.

We use the default mean absolute error criterion when using Eureqa. All other algorithms can be expressed as restrictions of our modelling language (see table 5.1) so we perform inference using the same search methodology and selection criterion[3] with appropriate restrictions to the language. For MKL, trend-cyclical-irregular and spectral kernels, the greedy search procedure of ABCD corresponds to a forward-selection algorithm. For squared exponential and linear regression the procedure corresponds to marginal likelihood optimisation. More advanced inference methods are typically used for changepoint modelling but we use the same inference method for all algorithms for comparability.

We restricted to regression algorithms for comparability; this excludes models which regress on previous values of times series, such as autoregressive or moving-average models (e.g. Box et al., 1976). Constructing a language for this class of time-series model would be an interesting area for future research.

[3]We experimented with using unpenalised marginal likelihood as the search criterion but observed overfitting, as is to be expected.

2.4 Component 4 : Uncorrelated noise with linearly increasing standard deviation

This component models uncorrelated noise. The standard deviation of the noise increases linearly.

This component explains 100.0% of the residual variance; this increases the total variance explained from 99.8% to 100.0%. The addition of this component reduces the cross validated MAE by 0.00% from 9.10 to 9.10. This component explains residual variance but does not improve MAE which suggests that this component describes very short term patterns, uncorrelated noise or is an artefact of the model or search procedure.

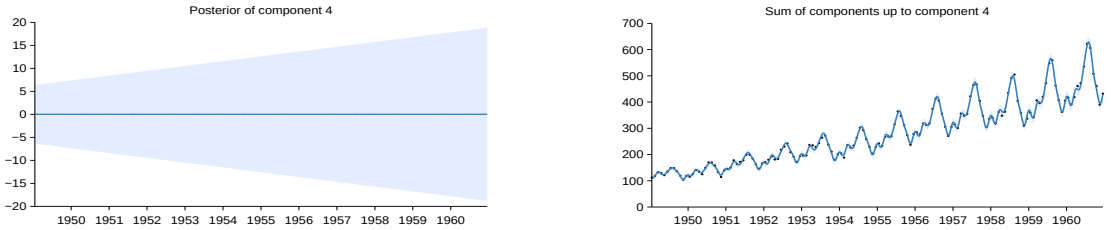


Figure 8: Pointwise posterior of component 4 (left) and the posterior of the cumulative sum of components with data (right)

Figure 5.14: Modeling heteroscedasticity in the airline data.

Interpretability versus accuracy BIC trades off model fit and complexity by penalising the number of parameters in a kernel expression. This can result in ABCD favouring kernel expressions with nested products of sums, producing descriptions involving many additive components. While these models have good predictive performance the large number of components can make them less easy to interpret. We experimented with distributing all products over addition during the search, causing models with many additive components to be more heavily penalised by BIC. We call this procedure ABCD-interpretability, in contrast to the unrestricted version of the search, ABCD-accuracy.

Extrapolation To test extrapolation we trained all algorithms on the first 90% of the data, predicted the remaining 10% and then computed the root mean squared error (RMSE). The RMSEs are then standardised by dividing by the smallest RMSE for each data set so that the best performance on each data set will have a value of 1.

Figure 5.15 shows the standardised RMSEs across algorithms. ABCD-accuracy outperforms ABCD-interpretability but both versions have lower quartiles than all other methods.

Overall, the model construction methods with greater capacity perform better: ABCD outperforms trend-cyclical-irregular, which outperforms Bayesian MKL, which outper-

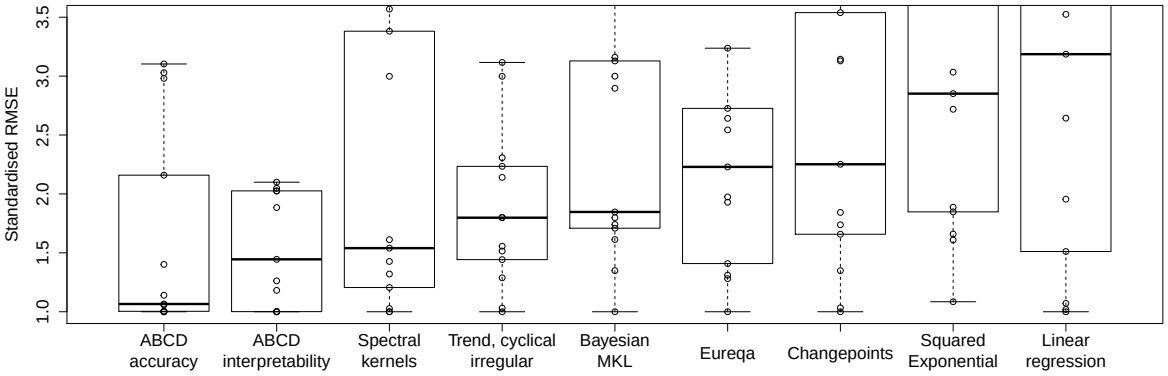


Figure 5.15: Raw data, and box plot (showing median and quartiles) of standardised extrapolation RMSE (best performance = 1) on 13 time-series. The methods are ordered by median.

forms squared exponential. Despite searching over a rich model class, Eureqa performs relatively poorly, since very few datasets are parsimoniously explained by a parametric equation.

Not shown on the plot are large outliers for spectral kernels, Eureqa, squared exponential and linear regression with values of 11, 493, 22 and 29 respectively. All of these outliers occurred on a data set with a large discontinuity (see the call centre data in appendix B).

Interpolation To test the ability of the methods to interpolate, we randomly divided each data set into equal amounts of training data and testing data. We trained each algorithm on the training half of the data, produced predictions for the remaining half and then computed the root mean squared error (RMSE). The values of the RMSEs are then standardised by dividing by the smallest RMSE for each data set i.e. the best performance on each data set will have a value of 1.

Figure 5.16 shows the standardised RMSEs for the different algorithms. The box plots show that all quartiles of the distribution of standardised RMSEs are lower for both versions of ABCD. The median for ABCD-accuracy is 1; it is the best performing algorithm on 7 datasets. The largest outliers of ABCD and spectral kernels are similar in value.

Changepoints performs slightly worse than MKL despite being strictly more general than Changepoints. The introduction of changepoints allows for more structured models, but it introduces parametric forms into the regression models (i.e. the sigmoids

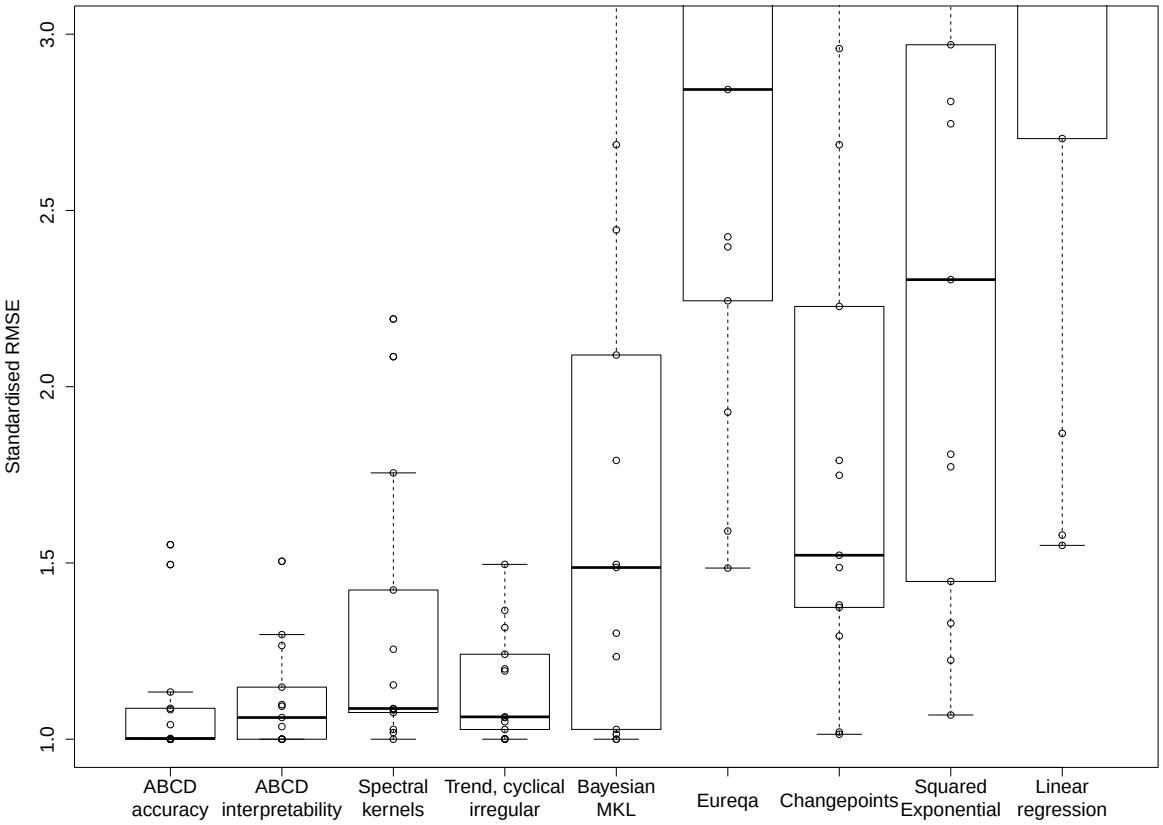


Figure 5.16: Box plot of standardised RMSE (best performance = 1) on 13 interpolation tasks.

expressing the changepoints). This results in worse interpolations at the locations of the change points, suggesting that a more robust modelling language would require a more flexible class of changepoint shapes or improved inference (e.g. fully Bayesian inference over the location and shape of the changepoint).

Eureqa is not suited to this task and performs poorly. The models learned by Eureqa tend to capture only broad trends of the data since the fine details are not well explained by parametric forms.

5.7.1 Tables of standardised RMSEs

See table 5.4 for raw interpolation results and table 5.5 for raw extrapolation results. The rows follow the order of the datasets in appendix B. The following abbreviations are used: ABCD-accuracy (ABCD-acc), ABCD-interpretability (ABCD-int), Spectral ker-

nels (SP), Trend-cyclical-irregular (TCI), Bayesian MKL (MKL), Eureqa (EL), Change-points (CP), Squared exponential (SE) and Linear regression (Lin).

ABCD-acc	ABCD-int	SP	TCI	MKL	EL	CP	SE	Lin
1.04	1.00	2.09	1.32	3.20	5.30	3.25	4.87	5.01
1.00	1.27	1.09	1.50	1.50	3.22	1.75	2.75	3.26
1.00	1.00	1.09	1.00	2.69	26.20	2.69	7.93	10.74
1.09	1.04	1.00	1.00	1.00	1.59	1.37	1.33	1.55
1.00	1.06	1.08	1.06	1.01	1.49	1.01	1.07	1.58
1.50	1.00	2.19	1.37	2.09	7.88	2.23	6.19	7.36
1.55	1.50	1.02	1.00	1.00	2.40	1.52	1.22	6.28
1.00	1.30	1.26	1.24	1.49	2.43	1.49	2.30	3.20
1.00	1.09	1.08	1.06	1.30	2.84	1.29	2.81	3.79
1.08	1.00	1.15	1.19	1.23	42.56	1.38	1.45	2.70
1.13	1.00	1.42	1.05	2.44	3.29	2.96	2.97	3.40
1.00	1.15	1.76	1.20	1.79	1.93	1.79	1.81	1.87
1.00	1.10	1.03	1.03	1.03	2.24	1.02	1.77	9.97

Table 5.4: Interpolation standardised RMSEs. The rows follow the order of the datasets in appendix B.

5.8 Conclusion and discussion

Towards the goal of automating exploratory statistical analyses we have presented a system which constructs an appropriate model from an open-ended language and automatically generates detailed reports that describe patterns in the data captured by the model. We have demonstrated that our procedure can discover and describe a variety of patterns on several time series. Our procedure's extrapolation and interpolation performance on time-series are highly competitive with existing model construction techniques. We believe that this line of research has the potential to make powerful statistical model-building techniques accessible to non-experts.

5.8.1 We should probably stop using the squared exponential kernel

With only the squared exponential kernel representing smoothness in the language, any kernel expression will have this one type of smoothness since the product of two SE kernels is another SE with different parameters. This is a cute mathematical property,

ABCD-acc	ABCD-int	SP	TCI	MKL	EL	CP	SE	Lin
1.14	2.10	1.00	1.44	4.73	3.24	4.80	32.21	4.94
1.00	1.26	1.21	1.03	1.00	2.64	1.03	1.61	1.07
1.40	1.00	1.32	1.29	1.74	2.54	1.74	1.85	3.19
1.07	1.18	3.00	3.00	3.00	1.31	1.00	3.03	1.02
1.00	1.00	1.03	1.00	1.35	1.28	1.35	2.72	1.51
1.00	2.03	3.38	2.14	4.09	6.26	4.17	4.13	4.93
2.98	1.00	11.04	1.80	1.80	493.30	3.54	22.63	28.76
3.10	1.88	1.00	2.31	3.13	1.41	3.13	8.46	4.31
1.00	2.05	1.61	1.52	2.90	2.73	3.14	2.85	2.64
1.00	1.45	1.43	1.80	1.61	1.97	2.25	1.08	3.52
2.16	2.03	3.57	2.23	1.71	2.23	1.66	1.89	1.00
1.06	1.00	1.54	1.56	1.85	1.93	1.84	1.66	1.96
3.03	4.00	3.63	3.12	3.16	1.00	5.83	5.35	4.25

Table 5.5: Extrapolation standardised RMSEs. The rows follow the order of the datasets in appendix B.

but a different kernel encoding for smoothness could have been chosen and the langauge of kernels restricted to only having one of these kernels in each product. In this sense the SE kernel has been used purely for convenience, rather than truly reflecting our beliefs about functions. Stein (1999) recommends instead the Matérn class of kernels, which possibly have more realistic smoothness assumptions than the squared exponential kernel in many situations. van der Vaart and van Zanten (2011) also demonstrate that the Matérn class of kernels produce regression algorithms with improved convergence rates compared to the square exponential for all but very smooth functions.

Chapter 6

Application of compositional kernels to a data mining competition

In this chapter I will discuss how I used Gaussian processes with compositionally constructed kernels as part of an ensemble of predictors in a data mining competition. In particular, this competition involved forecasting hourly loads of a US energy utility as part of the load forecasting track of the Global Energy Forecasting Competition 2012 hosted on Kaggle. The chapter begins with a description of the relatively simple techniques I used to be ranked 2nd in the competition; this description is based on Lloyd (2013). This work was completed before the work described in chapter 4 and hence the construction of kernels was not automated. The chapter therefore concludes with an investigation of whether the kernels I used could have been automatically constructed. This investigation identifies some short comings in the literature on approximation inference methods for Gaussian processes and I hypothesise about possible extensions to approximate inference methods that would overcome these difficulties.

6.1 Introduction

6.1.1 Competition and data

The data for this competition consisted of an hourly time series of energy utility loads at 20 zones with unknown location and temperatures at 13 stations also with unknown location. The data stretched over a period of 4.5 years and 8 non-consecutive weeks of load data were missing. The task of the competition was to fill in the missing 8 weeks (backcasting) and to predict loads for the week following the extent of the data

where temperature data was not available (forecasting); use of additional sources of temperature data was expressly prohibited by the competition organisers.

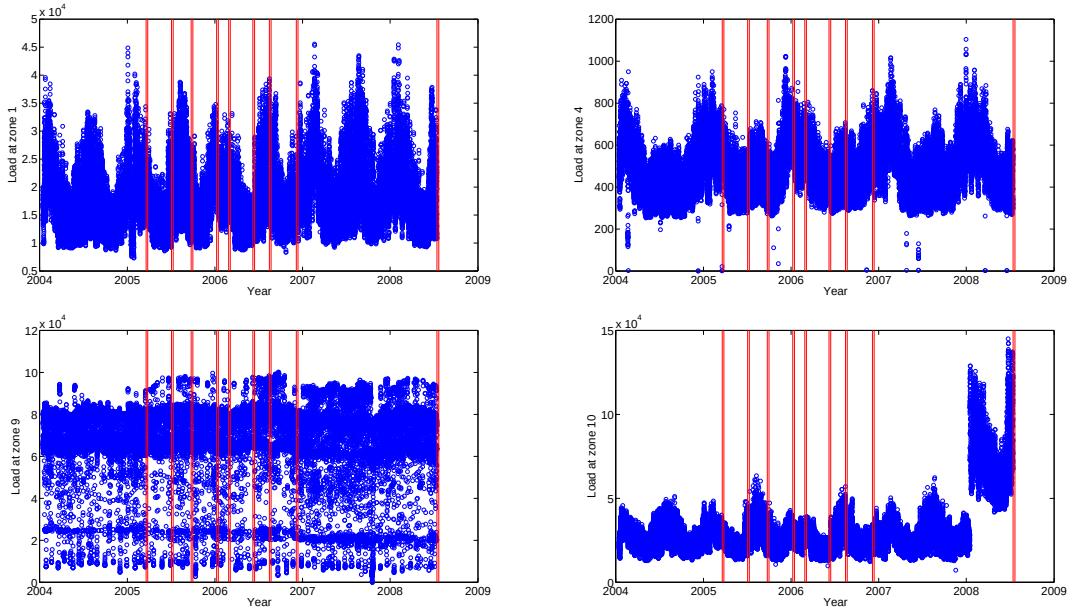


Figure 6.1: Load data at zones 1 (top left), 4 (top right), 9 (bottom left) and 10 (bottom right). All other zones look very similar to zones 1 and 4. The vertical lines indicate the start and end of the regions where loads were to be predicted.

The full data at four of the load zones is shown in figure 6.1. The other load zones are very similar to the top two plots showing clear seasonality; the bottom two plots are the only atypical zones. The vertical lines indicate the starts and ends of the backcasting and forecasting regions. Full temperature data at station 6 is shown in figure 6.2; all temperature stations look similar.

The evaluation metric of the competition (weighted root mean squared error) was heavily biased towards good performance on the forecasting task (much higher weightings were placed on forecast errors). Whilst the competition was running, contestants could submit predictions to the Kaggle website and obtain the value of the evaluation metric evaluated on 25% of the held out data. This value was known as the public score since it was published during the competition. Perhaps not too surprisingly this resulted in many participants overfitting heavily to this particular split of the data; the leaderboard changed substantially when final scores were evaluated on the remaining 75% of the data (yielding the private scores). The final public and private scores for the top 5 teams are shown in table 6.1. Further details can be found in the competition summary paper (Hong et al., 2014).

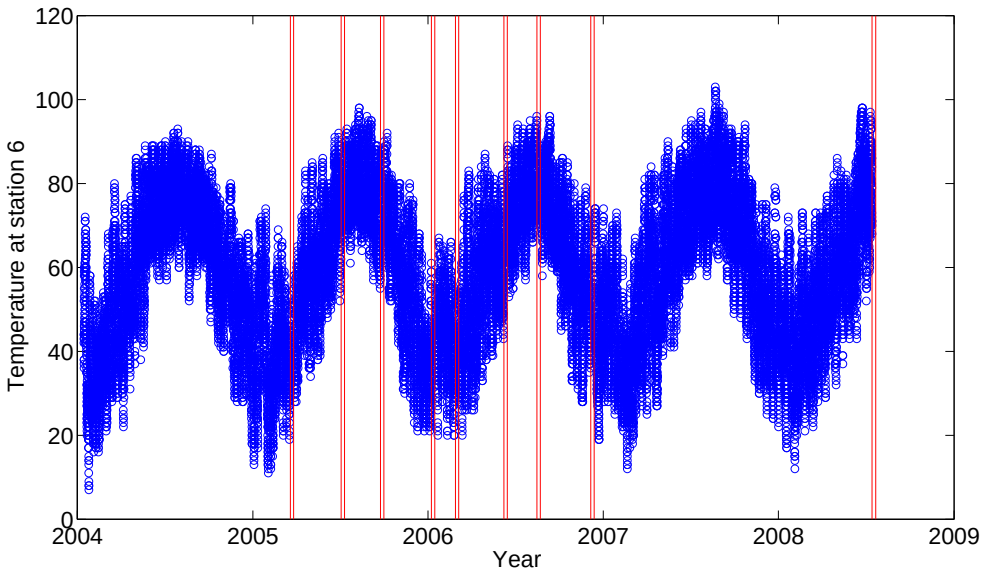


Figure 6.2: Temperature data at station 6. All other stations look very similar. The vertical lines indicate the start and end of the regions where loads were to be predicted.

6.1.2 Structure of chapter

This chapter is split into sections describing each technique used for forecasting temperatures and loads. Within each section I have motivated the particular choice of algorithm, discussed how it was used and described how to replicate the results using the spreadsheets and scripts available at `https://github.com/jamesrobertlloyd/GEFCOM2012`. The source code is not necessary for understanding this chapter, it is referenced purely for those interested in replicating my work (submissions to the competition were required to be entirely replicable). The chapter then concludes with attempts to apply the kernel construction procedure of chapter 4 in order to automatically build the kernels I used in this competition.

6.1.3 Methodology

I approached the competition in the spirit of a data mining competition. Consequently, some of the choices detailed below were based on intuition to save time and focus on the aspects of the data most likely to give the largest increases in performance. Many of these choices could have been replaced by appropriate searches and cross validation; where more complex techniques would be required I have briefly described how one could perform a more objective analysis. In general this chapter is much less rigorous

Team name	Public score	Private score
CountingLab	70700	67215
James Lloyd	71164	71416
Tololo	52669	71780
TinTin	64352	73307
Quadrivio	72825	78196

Table 6.1: Scores of top 5 entries to GEFCom. The public score was evaluated on 25% of the test data and was known to all participants during the challenge. The private score was evaluated on the other 75% of the data at the end of the competition.

then the rest of this thesis but it is hoped it provides an interesting example of the practical uses of the techniques described in chapter 4 and also highlights how data analysis actually proceeds when there are strict time deadlines and the only measure of success is predictive performance.

My methodology for load back / forecasting was to try different general purpose regression algorithms and then ensemble (average) the predictions. The final ensemble comprised predictions from a gradient boosting machine (GBM), Gaussian process (GP) regression and the benchmark solution provided by the competition organisers (a linear model).

6.2 Data cleansing

A sensible first step in any prediction task is to look at your data; in particular searching for anomalies. This was performed for both temperature and load data using the spreadsheets `temp/temp.xlsx` and `load/load.xlsx` by plotting the data as time series and using various types of conditional formatting to search for irregularities visually.

This visual analysis revealed a large discontinuity in load series 10 and atypical dynamics in series 9 (see figure 6.3 and others in this chapter for comparison). No other large anomalies were detected during the first inspection of the data and smaller irregularities were not revisited since the algorithms detailed below performed acceptably well without further data cleansing.

No adjustments were made for holidays or other irregular events such as electricity black-outs. Ignoring them was merely a practicality of time constraints rather than a design choice.

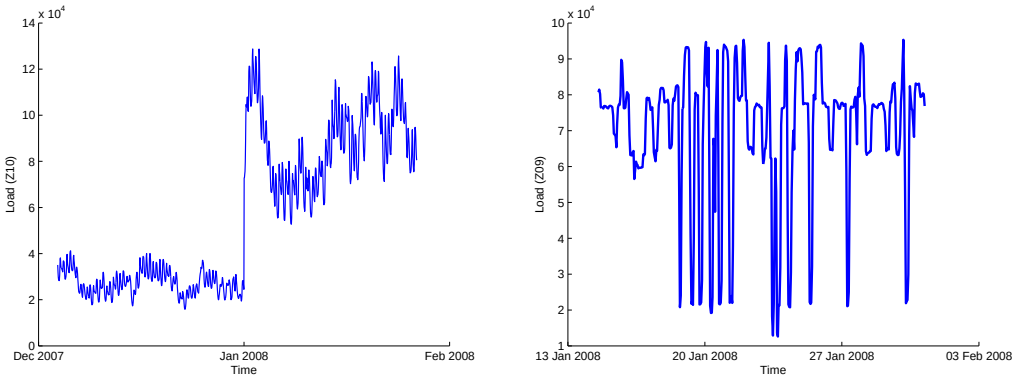


Figure 6.3: Left: Raw data at zone 10, showing a large discontinuity. Right: Raw data at zone 9, showing atypical load dynamics.

6.3 Temperature forecasting

6.3.1 Initial analysis and remarks

The error metric used in GEFCom2012 was heavily weighted towards times at which temperatures were unknown (i.e. the load forecast rather than backcasts). Consequently, good temperature predictions seemed crucial for overall success.

When submitting a solution to Kaggle, the error metric was computed on 25% of the held out data and returned to the user. This allowed a user to optimise temperature predictions by optimising the score of resulting load predications (i.e. computing load predictions based on different temperature predictions and selecting the temperature prediction with the highest corresponding load prediction score). Depending on how the test data was split (into the 25% test and 75% validation partition) this may have allowed users to come very close to knowing the true future temperatures.

I therefore used a flexible but simple method for forecasting temperatures that could be easily tuned. Figure 6.4 shows data from temperature station 1 along with various curves used for prediction (described later). The black solid line is the raw data, which shows that temperatures follow a smooth trend with a daily pattern of rising and falling temperatures. For simplicity, I modelled the smooth trend and daily periodicity separately and modelled each temperature station in isolation.

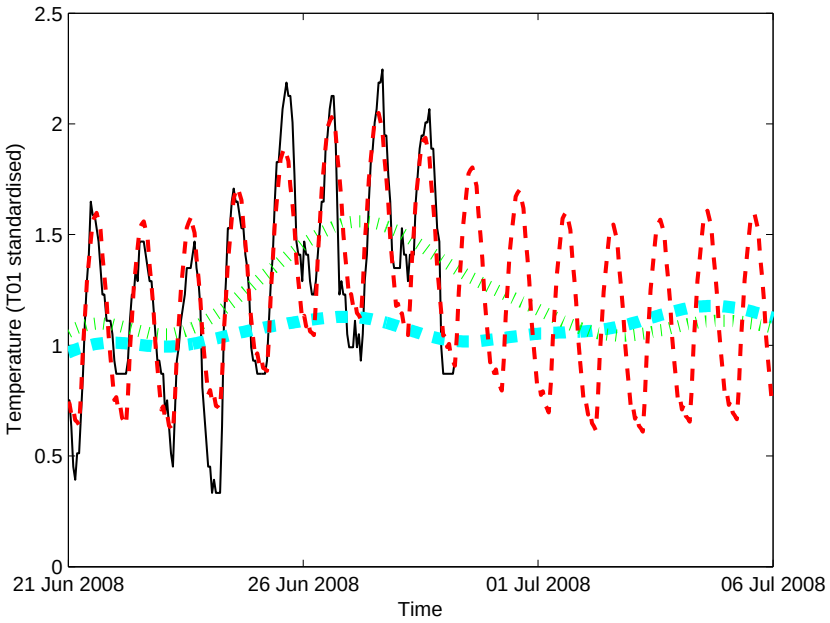


Figure 6.4: Raw data at temperature station 1 (after removing mean and scaling to have unit standard deviation) (black solid). Historic average of smoothed temperatures (blue dashed thick). Current and predicted smoothed temperatures (green dotted thick) and final temperature predictions (red dashed).

6.3.2 Methodology

The smooth trend was estimated within the data using local linear regression (e.g. chapter 6 of Hastie et al., 2009) with a bandwidth of one day; see section 6.7 for a discussion of how this method could have been usefully replaced by Gaussian process regression. I assumed that the smooth temperature trend would, on average, smoothly return to the historical average. A suitable modelling technique that would give this type of prediction is Gaussian process regression; the difference between the smoothed temperature and its historical average was regressed against time using a kernel of the form SE + WN i.e. a smooth function plus noise.

In Figure 6.4, the thick blue dashed curve shows the historical average of smoothed temperatures and the thick green dotted line shows the current smoothed temperature and prediction. The parameters of the GP model (length scale and scale factor of the squared exponential kernel) were tuned by hand, initially choosing sensible values based on plots of the data and then trying to optimise the public test score of derived load

predictions.

The difference between the temperature and the smoothed temperature was assumed to be a smooth periodic function with a period of one day. This modelling assumption was implemented using a Gaussian process with a periodic kernel (again regressing the difference against time). Parameters of the GP model were chosen by optimising the marginal likelihood of the model given particular parameters using optimisation methods supplied in the GPML toolbox[1]. Only the last 250 data points (which corresponds to approximately 10 days) were used to fit this part of the model in order to capture recent temperature dynamics.

6.3.3 Replication of results

The spreadsheet and scripts used to manipulate the temperature data can be found in the folder `temp`. The MATLAB script makes use of the GPML toolbox which is contained in the folder `GPML`. The temperature data was reshaped into a (time) $\times$ (temperature station) array using the spreadsheet `temp.xlsx` and then saved in `times_series_raw.csv`. The method described above is implemented by the MATLAB script `predict_temp.m` which saves the predicted time series in `GP_pred_temp.csv` and the smoothing of these predictions in `smooth_temp_GP.mat` and `smooth_temp_GP.csv`.

6.3.4 Discussion

The red curve in Figure 6.4 shows the result of the methodology applied to temperature station 1. The fit to the data is far from perfect but it has captured the broad features of the data. Given the time constraints of this competition I did not investigate the temperature / weather forecasting literature; far better methods must certainly exist!

6.4 Gradient boosting machines

6.4.1 Initial analysis and remarks

Figure 6.5 plots a subset of the data for loads at zone 1 and temperature station 9. We can see that that the load has a smoothly varying trend with roughly periodic daily deviations from this trend. The load also appears to be negatively correlated with

[1] `http://www.gaussianprocess.org/gpml/code/matlab/doc/`

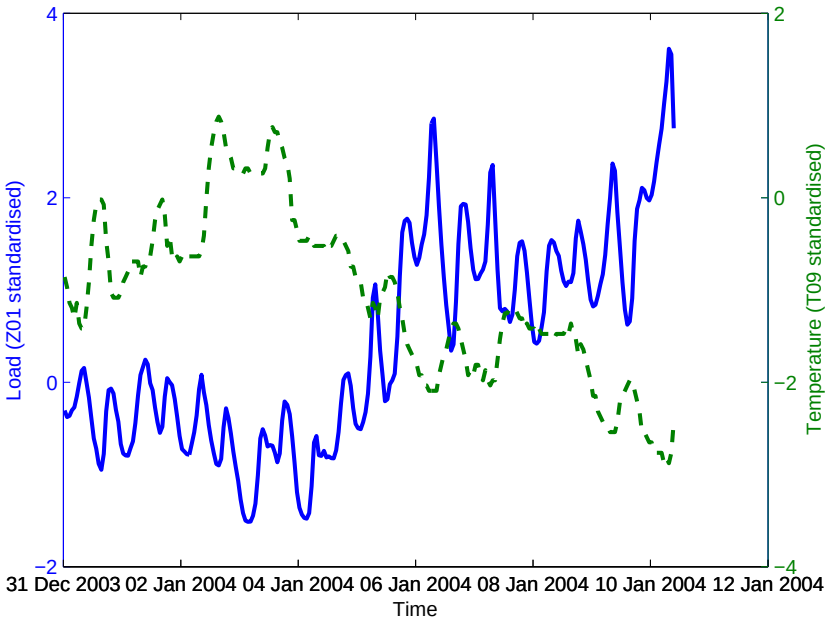


Figure 6.5: Raw data at load zone 1 (after removing mean and scaling to have unit standard deviation) (blue solid) and raw data at temperature station 9 (after removing mean and scaling to have unit standard deviation) (green dashed).

temperature in this region of data. It is also reasonable to assume that load dynamics will be different on weekends compared to weekdays.

Based on these observations, I modelled loads as a function of time of day, time within week, temperatures and smoothed temperatures (all stations). All zones were modelled independently for simplicity.

6.4.2 Methodology

A standard 'black-box' technique for learning an unknown regression function is that of using gradient boosting machines (GBM) (e.g. chapter 10 of Hastie et al., 2009). Gradient boosting machines here refer to a method for learning a sum of decision trees using boosting with bootstrap samples of the data when learning each decision tree. For each zone I used all of the data to train a GBM and then used that to predict all outputs i.e. I learnt a global function for each zone.

I used a standard R implementation of GBM using most of the default settings[2]. After some brief experimentation with parameter values I used a shrinkage factor of

0.01, 10000 trees and an interaction depth of 3. I experimented with other parameter values, using the public test score as a guide, but nothing I tried improved upon my initial guess and therefore I did not expend much effort trying to optimise parameter values.

6.4.3 Replication of results

The scripts and data used can be found in the folder `gbm`. The load data was reshaped into a (time) × (load zone) array using the spreadsheet `load.xlsx` in the folder `load`. This was then concatenated with the temperature predictions and smoothed temperature predictions and saved in `gbm_input.csv`. The R script `basic_gbm.R` then performs the method described above to produce predictions which are saved in `gbm_output.csv`.

6.4.4 Discussion

The time series for zone 10 had a very large discontinuity (see left of figure 6.3). GBM was used as a global prediction method which means it will have produced very bad predictions for zone 10. I later fixed this by modelling the loads either side of the discontinuity separately. Perversely, this resulted in a worse public test score, despite being a much more sensible model. However, this change resulted in a much improved private test score; it is all too easy to overfit without rigid discipline!

6.5 Gaussian processes

6.5.1 Initial analysis and remarks

The observations made about the load data in section 6.4.1 (i.e. smooth trends, near periodic variations, dependence on temperature) can be modelled using Gaussian process regression. As demonstrated in chapter 4 specifying the kernel function of a Gaussian process allows one to encode these structural assumptions into a regression model.

[2] `http://cran.r-project.org/web/packages/gbm/index.html`

6.5.2 Methodology

Three different kernel functions were used; one for backcasting, one for forecasting and one for backcasting zone 9. The forms of the three kernels used were as follows[3]

$$\mathrm{SE}_t + \mathrm{SE}_S + \mathrm{SE}_t \times \mathrm{SE}_T \times \mathrm{Per}_t \tag{6.5.1}$$

$$\mathrm{SE}_t + \mathrm{SE}_t + \mathrm{Per}_t \tag{6.5.2}$$

$$\mathrm{SE}_t + \mathrm{SE}_S + \mathrm{SE}_T \times \mathrm{Per}_t \tag{6.5.3}$$

where the subscripts indicate which dimension the kernel acts upon; (t)ime, (T)emperature or (S)moothed temperature[4]. For backcasting, the model was applied to the 1000 surrounding data points; for forecasting the last 500 data points were used. Only one temperature station was used in this method; for each back / forecasting region the model evidence of the model was computed for each temperature station. Bayesian model averaging (e.g. Hoeting et al., 1999) was then used to combine the predictions. In practice this was usually numerically equivalent to selecting the model / temperature station with the highest model evidence.

Kernel (6.5.1) encodes the assumptions that loads can be explained as a smoothly varying trend and an approximately periodic component. The smooth trend is the sum of a smooth function of smoothed temperature and a smooth function of time (to explain any deviations from the trend implied by temperature). The periodic component can vary through time and by being multiplied by SE_t and SE_T the shape of the periodicity will be more similar to recent times and any points in time with similar temperatures. This kernel was used for backcasting all zones apart from zone 9. An example of this kernel in action is shown in Figure 6.6. The left plot shows the load data (black solid) and the prediction (red dashed). On the right is the temperature time series which resulted in the model with the highest model evidence. Comparing the two plots we can see that the deviations of the prediction from the trend through time correspond to fluctuations in the temperature time series.

Kernel (6.5.2) encodes the assumption that loads can be explained as the sum of two smooth functions with different length scales and an exactly periodic smooth function. This simpler kernel was used for forecasting since it gave more stable extrapolations of the data.

Kernel (6.5.3) is a simpler version of kernel (6.5.1) used for backcasting zone 9.

[3] See source code for parameter values

[4] A local linear regression of temperature with a bandwidth of one day.

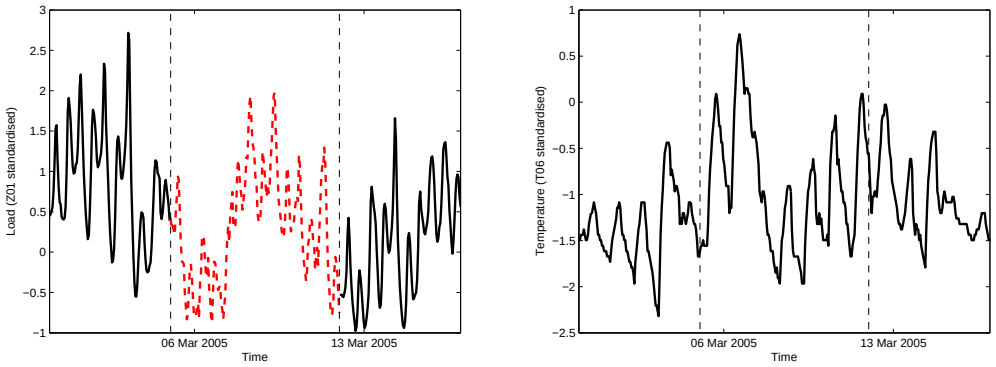


Figure 6.6: Left: Raw data (after scaling and translation) (black solid) and Gaussian process predictions (red dashed) for zone 1. Right: Corresponding temperature time series used for prediction. Notice that the peak in temperature corresponds to a trough in the load predictions i.e. the model has predicted a negative correlation between temperatures and loads.

The periodic component is now only similar to other times with similar temperatures. This prevented the predictions from being so greatly affected by the large isolated load fluctuations present in zone 9.

The forms and parameters of the kernel functions (see the accompanying code for all of the values) were chosen by trial and error (observing which kernels gave reasonable looking predictions and computing public test scores).

6.5.3 Replication of results

The spreadsheet and scripts used can be found in the folder `gp`. The load data was reshaped into a (time) $\times$ (load zone) array using the spreadsheet `load.xlsx` in the folder `load` and then saved in `load_input.csv` which encodes missing values as zeros. The outputs of the temperature prediction `GP_pred_temp.csv` and `smooth_temp_GP.mat` are used directly.

The MATLAB script `gp_elec.m` performs the methodology described above, saving the predictions to `gp_pred.csv`.

6.5.4 Discussion

Selecting the form and parameters of the kernels by hand required a moderate amount of familiarity with Gaussian processes. It is likely that better performance could have been achieved by the larger and more principled search over kernel functions introduced in chapter 4. Barriers to its immediate application are however discussed in section 6.7.

6.6 Selecting the final ensemble

6.6.1 Remarks

The final prediction was formed as the ensemble (weighted average) of predictions from three separate methods; gradient boosting machines, Gaussian process regression and the benchmark solution provided by the competition organisers (a linear model). Ensembling is a very powerful technique for combining predictions from separate models. If two predictions (viewed as random quantities) are statistically uncorrelated then their average will have lower variance than either one separately.

In practice, different algorithms are correlated and ensembling highly correlated predictions will rarely lead to improved performance. In particular, I also tried the random forest algorithm (Breiman, 2001) but the predictions were too correlated with gradient boosting machines (they are both based on an ensemble of decision trees) to be a useful addition to the ensemble.

6.6.2 Methodology

The ensemble weights were chosen by hand, using the public test score as the metric to be optimised. The public test scores did not vary considerably once sensible parameters had been found so more advanced techniques were not used to optimise the ensemble weights.

6.6.3 Replication of results

The spreadsheets used to manipulate model output can be found in the folder `ensemble`. The spreadsheet `sub_creator.xlsx` was used to convert (time) $\times$ (load zone) arrays of model output into the required format for submission. The output of this spreadsheet was then copied in `ensemble_creator.xlsx` which performs the averaging to produce ensembles of the predictions.

6.6.4 Discussion

This method could have been been slightly improved by programmatically selecting the ensemble weights, optimising some form of validation metric. One could also have tried to optimise the public test score directly. Since only 2 submissions could be tested per day, one would want to use an optimisation technique that made very efficient use of data (e.g. Osborne et al., 2009; Snoek et al., 2012).

Larger improvements could likely be achieved by considering a larger number of base predictions. It is likely that the algorithms used by other competitors in GEFCom2012 would provide useful additions to the ensemble of predictions. See Hong et al. (2014) and the references therein for an overview of the methods used by other participants.

6.7 An attempt to construct the composite kernels automatically

My participation in this data mining competition provided excellent experience before working on the research described in chapters 4 and 5. A natural question to ask is whether or not the compositional kernel search could have been directly applied to this competition, potentially speeding up my development of models and improving performance. The answer to this question is 'mostly' and reveals some interesting shortcomings in methods of approximate inference for Gaussian processes. We now present a detailed analysis of the temperature data. These comments apply similarly to the load data.

The function used to forecast temperatures displayed in figure 6.4 can be described as

$$\begin{aligned} &\text{Average historical smoothed temperatures} && (6.7.1)\\ +\ &\text{A smooth function with a lengthscale of a few days} && (6.7.2)\\ +\ &\text{A daily periodic function} && (6.7.3) \end{aligned}$$

which could alternatively be described as

$$\begin{aligned} &\text{An annual periodic function} && (6.7.4)\\ +\ &\text{A smooth function with a lengthscale of a few days} && (6.7.5)\\ +\ &\text{A daily periodic function} && (6.7.6) \end{aligned}$$

which could be modelled by a Gaussian process with kernel ZMPer + SE + ZMPer with appropriate parameters.

Exact inference for Gaussian processes has $\mathcal{O}(N^3)$ computational complexity where N is the number of data points. This makes it (currently) prohibitive to perform exact inference for N much larger than 1000. The complete GEFCom dataset contained roughly 40,000 data points meaning that one has to consider approximate inference schemes.

A simple approximate inference scheme is to use a subset of the data to perform inference(e.g. Quiñonero Candela and Rasmussen, 2005). Running the kernel search procedure on a randomly selected 500 data point subset of the data for temperature station 1 results in a model composed of the following additive components:

- A very smooth function
- A periodic function with a period of 1.0 years
- An approximately periodic function with a period of 24.0 hours
- Uncorrelated noise

This model has captured the long term trend of the data and the two forms of periodicity but it has failed to identify that short term (over several days) temperature fluctuations are temporally correlated (this is due to weather effects). This will result in short term extrapolations that are potentially separated by a discontinuity from the data.

Presented with this failure one might consider instead using a contiguous subset closest to the times at which extrapolation will be performed. The results of running the kernel search on this data subset is a model composed of

- A constant
- A periodic function with a period of 24.0 hours
- A smooth function with a lengthscale of 16.5 hours
- A rapidly varying smooth function with a lengthscale of 2.0 hours
- Uncorrelated noise

which is visualised in figure 6.7. This model has successfully captured the correlation in the data at a scale of days but since it only had access to about 20 days of data it

is completely unaware of the annual periodicity / historical average temperatures. Not including the annual periodicity may result in poor extrapolations when looking more than a few days ahead.

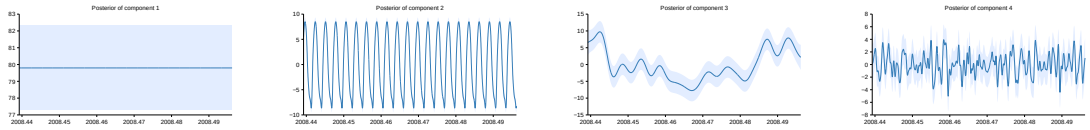


Figure 6.7: Model constructed for contiguous subset of temperature data. From left to right: Constant offset, daily periodicity and smooth functions.

What we have observed here is that different approximation strategies are appropriate for learning different types of function. Exactly periodic functions or smooth functions with very long lengthscales can be learnt effectively using a random subset of the data. In contrast, shorter lengthscale components can be learnt by assuming that the function being learnt is not strongly affected by data far away from where the function is being modelled. This is representative of a common theme; while there are myriad methods for performing approximate inference in Gaussian processes almost all of them either exploit the fact that long lengthscale components do not require much data to be learnt effectively and short lengthscale components have approximate statistical independence relationships that allow for local approximate inference methods. The above example demonstrates that approximate inference methods may need to deal with these two types of structures simultaneously.

Continuing with the theme of using a subset of data one could construct a subset of data which is a mixture of the two methods already described. Using this data subset results in a model composed of

- A very smooth function
- A periodic function with a period of 1.0 years
- An approximately periodic function with a period of 24.0 hours
- A rapidly varying smooth function with a lengthscale of 19.5 hours
- A rapidly varying smooth function with a lengthscale of 2.0 hours
- Uncorrelated noise

By using this subset the kernel search has been able to infer both periodicities and smooth functions at multiple lengthscales. However, using my own prior knowledge about temperature dynamics a more complete description of the data might be

$$\begin{aligned} &\text{An annual periodic function} && (6.7.7)\\ &+ \text{A long term smooth function} && (6.7.8)\\ &+ \text{Shorter term smooth functions} && (6.7.9)\\ &+ \text{A daily approximately periodic function} && (6.7.10)\\ &+ \text{A daily periodic function which is similar to values in different years} && (6.7.11) \end{aligned}$$

where the final term represents that the average daily profile of temperatures might change shape and size depending on the time of year. The last term could be modelled with a Gaussian process using a kernel of the form $\mathrm{SE} \times \mathrm{Per} \times \mathrm{Per}$[5]. The most effective subset of data, or more generally type of approximation scheme, is different for these different functions. The periodicity and long term smooth function can be learned with a random subset of the data. The shorter term smooth functions will require more data to be learned effectively but one will be able to make use of approximate statistical independence relationships between far away data points to learn the function at any point using local data. For the final function, to learn the function at any point one only needs to see data nearby in time and the corresponding time points in previous years.

It has previously been remarked that approximate inference strategies should be able to capture both global and local functions (e.g. Snelson and Ghahramani, 2007; Vanhatalo and Vehtari, 2012). However, I am unaware of any approximate inference strategy that attempts to exploit more than two lengthscales simultaneously. It seems reasonable that approximate inference strategies that exploit all aspects of structured kernels would make for profitable research.

6.8 Conclusion

In this competition I applied general purpose machine learning / regression algorithms and made few adjustments for the specific domain the data arose from. Despite this, the resulting predictions were highly competitive. It is hoped that this report shows that

[5]This term was not inferred on any subsets of data. It is possible that the data does not actually display this changing periodicity pattern that I expect; it seems more likely however that 500 data points were insufficient evidence for such a complicated structure to be inferred.

simple techniques can be very effective for predictive modelling and can inspire practical techniques for load forecasting.

With the benefit of hindsight I also demonstrated that some of the models used in this competition could have been automatically constructed using the methods described in chapters 4 and 5. However, to do so required a careful choice of approximate inference strategy and revealed a potential area for future research in approximate inference methods for Gaussian processes.

Chapter 7

Statistical model criticism using kernel two sample tests

The automatically produced statistical reports that were the output of the work of chapter 5 all conclude with a model criticism section. In this section, the statistical model constructed by the kernel search algorithm is presented to the user as a hypothesis and as such it subjected to criticism and attempts to falsify its claims. On entering the literature of statistical model criticism I quickly found myself in the cross fire of a version of the frequentists versus Bayesians argument which appears to have been left unresolved. Moreover, I have found there to be little actionable advice on which methods of model criticism to use when, which presented a challenge when trying to select appropriate model criticism methods to be used by a system capable of constructing infinitely many different statistical models. This chapter is my initial response to these gaps in the literature.

7.1 Introduction

Statistical model checking or criticism[1] is an important part of a complete statistical analysis. When one fits a linear model to a data set via least squares a complete analysis includes computing e.g. Cook's distances (Cook and Weisberg, 1982) to identify influential points or plotting residuals against fitted values to identify non-linearity or heteroscedasticity. Similarly, modern approaches to Bayesian statistics view model crit-

[1] We follow Box (1980) using the term 'model criticism' for similar reasons to O'Hagan (2003), namely "model checking ... conveys to me a sense of a rather passive activity that does not expect to uncover any problems, while model criticism conveys a more vigorous and open-minded process".

icism as in important component of an iterative cycle of model construction, inference and criticism (e.g. Blei, 2014; Gelman et al., 2013).

As statistical models become more complex and diverse in response to the challenges of modern data sets there will be an increasing need for a greater range of model criticism procedures that are either automatic or generally applicable. This will be especially true as automatic modelling methods (e.g. Grosse et al., 2012; Lloyd et al., 2014; Thornton et al., 2013) and probabilistic programming (e.g. Goodman et al., 2008; Milch et al., 2005; Stan Development Team, 2014) mature.

Model criticism typically proceeds by choosing a statistic of interest, computing it on data and comparing this to a suitable null distribution. Ideally these statistics are chosen to assess the utility of the statistical model under consideration (see applied examples (e.g. Gelman et al., 2013; Meulders et al., 1998)) but this can require considerable expertise on the part of the modeller. We propose a conceptually different approach by using a statistic defined as a supremum over a broad class of measures of discrepancy between two distributions, the maximum mean discrepancy (MMD) (e.g. Borgwardt et al., 2006; Gretton et al., 2008)). The advantage of this approach is that the discrepancy measure attaining the supremum automatically identifies regions of the data which are most poorly represented by the statistical model fit to the data.

We demonstrate this approach to model criticism on toy data sets, restricted Boltzmann machines and deep belief networks trained on MNIST digits and Gaussian process (e.g. Rasmussen and Williams, 2006) regression models trained on several time series. Our proposed method identifies discrepancies between the data and fitted models that would not be apparent from the predictive performance focused metrics one typically finds in a machine learning paper. It is our belief that more effort should be expended on attempting to falsify models fitted to data, using model criticism techniques or otherwise. Not only will this aid research in targeting areas for improvement but it should give greater confidence in any conclusions drawn from a model.

7.2 Model criticism

Suppose we observe data $Y^{\text{obs}} = (y_i)_{i=1\ldots n}$ and we attempt to fit a model M with parameters θ. After performing a statistical analysis we will have either an estimate, $\hat{\theta}$, or an (approximate) posterior, $p(\theta \,|\, Y^{\text{obs}}, M)$, for the parameters. How can we check the validity of this analysis?

7.2.1 Criticising prior assumptions

The classical approach to model criticism is to attempt to falsify the null hypothesis that the data could have been generated by the model M for some value of the parameters θ i.e. $Y^{\text{obs}} \sim p(Y \mid \theta, M)$. This is typically achieved by constructing a statistic T of the data whose distribution does not depend on the parameters θ i.e. a pivotal quantity. The extent to which the observed data Y^{obs} differs from expectations under the model M could then be quantified with a tail-area based p-value

$$p_{\text{freq}}(Y^{\text{obs}}) = \mathbb{P}(T(Y) \geq T(Y^{\text{obs}})) \quad \text{where} \quad Y \sim p(Y \mid \theta, M) \quad \text{for any } \theta. \tag{7.2.1}$$

Analogous quantities in a Bayesian analysis are the prior predictive p-values of Box (Box, 1980). The null hypothesis is replaced with the claim that the data could have been generated from the prior predictive distribution $Y^{\text{obs}} \sim \int p(Y \mid \theta, M)p(\theta \mid M)\mathrm{d}\theta$. A tail-area p-value can then be constructed for any statistic T of the data

$$p_{\text{prior}}(Y^{\text{obs}}) = \mathbb{P}(T(Y) \geq T(Y^{\text{obs}})) \quad \text{where} \quad Y \sim \int p(Y \mid \theta, M)p(\theta \mid M)\mathrm{d}\theta. \tag{7.2.2}$$

Both of these procedures construct a function of the data $p(Y^{\text{obs}})$ whose distribution under a suitable null hypothesis is uniform i.e. a p-value. The p-value quantifies how surprised one should be after observing data Y^{obs} having expected it to have been generated by the model. The different null hypotheses reflect the different uses of the word 'model' in frequentist and Bayesian analyses. A frequentist model is a class of probability distributions over data indexed by parameters whereas a Bayesian model is a joint probability distribution over data and parameters.

7.2.2 Criticising estimated models or posterior distributions

A contrasting method of Bayesian model criticism is the calculation of posterior predictive p-values (e.g. Guttman, 1967; Rubin, 1984) p_{post} where the prior predictive distribution is replaced with the posterior predictive distribution $Y \sim \int p(Y \mid \theta, M)p(\theta \mid Y^{\text{obs}}, M)\mathrm{d}\theta$. The corresponding test for an analysis resulting in a point estimate of the parameters $\hat{\theta}$ would use the plug-in predictive distribution $Y \sim p(Y \mid \hat{\theta}, M)$ to form the plug-in p-value p_{plug}.

These p-values quantify how surprised one should be if, after performing inference having observed data Y^{obs}, one were to observe new data whose value of the statistic T was equal to that of the original data. Put more simply, they quantify how surprising

the data Y^{obs} is even after having observed it. A simple variant of this method of model criticism is to use held out data Y^*, generated from the same distribution as Y^{obs}, to compute a p-value i.e. $p(Y^*) = \mathbb{P}(T(Y) \geq T(Y^*))$. This quantifies how surprising the held out data is after having observed Y^{obs}.

7.2.3 Which type of model criticism should be used?

Different forms of model criticism are appropriate in different contexts, but we believe that posterior predictive and plug-in p-values will be most often useful for the types of statistical model considered in the machine learning literature. For example, suppose one is fitting a nonparametric or otherwise very flexible model to data e.g. a deep belief network. Classical p-values would assume a null hypothesis that the data could have been generated from some deep belief network. Since the space of all possible deep belief networks is very large it will be difficult to ever falsify this hypothesis. A more interesting null hypothesis to test in this example is whether or not our particular deep belief network can faithfully mimic the distribution it was trained on. This is the null hypothesis of posterior or plug-in p-values.

7.3 Model criticism for i.i.d. data using two sample tests

We assume that our data Y are i.i.d. samples from some unknown distribution $(y_i)_{i=1\ldots n} \overset{\text{iid}}{\sim} p(y \mid \theta, M)$. After performing inference resulting in a point estimate of the parameters $\hat{\theta}$, the null hypothesis associated with a plug-in p-value is $(y_i^{\text{obs}})_{i=1\ldots n} \overset{\text{iid}}{\sim} p(y \mid \hat{\theta}, M)$.

We can test this null hypothesis using a two sample test (e.g. Bickel, 1969; Friedman and Rafsky, 1979; Hotelling, 1951). In particular, we have samples of data $(y_i)_{i=1\ldots n}$ and we can generate samples from the plug-in predictive distribution $(y_i^{\text{rep}})_{i=1\ldots m} \overset{\text{iid}}{\sim} p(y \mid \hat{\theta}, M)$ and then test whether or not these samples could have been generated from the same distribution.

The methods for constructing p-values in the previous section all started with a statistic T of the entire data. This however is not necessary; a p-value is simply a random variable which has a uniform distribution under the null hypothesis. Instead of computing a statistic $T(Y^{\text{obs}})$ of the entire data we consider statistics of data points $t(y)$

and then consider the mean discrepancy between the two samples

$$\mathbb{E}(t(y^{\text{rep}})) - \mathbb{E}(t(y)). \tag{7.3.1}$$

This quantity measures how different the two distributions are on average as measured by the statistic t. The benefit of mean discrepancy measures is that we can analytically maximise the discrepancy over a large class of statistics t, allowing us to find the statistic that most shows any discrepancy between the two distributions.

7.4 Kernel maximum mean discrepancy (MMD) two sample tests

Consider the two sample problem. We are given samples $X = (x_i)_{i=1\ldots m}$ and $Y = (y_i)_{i=1\ldots n}$ drawn i.i.d. from distributions p and q respectively. Can we determine if $p \neq q$?

An answer to this problem is to consider maximum mean discrepancy (MMD) (Gretton et al., 2008) statistics (also called integral probability metrics (Müller, 1997))

$$\text{MMD}(\mathcal{F}, p, q) = \sup_{f \in \mathcal{F}} (\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]) \tag{7.4.1}$$

where $\mathcal{F}$ is a set of functions. When $\mathcal{F}$ is a unit ball in a reproducing kernel Hilbert space (RKHS) the function attaining the supremum can be derived analytically and is called the witness function

$$f(x) = \mathbb{E}_{x'\sim p}[k(x, x')] - \mathbb{E}_{x'\sim q}[k(x, x')] \tag{7.4.2}$$

where k is the kernel of the RKHS.

Substituting this expression into equation (7.4.1) yields

$$\text{MMD}^2(\mathcal{F}, p, q) = \mathbb{E}_{x,x'\sim p}[k(x, x')] + 2\mathbb{E}_{x\sim p, y\sim q}[k(x, y)] + \mathbb{E}_{y,y'\sim q}[k(y, y')]. \tag{7.4.3}$$

This expression only involves expectations of the kernel k which can be estimated empirically by

$$\text{MMD}_b^2(\mathcal{F}, X, Y) = \frac{1}{m^2}\sum_{i,j=1}^{m} k(x_i, x_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(x_i, y_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(y_i, y_j). \tag{7.4.4}$$

One can also estimate the witness function from finite samples

$$\hat{f}(x) = \frac{1}{m}\sum_{i=1}^{m} k(x, x_i) - \frac{1}{n}\sum_{i=1}^{n} k(x, y_i) \tag{7.4.5}$$

i.e. the empirical witness function is the difference of two kernel density estimates (e.g. Parzen, 1962; Rosenblatt, 1956). This means that we can interpret the witness function as showing where the estimated densities of p and q are most different.

7.4.1 Kernel choice

The nature of the two sample test defined by the kernel MMD depends on the choice of the kernel. In this paper we use the radial basis function kernel, also known as the squared exponential or exponentiated quadratic. This kernel encodes for smooth functions characterised by a typical lengthscale (e.g. Rasmussen and Williams, 2006). A typical heuristic for selecting the lengthscale is to use the median distance between all points as the lengthscale (e.g. Gretton et al., 2008). However, since we interpret the witness function as the difference of two kernel density estimates we also consider selecting the lengthscale which gives the best density estimates (see section 7.5.2).

7.4.2 Estimation of the null distribution

There are a number of different ways in which the null distribution of the MMD statistic (7.4.4) can be estimated (e.g. Gretton et al., 2008). We use the bootstrap variant for its simplicity and general applicability. Note that the MMD two sample test using the bootstrap estimation of the null distribution is an example of a permutation test, also known as exact tests.

7.5 Examples on toy data

7.5.1 Newcomb's speed of light data

A histogram of Simon Newcomb's 66 measurements used to determine the speed of light (Stigler, 1977) is shown on the left of figure 7.1. We consider fitting a normal distribution to this data by maximum likelihood.

To perform a MMD two sample test we sampled 1000 points from the fitted distribution, used the median heuristic to select a lengthscale and estimated the null distribution

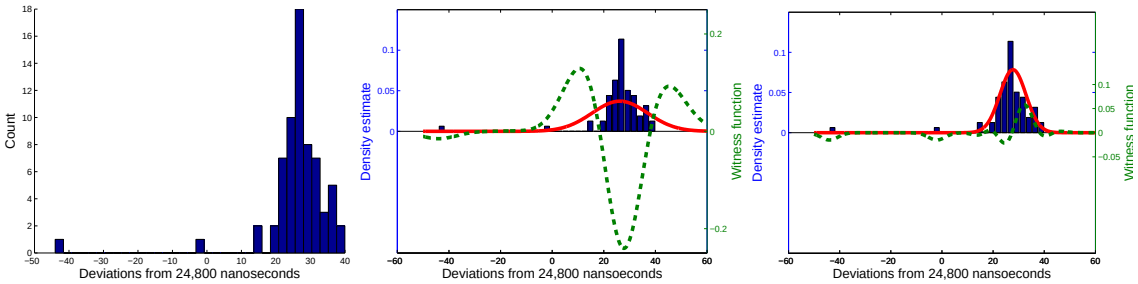


Figure 7.1: Left: Histogram of Simon Newcomb's speed of light measurements. Middle: Histogram together with density estimate (red solid line) and MMD witness function (green dashed line). Right: Histogram together with improved density estimate and witness function.

using 1000 bootstrap replications. The estimated p-value of the test was less than 0.001 i.e. a clear disparity between the model and data. The data, fitted density estimate (the normal distribution) and witness function are shown in the middle of figure 7.1. The witness function has a trough at the centre of the data and peaks either side. This indicates that the fitted model has placed too little mass in its centre and too much mass outside its centre.

This suggests that we should modify our model by either using a distribution with heavy tails or explicitly modelling the possibility of outliers which could have resulted in the variance being over-estimated. However, to demonstrate some of the properties of the MMD statistic we make an unusual choice of fitting a Gaussian by maximum likelihood, but ignoring the two outliers in the data. The new fitted density estimate (the normal distribution) and witness function of an MMD test are shown on the right of figure 7.1. The estimated p-value associated with the MMD two sample test is roughly 0.5, despite the fitted model being a very poor explanation of the outliers. This demonstrates that the MMD test using a radial basis function kernel identifies dense discrepancies, rather than outliers. However, methods that are not robust to outliers (e.g. fitting a Gaussian by maximum likelihood) will likely show dense discrepancies that will be identified by the test.

7.5.2 High dimensional data

The interpretability of the witness functions comes from being equal to the difference of two kernel density estimates (7.4.5). In high dimensional spaces, kernel density estimation is a very high variance procedure that can result in poor density estimates which will destroy the interpretability of the method. In response, we consider using

dimensionality reduction techniques before performing two sample tests.

To test how the MMD statistic can be used for high dimensional data we generated synthetic data using the following recipe. 5 points in a 10 dimensional space were drawn at random from a random 4 dimensional subspace[2]. Data was generated as isotropic Gaussian distributions centred on 4 of the 5 points. Finally, data was centred on the fifth point drawn from an isotropic t-distribution with 2 degrees of freedom. In sum, the data is a mixture of Gaussians and a t-distribution.

We then fit a mixture of Gaussians (e.g. McLachlan and Peel, 2004) with 5 centres to the data and then generated samples from the fitted distribution in order to perform an MMD two sample test. We reduced the dimensionality of the data using principal component analysis (PCA), selecting the first two principal components. To ensure that the MMD test remains well calibrated we include the PCA dimensionality reduction within the bootstrap estimation of the null distribution. The data and posterior predictive samples are plotted on the left of figure 7.2. While we can see that one cluster is different from the rest, it is difficult to assess by eye if these distributions are different — due in part to the difficulty of plotting two sets of samples on top of each other.

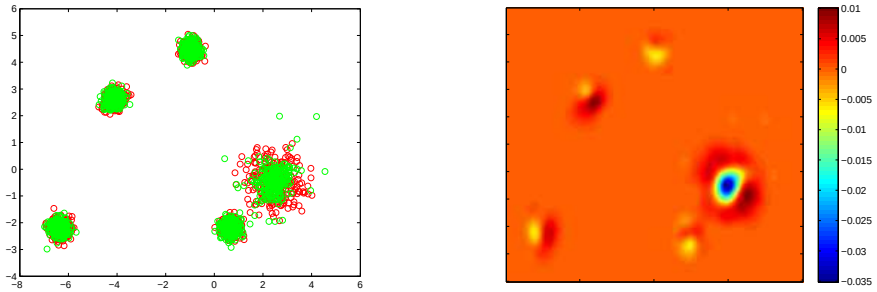


Figure 7.2: Left: PCA projection of synthetic high dimensional cluster data (green circles) and projection of samples from fitted model (red circles). Right: Witness function of MMD two sample test. The erroneously fit cluster is clearly identified.

Using the median heuristic to select a lengthscale results in a test that returns a p-value of 0.91, indicating that the test has not identified any discrepancies. Indeed, the lengthscale chosen by this heuristic is 4.4 which is of the order of the distances between clusters. The test therefore is blind to discrepancies smaller than distances between clusters[3], and since the mixture of Gaussians has correctly identified the 5 centres and sizes of the mixture distribution, the test does not find any discrepancies.

[2]The details are not especially important; code for replication is available at `https://github.com/jamesrobertlloyd/kmc-research`

[3]With enough data the test would eventually identify discrepancies on any scale — however, the

However, taking the density estimate interpretation of the witness function more seriously suggests choosing lengthscales that result in the best density estimates. We therefore selected a lengthscale by 5 fold cross validation using predictive likelihood of the kernel density estimate as the selection criterion. With this lengthscale the MMD test returns a p-value of 0.05 and the witness function (right of figure 7.2) clearly identifies the cluster that has been incorrectly modelled.

Presented with this discrepancy a statistical modeller might try a more flexible clustering model (e.g. Iwata et al., 2013; Peel and McLachlan, 2000) (a mixture of t-distributions would work on this example). However, the p-value of the MMD statistic can also be made non-significant by fitting a mixture of 10 Gaussians. We mention this as a reminder that the test proposed here does not attempt to falsify a class of models, it tests only whether or not the data could plausibly have been generated by a particular fitted model.

7.6 Applications to real data and complex statistical models

7.6.1 What exactly do neural networks dream about?

"To recognize shapes, first learn to generate images" quoth Hinton (Hinton, 2007). Restricted Boltzmann Machine (RBM) pretraining of neural networks was shown by Hinton et al. (2006) to learn a deep belief network (DBN) for the data i.e. a generative model. In agreement with this observation, as well as computing estimates of marginal likelihoods and testing errors, it has been standard to demonstrate the effectiveness of a neural network by generating samples from the distribution it has learned.

When trained on the MNIST handwritten digit data, samples from RBMs and DBNs certainly look like digits, but it is hard to detect any systematic anomalies purely by visual inspection. We now use the kernel MMD two-sample test to investigate how faithfully RBMs and DBNs can capture the distribution over handwritten digits.

RBMs mistake the identity of digits

We trained an RBM with architecture $(784) \leftrightarrow (500) \leftrightarrow (10)^4$ using 15 epochs of persistent contrastive divergence PCD-15, a batch size of 20 and a learning rate of 0.1 (i.e.

required amount of data can easily be very large

we used the same settings as the code available at the deep learning tutorial (Deep learning tutorial authors, 2014)). We generated 3000 independent samples from the learned generative model by initialising the network with a random training image and performing 1000 Gibbs updates with the digit labels clamped[5] to generate each image (as in e.g. Hinton (2007)).

The top left of figure 7.3 shows twenty random samples[6] from this model. They certainly look mostly like digits, but has the true distribution over digits been faithfully captured? A priori the answer to this question is almost certainly no, but it is not immediately obvious how the learned distribution will deviate from the true distribution.

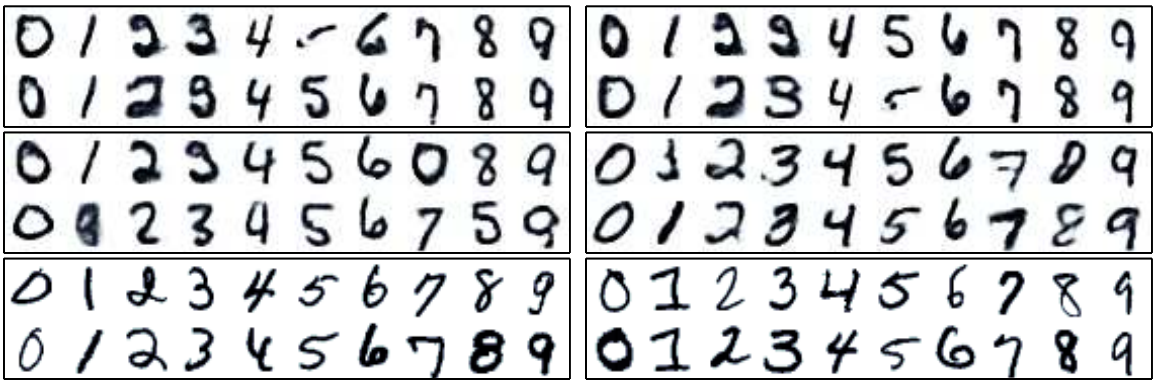


Figure 7.3: Top left: Random samples from an RBM. Top right: Troughs of the witness function for the RBM (digits that are over-represented by the model). Middle left: Troughs of the witness function for samples from 1500 RBMs. Middle right: Troughs of the witness function for the DBN. Bottom left: Peaks (digits that are under-represented by the model) of the witness function for samples from 1500 RBMs. Bottom right: Peaks of the witness function for the DBN.

Since we generated digits from the class conditional distributions we compare each class separately. However, for reference, the top plot of figure 7.4 displays the two dimensional PCA projection of MNIST digits (green circles) and samples from the RBM trained on this data (red crosses). It is entirely clear that the RBM has not faithfully captured the distribution of the data it was trained on. The situation is even worse when looking at the class conditional distributions of single digits. The corresponding scatter plot for the distribution over the digit zero is shown in the bottom left plot of

[4]That is, 784 input pixels and 10 indicators of the class label are connected to 500 hidden neurons.

[5]Without clamping the label neurons, the generative distribution is heavily biased towards certain digits.

[6]Specifically these are the activations of the pixel neurons before sampling sampling binary values. This is an attempt to be consistent with the grayscale input distribution of the images. Analogous discrepancies would be discovered if we had instead sampled binary pixel values.

figure 7.4. To the right of this figure is the corresponding witness function estimate. In this case it is easy to judge by eye where the discrepancies between model and data are most extreme without the witness function. The usefulness of the test comes into play for more borderline situations demonstrated later.

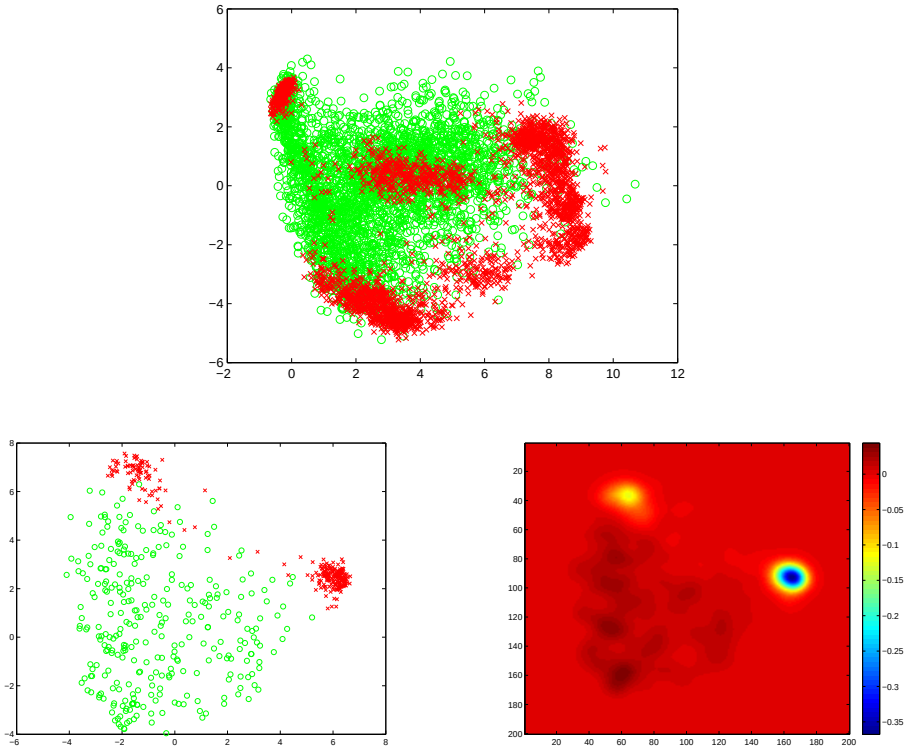


Figure 7.4: Top: PCA projection of MNIST digits (green circles) and sample from a trained RBM (red crosses). These are certainly not from the same distribution. Left: Same as top plot but applied to the zero digit class conditional distribution. Right: Estimated witness function for left plot.

Rather than show plots of the witness function for each digit we summarise the witness function by examples of digits closest to the peaks and troughs of the witness function (the witness function estimate is differentiable so we can find the peaks and troughs by gradient based optimisation). We apply the MMD two-sample test to each class conditional distribution, using PCA to reduce to 2 dimensions and selecting the lengthscale via cross validation as in section 7.5.2.

The top right box of figure 7.3 shows the digits closest to the two most extreme troughs[7] of the witness function for each class; the troughs indicate where the fitted

distribution over-represents the distribution of true digits. The estimated p-value for all tests was less than 0.001. The most obvious error with these digits is that the first 2 and 3 look quite similar.

To test that this was not just a poorly trained single RBM, we trained 1500 RBMs (with differently initialised pseudo random number generators) and generated one sample from each and performed the same tests. The estimated p-values were again all less than 0.001 and the summaries of the troughs of the witness function are shown in the middle left box of figure 7.3. On the first toy data example we observed that the MMD statistic does not highlight outliers and therefore we can conclude that RBMs are making consistent mistakes e.g. generating a 0 from the 7 distribution or a 5 when it should have been generating an 8.

DBNs have nightmares about ghosts

We now test the effectiveness of deep learning to represent the distribution of MNIST digits. In particular, we fit a DBN with architecture $(784) \leftarrow (500) \leftarrow (500) \leftrightarrow (2000) \leftrightarrow (10)$ using RBM pre-training and a generative fine tuning algorithm described in Hinton et al. (2006). Performing the same tests with 3000 samples results in estimated p-values of less than 0.001 except for the digit 4 (0.150) and digit 7 (0.010). Summaries of the witness function troughs are shown in the middle right box of figure 7.3.

The witness function no longer shows any class label mistakes (except perhaps for the digit 1 which looks very peculiar) but the 2, 3, 7 and 8 appear 'ghosted' — the digits fade in and out. For comparison the bottom right box of figure 7.3 shows digits closest to the peaks of the witness function; there is no trace of ghosting.

Returning to the RBMs, we do not see ghosting either, but the digits nearest the witness function troughs are somewhat blurred (see bottom left box for comparison with peaks). Assuming that the top level associative memory of the DBN also suffers from blurring, this will result in occasionally incorrect neurons in the second hidden layer on the DBN. These incorrect bits will then propagate down the DBN resulting in spurious features in several visible neurons, resulting in ghosting.

[7]The exact ordering of the peaks and troughs is as follows. We partition the space by grouping samples where the witness function has the same sign and gradient based optimisation of the witness function starting from each sample would reach the same peak or trough. The contribution to the MMD from each of these groups is used to order the peaks and troughs.

7.6.2 Testing non i.i.d. data

The test described so far applies when the model being tested has an i.i.d. predictive distribution. This is of course restrictive, so we now demonstrate how we can construct a test for a non i.i.d. model based on the MMD statistic. In particular we consider regression.

A test of local heteroscedasticity and non-normality for regression

We now assume that our data consists of pairs of inputs and outputs $(x_i^{\text{obs}}, y_i^{\text{obs}})_{i=1\ldots n}$. A typical formulation of the problem of regression is to estimate the conditional distribution of the outputs given the inputs $p(y \mid x, \theta)$. This is consistent with assuming that input-output pairs are generated i.i.d. from some distribution $p(y, x \mid \theta)$, but conditioned on observing the particular input values $(x_i^{\text{obs}})_{i=1\ldots n}$.

Following this observation, we might consider generating data from the plug-in conditional distribution $y_i^{\text{rep}} \sim p(y \mid x_i^{\text{obs}}, \hat{\theta})$ and computing the empirical MMD estimate (7.4.4) between $(x_i^{\text{obs}}, y_i^{\text{obs}})_{i=1\ldots n}$ and $(x_i^{\text{obs}}, y_i^{\text{rep}})_{i=1\ldots n}$. The only difference between this test and the MMD two sample test is that our data is generated from conditional distributions, rather than being i.i.d. . The null distribution of this statistic can be trivially estimated by sampling several sets of replicate data from the plug-in predictive distribution.

To demonstrate this test we apply it to 4 regression algorithms and 13 time series analysed in chapter 5. In this previous chapter we compared several methods for constructing Gaussian process regression models. For plots of the data sets see appendix B. While it is clear that simple smoothing methods will fail to capture all of the structure in the data, it is not clear a priori which extra patterns the more advanced methods will and will not be able to capture.

To construct p-values we use held out data using the same split of training and testing data as the interpolation experiment in chapter 5. Gaussian processes when applied to regression problems learn a joint distribution of all output values. However, this joint distribution information is rarely used; typically only the pointwise conditional distributions $p(y \mid x_i^{\text{obs}}, \hat{\theta})$ are used which is consistent with the test proposed here.

Table 7.1 shows a table of p-values for 13 data sets and 4 model construction methods. The four methods are Gaussian process regression using a squared exponential kernel (SE), trend-cyclical-irregular models (e.g. Lind et al., 2006) (TCI), spectral mixture kernels (Wilson and Adams, 2013) (SP) and the method proposed in chapter 5 (ABCD).

Values in bold indicate a positive discovery after a Benjamini–Hochberg (Benjamini and Hochberg) procedure with a false discovery rate of 0.05 applied to each model construction method. SE, TCI and SP have a very similar pattern of significant p-values whereas ABCD has fewer significant p-values.

Dataset	SE	TCI	SP	ABCD
Airline	0.36	**0.00**	0.07	0.15
Solar	**0.00**	**0.00**	**0.00**	0.05
Mauna	0.99	0.41	0.34	0.21
Wheat	**0.00**	**0.00**	**0.00**	0.19
Temperature	0.54	0.83	0.68	0.75
Internet	**0.00**	**0.01**	0.05	**0.01**
Call centre	**0.02**	**0.00**	**0.00**	0.07
Radio	**0.00**	**0.00**	**0.00**	**0.00**
Gas production	**0.00**	**0.01**	**0.01**	0.11
Sulphuric	0.29	0.38	0.34	0.52
Unemployment	**0.00**	**0.02**	**0.00**	**0.01**
Births	**0.00**	**0.02**	**0.00**	0.12
Wages	**0.00**	**0.01**	**0.01**	**0.00**

Table 7.1: Two sample test p-values applied to 13 time series and 4 regression algorithms. Bold values indicate a positive discovery using a Benjamini–Hochberg procedure with a false discovery rate of 0.05 for each model construction method.

We now investigate the type of discrepancies found by this test by looking at the witness function (which can still be interpreted as the difference of kernel density estimates). Figure 7.5 shows the solar and gas production data sets, the posterior distribution of the SE fits to this data and the witness functions for the SE fit. The solar witness function has a clear narrow peak, indicating that the data is more dense than expected by the fitted model in this region. We can see that this has identified a region of low variability in the data i.e. it has identified local heteroscedasticity not captured by the model. Similar conclusions can be drawn about the gas production data and witness function.

Of the four methods compared here, only ABCD is able to model heteroscedasticity, explaining why it is the only method with a substantially different set of significant p-values. However, the procedure is still potentially failing to capture structure on four of the datasets.

Figure 7.6 shows the unemployment and Internet data sets, the posterior distribution for the ABCD fits to the data and the witness functions of the ABCD fits. The

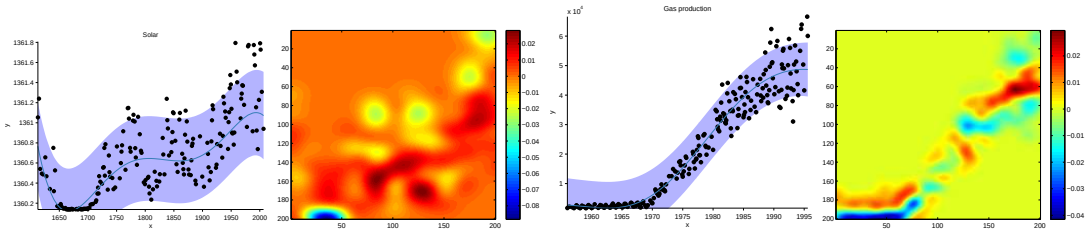


Figure 7.5: From left to right. Solar data with SE posterior. Witness function of SE fit to solar. Gas production data with SE posterior. Witness function of SE fit to gas production.

ABCD method has captured much of the structure in these data sets, making it difficult to visually identify discrepancies between model and data. The witness function for unemployment shows peaks and troughs at similar values of the input x. Comparing to the raw data we see that at these input values there are consistent outliers. Since ABCD is based on Gaussianity assumptions these consistent outliers have caused the method to estimate a large variance in this region, when the true data is non-Gaussian. There is also a similar pattern of peaks and troughs on the internet data suggesting that non-normality has again been detected. Indeed, the data appears to have a strict lower bound which is inconsistent with Gaussianity.

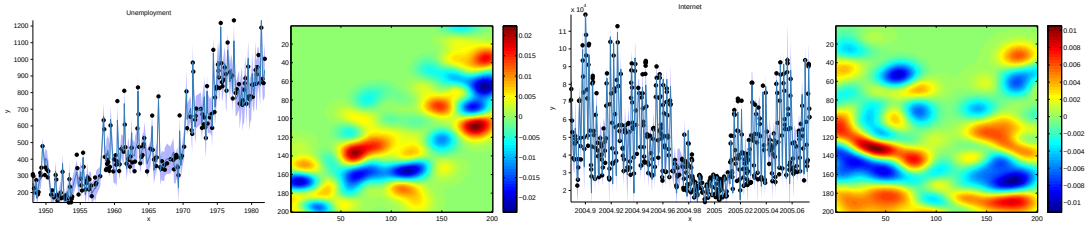


Figure 7.6: From left to right. Unemployment data with ABCD posterior. Witness function of ABCD fit to unemployment. Internet data with ABCD posterior. Witness function of ABCD fit to Internet.

7.7 Discussion of model criticism and related work

7.7.1 Are we criticising a particular model, or class of models?

In section 7.2 we interpreted the differences between classical, Bayesian prior/posterior and plug-in p-values as corresponding to different null hypotheses and interpretations of the word 'model'. In particular the classical p-value tests a null hypothesis that the

data could have been generated by a class of distributions (e.g. all normal distributions) whereas all other p-values test a particular probability distribution.

Robins et al. (2000) demonstrated that Bayesian and plug-in p-values are not classical p-values (frequentist p-values in their terminology) i.e. they do not have a uniform distribution under the relevant null hypothesis. However, this was presented as a failure of these methods; in particular they demonstrated that methods proposed by Bayarri and Berger (1999) based on posterior predictive p-values are asymptotically classical p-values.

This claimed inadequacy of posterior predictive p-values was rebutted (Gelman, 2003) and while their usefulness is becoming more accepted (see e.g. introduction of Bayarri and Castellanos (2007)) it would appear there is still confusion on the subject (Gelman, 2013). We hope that our interpretation of the differences between these methods as different null hypotheses — appropriate in different circumstances — sheds further light on the matter.

7.7.2 Should we worry about using the same data for training and criticism?

Plug-in and posterior predictive p-values test the null hypothesis that the observed data could have been generated by the fitted model or posterior predictive distribution. In some situations it may be more appropriate to attempt to falsify the null hypothesis that future data will be generated by the plug-in or posterior predictive distribution. As mentioned in section 7.2 this can be achieved by reserving a portion of the data to be used for model criticism alone, rather than fitting a model or updating a posterior on the full data. Cross validation methods have also been investigated in this context (Gelfand et al., 1992; Marshall and Spiegelhalter, 2007).

Other methods for evaluating statistical models Other typical methods of model evaluation include estimating the predictive performance of the model, analyses of sensitivities to modelling parameters / priors, graphical tests, and estimates of model utility. For a recent survey of Bayesian methods for model assessment, selection and comparison see Vehtari and Ojanen (2012) which phrases many techniques as estimates of the utility of a model. For some discussion of sensitivity analysis and graphical model comparison see e.g. Gelman et al. (2013).

In this manuscript we have focused on methods that compare statistics of data with

predictive distributions, ignoring parameters of the model. The discrepancy measures of Gelman et al. (1996) compute statistics of data and parameters; examples can be found in Gelman et al. (2013). O'Hagan (2003) also proposes a method and selectively reviews techniques for model criticism that also take model parameters into account.

In the spirit of scientific falsification (e.g. Popper, 2005), ideally all methods of assessing a model should be performed to gain confidence in any conclusions made. Of course, when performing multiple hypothesis tests care must be taken in the interpretation of individual p-values.

7.8 Conclusions

In this paper we have demonstrated an exploratory form of model criticism based on two sample tests using kernel maximum mean discrepancy. In contrast to other methods for model criticism, the test analytically maximises over a broad class of statistics, automatically identifying the statistic which most demonstrates the discrepancy between the model and data. We demonstrated how this method of model criticism can be applied to neural networks and Gaussian process regression and demonstrated the ways in which these models were misrepresenting the data they were trained on.

We have demonstrated how kernel MMD two sample tests can be applied to model criticism, but they can be applied to any aspect of statistical modelling where two sample tests are appropriate. This includes for example, Geweke's tests of Markov chain posterior sampler validity (Geweke, 2004) and tests of Markov chain convergence (e.g. Cowles and Carlin, 1996).

The two sample tests proposed in this paper naturally apply to i.i.d. data and models, but model criticism techniques should of course apply to models with other symmetries (e.g. exchangeable data, longitudinal data / time series, graphs, functions any many others). We have demonstrated an adaptation of the kernel MMD test to regression models However, it is unclear whether maximum mean discrepancy measures can be naturally extended to all of these classes; investigating such extensions would be a profitable area for future study (for very recent work in this direction see Chwialkowski et al. (2014)).

In proposing a new method of model criticism we hope we have also exposed the machine learning community to a useful set of tools for diagnosing potential inadequacies of models.

7.9 Discussion and future work

7.9.1 What happened to statistical power?

In this chapter we have discussed hypothesis tests and p-values but have not mentioned the statistical power of these tests i.e. the probability the the null hypothesis will be rejected by some procedure given that the null hypothesis is false. This requires us to specify an alternative to the null hypothesis but the prevailing view of model criticism is that it is an exploratory process that can be performed when no alternative models are currently being entertained. In particular, O'Hagan (2003) writes

> Model criticism ... is intended as an open-minded phase of investigation to identify any problems with the model. Formulation of explicit alternatives comes after the model criticism phase has identified some problems.

but also notes

> It is recognised that alternatives are at least loosely implicit in any choice of model criticism tools, but the role of model criticism is essentially exploratory.

It seems reasonable however to make efforts to characterise the class of distributions against which a particular test has high power to understand the types of discrepancy the test is likely to identify. For the MMD two sample test this distribution is characterised in section 5 of Gretton et al. (2012); the test has higher power for discrepancies whose mean embedding in the relevant RKHS has large norm. It would make profitable study to demonstrate how different choices of kernel can identify qualitatively different types of discrepancy.

7.9.2 Should we not just attempt to fit ever larger models?

If after completing a statistical analysis, one has the resources (intellectual, computational, etc.) to attempt to fit a larger, more complex or more realistic statistical model then one should do so. However, this process of model expansion will eventually become infeasible for all but the simplest of problems. At this point one might consider cheaper alternatives to performing probabilistic inference, such as model criticism techniques. Faced with a decision between multiple actions it seems likely that this problem should ultimately be phrased in decision theoretic terms. However, the problem of how

one should make optimal statistical decisions given computational and other time constraints is still relatively uncharted territory (Jordan, 2013).

Chapter 8

Conclusion

This thesis has made a broad range of contributions to the practice of probabilistic modelling. In chapter 2 we demonstrated how theorems based on symmetry arguments could be used to provide a unifying view of probabilistic models for network data and proposed a new nonparametric model inspired as a literal interpretation of the theorems. In chapter 3 we extended these theorems to arbitrary relational data, or any data that can be stored in a database. These results bring further unity and clarity to the ever growing literature of different probabilistic models for such data. They also highlight the current mismatch between theory and data with regards to sparsity of data.

In chapter 4 we demonstrated how a broad range of probabilistic models of functions could be expressed in a common modelling language of Gaussian process kernels. We demonstrated that this language could be effectively searched to automatically construct appropriate models for particular data and that these models were often highly interpretable. This claim of interpretability was more rigorously justified in chapter 5 where we demonstrated that the compositional structure of the language of kernels could be neatly mapped onto compositionally generated natural language descriptions of the models the kernels encode for.

In chapter 6 we demonstrated the effectiveness of Gaussian processes with compositionally constructed kernels in a data mining competition. We also demonstrated how the model construction procedure introduced in chapter 4 could be used to recreate some of the results of the competition whilst also revealing some limitations with the current approximate inference methods available for Gaussian processes. Finally, in chapter 7 we considered the problem of criticising a probabilistic model when we have reason to believe that we were either unable to faithfully express our prior information probabilistically or perform inference to a sufficient level of accuracy. In particular we introduced a

new method for model criticism that attempts to find the greatest discrepancies between a probabilistic model and data and thereby target future efforts to improve a particular model.

Appendix A

Example of an automatically generated model and description

The following pages include an automatically generated report prodcued by the procedure described in chapter 5. More examples are available at `http://mlg.eng.cam.ac.uk/lloyd/abcdoutput/`.

1 Executive summary

The raw data and full model posterior with extrapolations are shown in figure 1.

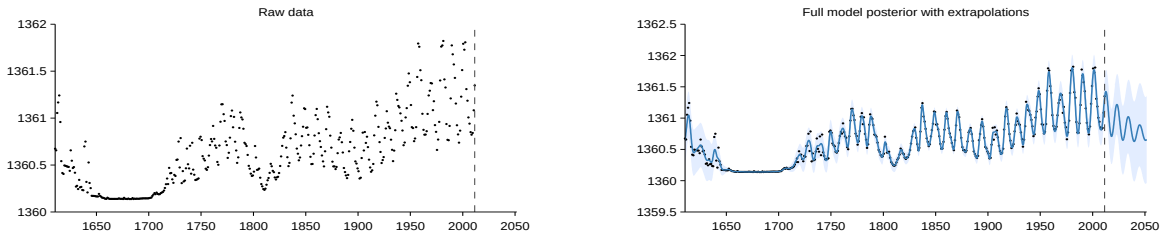


Figure 1: Raw data (left) and model posterior with extrapolation (right)

The structure search algorithm has identified eight additive components in the data. The first 4 additive components explain 92.3% of the variation in the data as shown by the coefficient of determination (R^2) values in table 1. The first 6 additive components explain 99.7% of the variation in the data. After the first 5 components the cross validated mean absolute error (MAE) does not decrease by more than 0.1%. This suggests that subsequent terms are modelling very short term trends, uncorrelated noise or are artefacts of the model or search procedure. Short summaries of the additive components are as follows:

- A constant.
- A constant. This function applies from 1643 until 1716.
- A smooth function. This function applies until 1643 and from 1716 onwards.
- An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards.
- A rapidly varying smooth function. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise with standard deviation increasing linearly away from 1837. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise with standard deviation increasing linearly away from 1952. This function applies until 1643 and from 1716 onwards.
- Uncorrelated noise. This function applies from 1643 until 1716.

#	R^2 (%)	ΔR^2 (%)	Residual R^2 (%)	Cross validated MAE	Reduction in MAE (%)
-	-	-	-	1360.65	-
1	0.0	0.0	0.0	0.33	100.0
2	37.4	37.4	37.4	0.23	32.0
3	72.8	35.4	56.6	0.18	21.1
4	92.3	19.4	71.5	0.15	16.8
5	98.1	5.9	75.9	0.15	0.4
6	99.7	1.6	85.6	0.15	0.0
7	100.0	0.3	99.8	0.15	0.0
8	100.0	0.0	100.0	0.15	0.0

Table 1: Summary statistics for cumulative additive fits to the data. The residual coefficient of determination (R^2) values are computed using the residuals from the previous fit as the target values; this measures how much of the residual variance is explained by each new component. The mean absolute error (MAE) is calculated using 10 fold cross validation with a contiguous block design; this measures the ability of the model to interpolate and extrapolate over moderate distances. The model is fit using the full data and the MAE values are calculated using this model; this double use of data means that the MAE values cannot be used reliably as an estimate of out-of-sample predictive performance.

Model checking statistics are summarised in table 2 in section 4. These statistics have revealed statistically significant discrepancies between the data and model in component 8.

The rest of the document is structured as follows. In section 2 the forms of the additive components are described and their posterior distributions are displayed. In section 3 the modelling assumptions of each component are discussed with reference to how this affects the extrapolations made by the model. Section 4 discusses model checking statistics, with plots showing the form of any detected discrepancies between the model and observed data.

2 Detailed discussion of additive components

2.1 Component 1 : A constant

This component is constant.

This component explains 0.0% of the total variance. The addition of this component reduces the cross validated MAE by 100.0% from 1360.6 to 0.3.

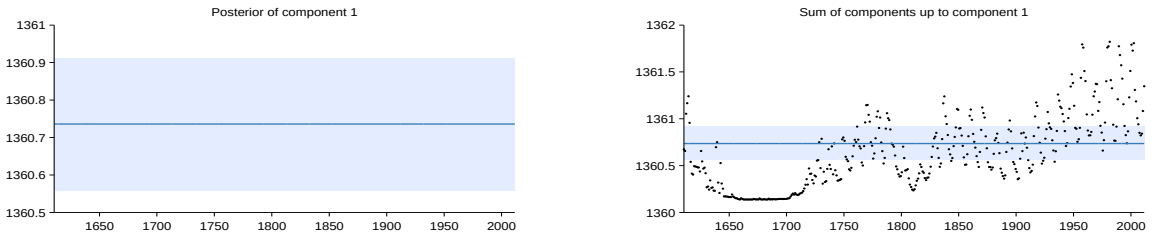


Figure 2: Pointwise posterior of component 1 (left) and the posterior of the cumulative sum of components with data (right)

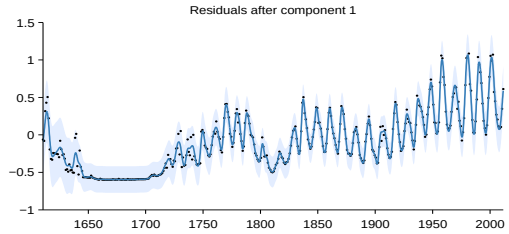


Figure 3: Pointwise posterior of residuals after adding component 1

2.2 Component 2 : A constant. This function applies from 1643 until 1716

This component is constant. This component applies from 1643 until 1716.

This component explains 37.4% of the residual variance; this increases the total variance explained from 0.0% to 37.4%. The addition of this component reduces the cross validated MAE by 31.97% from 0.33 to 0.23.

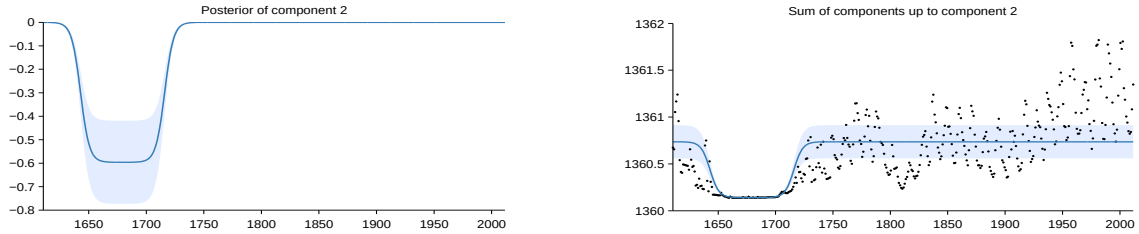


Figure 4: Pointwise posterior of component 2 (left) and the posterior of the cumulative sum of components with data (right)

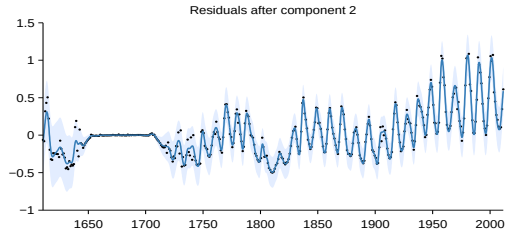


Figure 5: Pointwise posterior of residuals after adding component 2

2.3 Component 3 : A smooth function. This function applies until 1643 and from 1716 onwards

This component is a smooth function with a typical lengthscale of 23.1 years. This component applies until 1643 and from 1716 onwards.

This component explains 56.6% of the residual variance; this increases the total variance explained from 37.4% to 72.8%. The addition of this component reduces the cross validated MAE by 21.08% from 0.23 to 0.18.

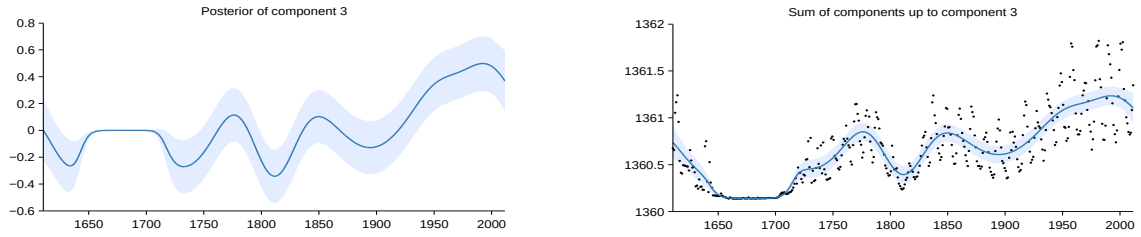


Figure 6: Pointwise posterior of component 3 (left) and the posterior of the cumulative sum of components with data (right)

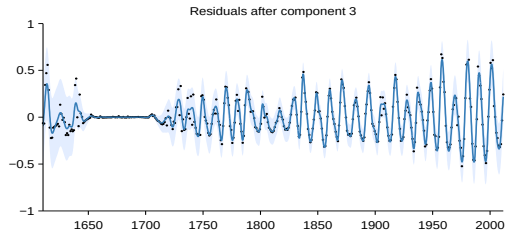


Figure 7: Pointwise posterior of residuals after adding component 3

2.4 Component 4 : An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards

This component is approximately periodic with a period of 10.8 years. Across periods the shape of this function varies smoothly with a typical lengthscale of 36.9 years. The shape of this function within each period is very smooth and resembles a sinusoid. This component applies until 1643 and from 1716 onwards.

This component explains 71.5% of the residual variance; this increases the total variance explained from 72.8% to 92.3%. The addition of this component reduces the cross validated MAE by 16.82% from 0.18 to 0.15.

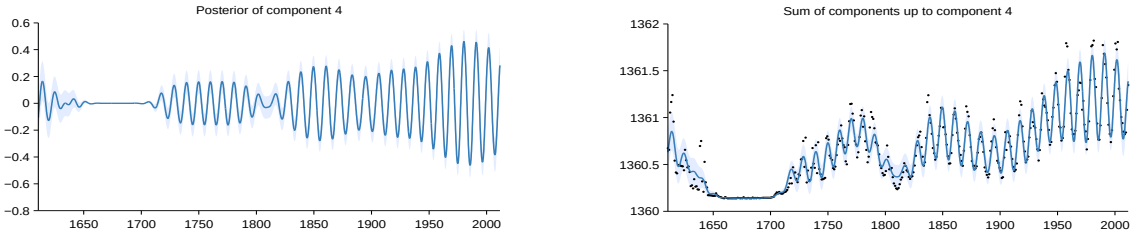


Figure 8: Pointwise posterior of component 4 (left) and the posterior of the cumulative sum of components with data (right)

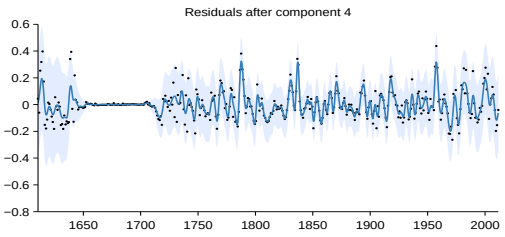


Figure 9: Pointwise posterior of residuals after adding component 4

2.5 Component 5 : A rapidly varying smooth function. This function applies until 1643 and from 1716 onwards

This component is a rapidly varying but smooth function with a typical lengthscale of 1.6 years. This component applies until 1643 and from 1716 onwards.

This component explains 75.9% of the residual variance; this increases the total variance explained from 92.3% to 98.1%. The addition of this component reduces the cross validated MAE by 0.35% from 0.15 to 0.15.

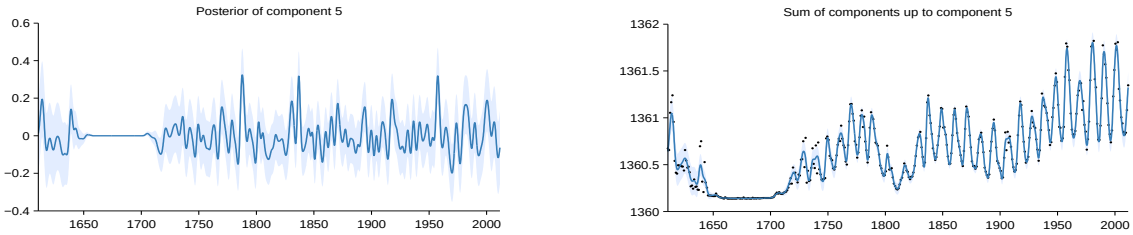


Figure 10: Pointwise posterior of component 5 (left) and the posterior of the cumulative sum of components with data (right)

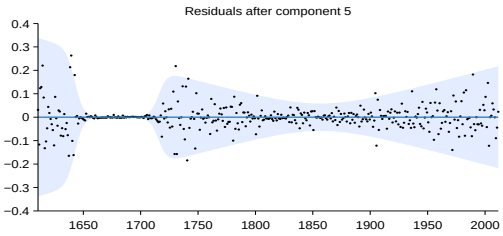


Figure 11: Pointwise posterior of residuals after adding component 5

2.6 Component 6 : Uncorrelated noise with standard deviation increasing linearly away from 1837. This function applies until 1643 and from 1716 onwards

This component models uncorrelated noise. The standard deviation of the noise increases linearly away from 1837. This component applies until 1643 and from 1716 onwards.

This component explains 85.6% of the residual variance; this increases the total variance explained from 98.1% to 99.7%. The addition of this component reduces the cross validated MAE by 0.00% from 0.15 to 0.15. This component explains residual variance but does not improve MAE which suggests that this component describes very short term patterns, uncorrelated noise or is an artefact of the model or search procedure.

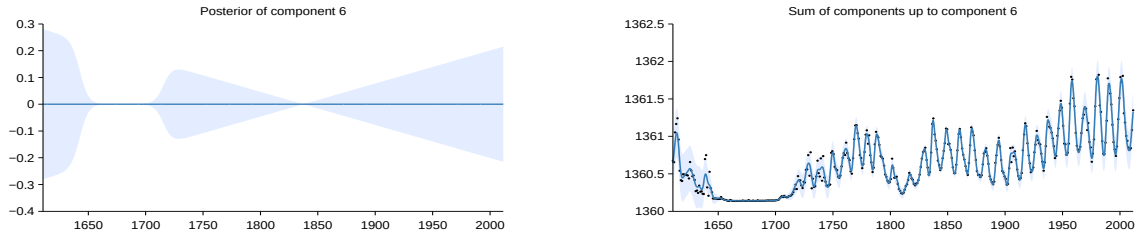


Figure 12: Pointwise posterior of component 6 (left) and the posterior of the cumulative sum of components with data (right)

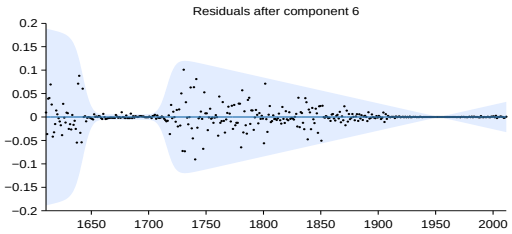


Figure 13: Pointwise posterior of residuals after adding component 6

2.7 Component 7 : Uncorrelated noise with standard deviation increasing linearly away from 1952. This function applies until 1643 and from 1716 onwards

This component models uncorrelated noise. The standard deviation of the noise increases linearly away from 1952. This component applies until 1643 and from 1716 onwards.

This component explains 99.8% of the residual variance; this increases the total variance explained from 99.7% to 100.0%. The addition of this component reduces the cross validated MAE by 0.00% from 0.15 to 0.15. This component explains residual variance but does not improve MAE which suggests that this component describes very short term patterns, uncorrelated noise or is an artefact of the model or search procedure.

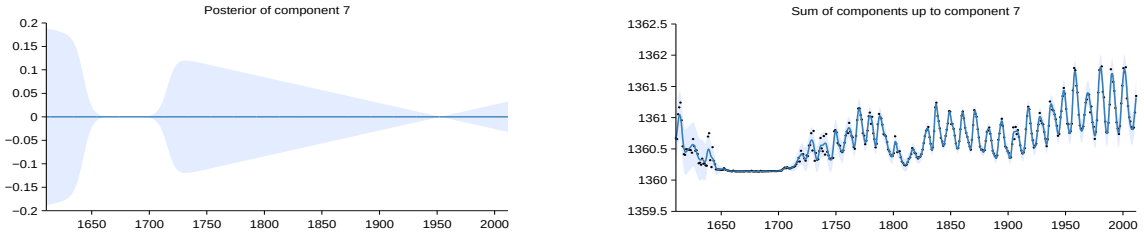


Figure 14: Pointwise posterior of component 7 (left) and the posterior of the cumulative sum of components with data (right)

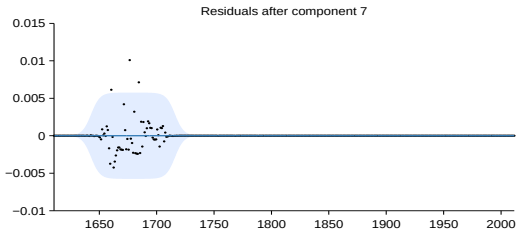


Figure 15: Pointwise posterior of residuals after adding component 7

2.8 Component 8 : Uncorrelated noise. This function applies from 1643 until 1716

This component models uncorrelated noise. This component applies from 1643 until 1716.

This component explains 100.0% of the residual variance; this increases the total variance explained from 100.0% to 100.0%. The addition of this component reduces the cross validated MAE by 0.00% from 0.15 to 0.15. This component explains residual variance but does not improve MAE which suggests that this component describes very short term patterns, uncorrelated noise or is an artefact of the model or search procedure.

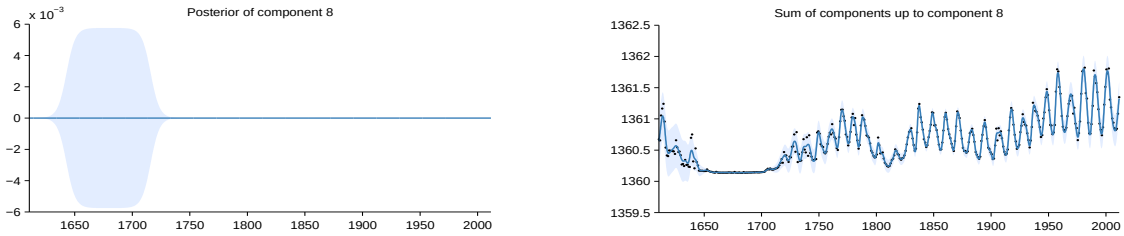


Figure 16: Pointwise posterior of component 8 (left) and the posterior of the cumulative sum of components with data (right)

3 Extrapolation

Summaries of the posterior distribution of the full model are shown in figure 17. The plot on the left displays the mean of the posterior together with pointwise variance. The plot on the right displays three random samples from the posterior.

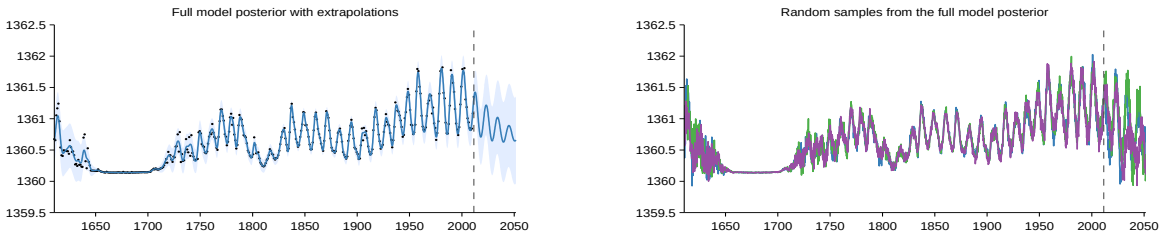


Figure 17: Full model posterior with extrapolation. Mean and pointwise variance (left) and three random samples (right)

Below are descriptions of the modelling assumptions associated with each additive component and how they affect the predictive posterior. Plots of the pointwise posterior and samples from the posterior are also presented, showing extrapolations from each component and the cuulative sum of components.

3.1 Component 1 : A constant

This component is assumed to stay constant.

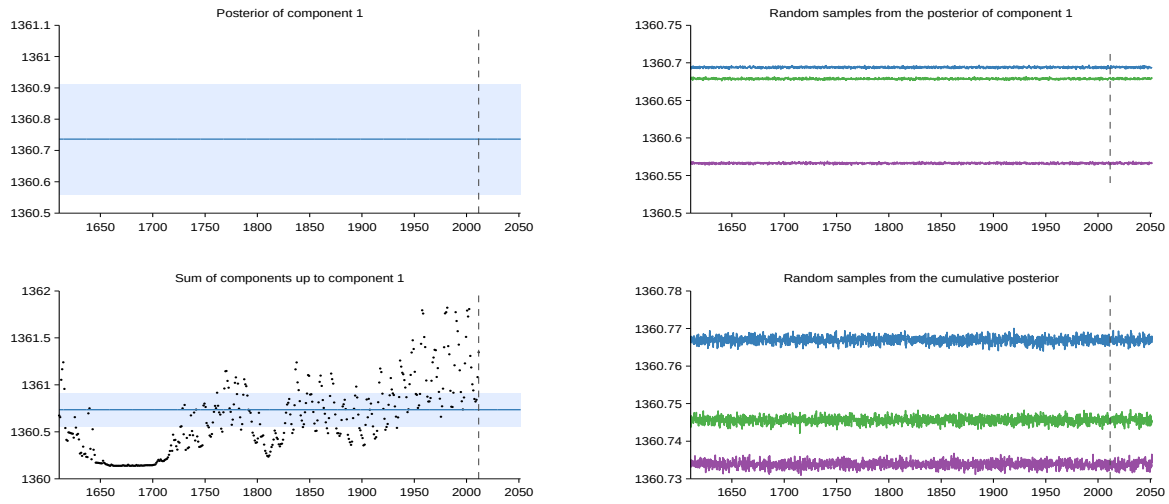


Figure 18: Posterior of component 1 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.2 Component 2 : A constant. This function applies from 1643 until 1716

This component is assumed to stop before the end of the data and will therefore be extrapolated as zero.

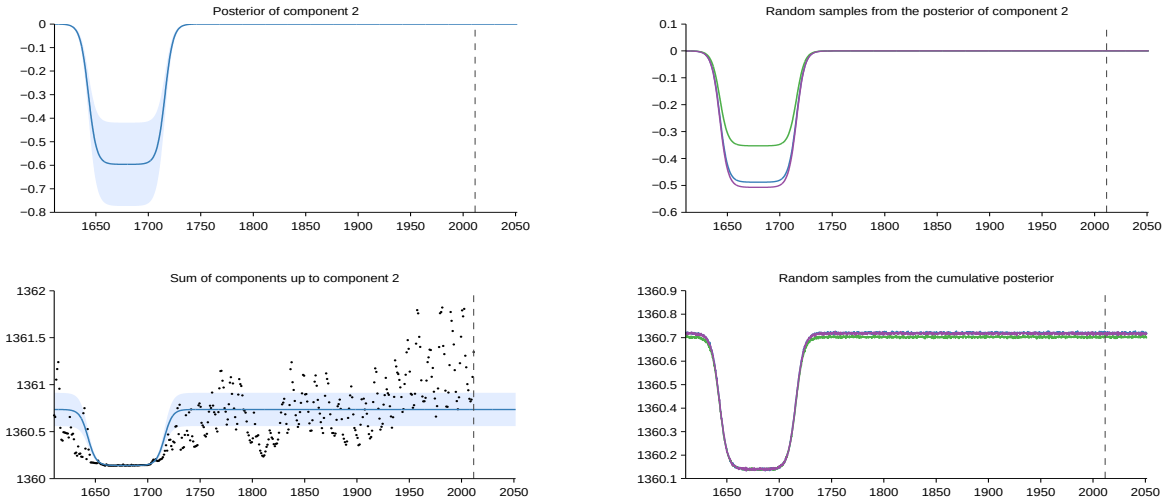


Figure 19: Posterior of component 2 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.3 Component 3 : A smooth function. This function applies until 1643 and from 1716 onwards

This component is assumed to continue smoothly but is also assumed to be stationary so its distribution will return to the prior. The prior distribution places mass on smooth functions with a marginal mean of zero and a typical lengthscale of 23.1 years. [This is a placeholder for a description of how quickly the posterior will start to resemble the prior].

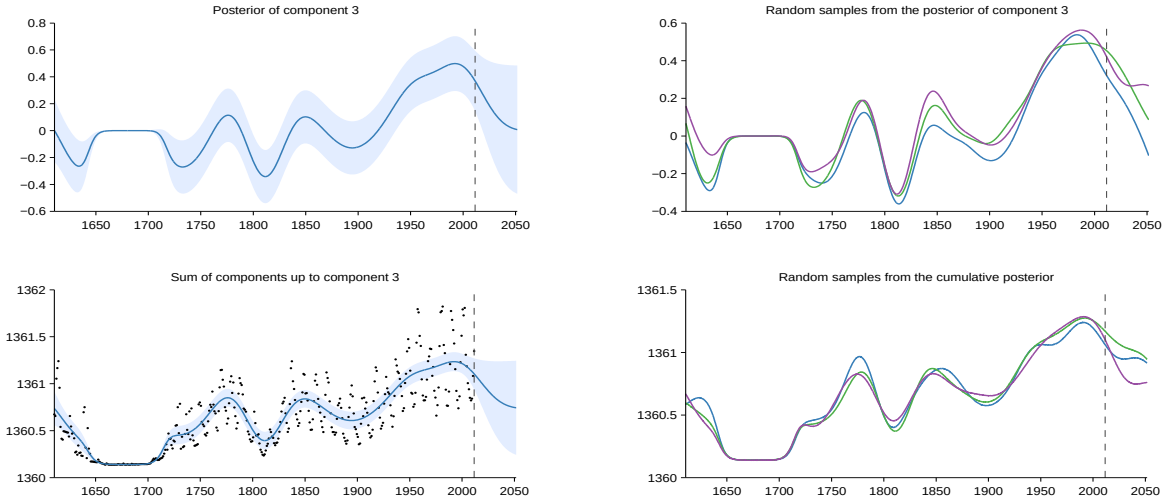


Figure 20: Posterior of component 3 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.4 Component 4 : An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards

This component is assumed to continue to be approximately periodic. The shape of the function is assumed to vary smoothly between periods but will return to the prior. The prior is entirely uncertain about the phase of the periodic function. Consequently the pointwise posterior will appear to lose its periodicity, but this merely reflects the uncertainty in the shape and phase of the function. [This is a placeholder for a description of how quickly the posterior will start to resemble the prior].

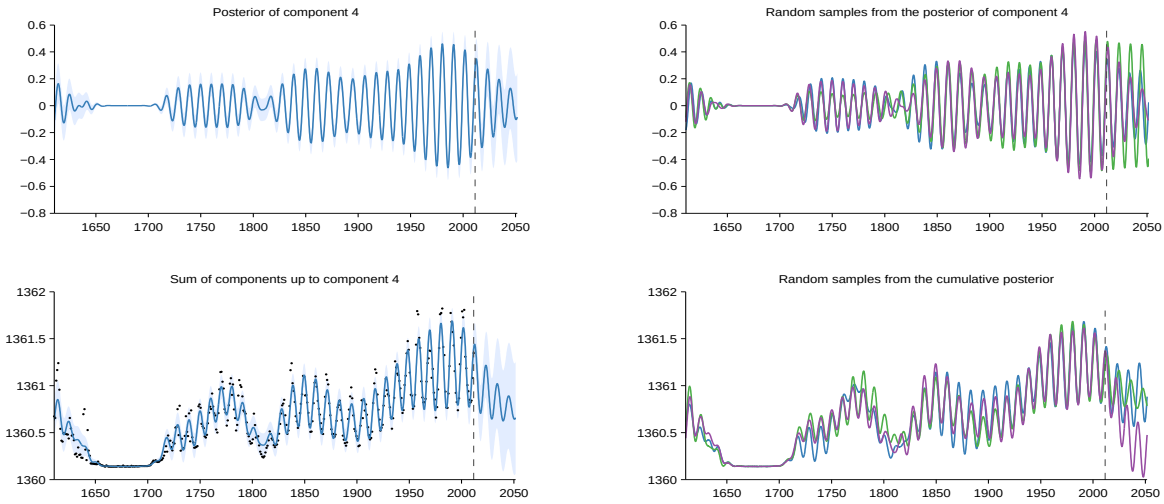


Figure 21: Posterior of component 4 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.5 Component 5 : A rapidly varying smooth function. This function applies until 1643 and from 1716 onwards

This component is assumed to continue smoothly but its distribution is assumed to quickly return to the prior. The prior distribution places mass on smooth functions with a marginal mean of zero and a typical lengthscale of 1.6 years. [This is a placeholder for a description of how quickly the posterior will start to resemble the prior].

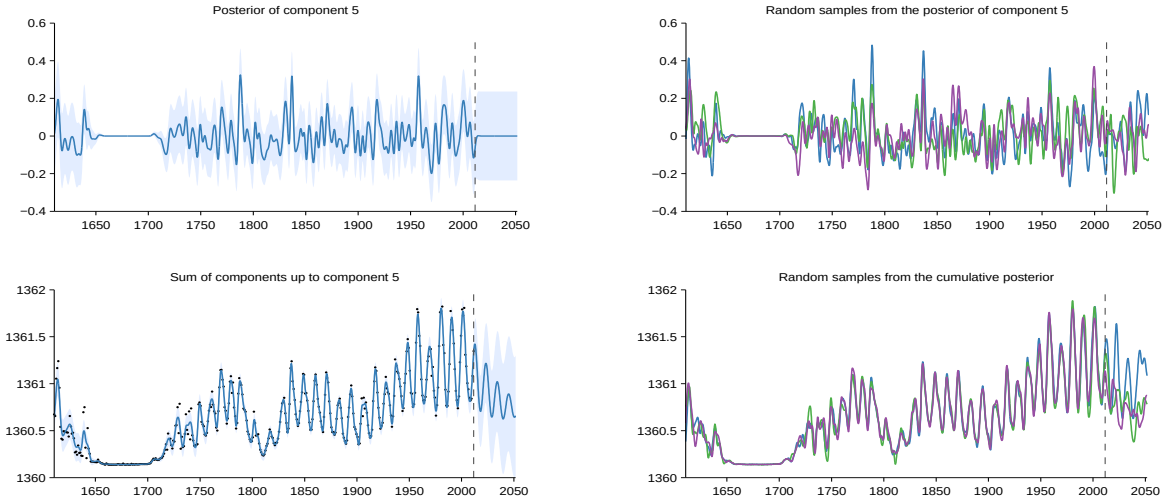


Figure 22: Posterior of component 5 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.6 Component 6 : Uncorrelated noise with standard deviation increasing linearly away from 1837. This function applies until 1643 and from 1716 onwards

This component assumes the uncorrelated noise will continue indefinitely. The standard deviation of the noise is assumed to continue to increase linearly.

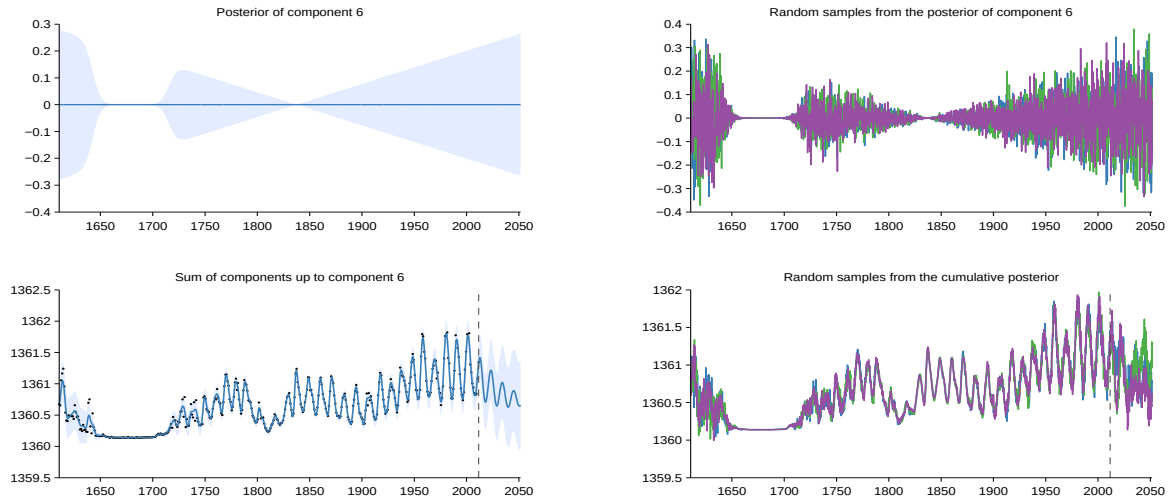


Figure 23: Posterior of component 6 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.7 Component 7 : Uncorrelated noise with standard deviation increasing linearly away from 1952. This function applies until 1643 and from 1716 onwards

This component assumes the uncorrelated noise will continue indefinitely. The standard deviation of the noise is assumed to continue to increase linearly.

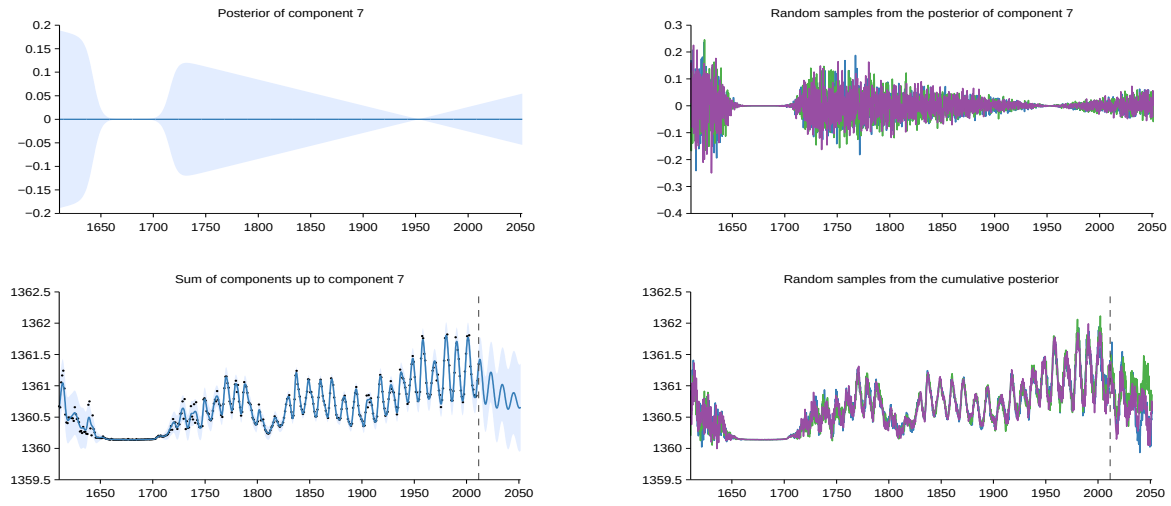


Figure 24: Posterior of component 7 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

3.8 Component 8 : Uncorrelated noise. This function applies from 1643 until 1716

This component is assumed to stop before the end of the data and will therefore be extrapolated as zero.

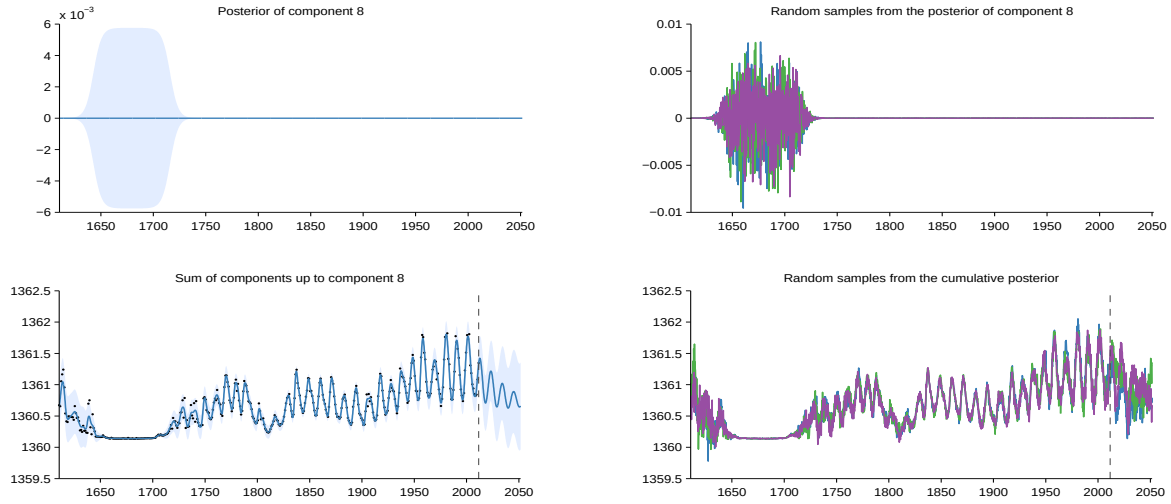


Figure 25: Posterior of component 8 (top) and cumulative sum of components (bottom) with extrapolation. Mean and pointwise variance (left) and three random samples from the posterior distribution (right).

4 Model checking

Several posterior predictive checks have been performed to assess how well the model describes the observed data. These tests take the form of comparing statistics evaluated on samples from the prior and posterior distributions for each additive component. The statistics are derived from autocorrelation function (ACF) estimates, periodograms and quantile-quantile (qq) plots.

Table 2 displays cumulative probability and p-value estimates for these quantities. Cumulative probabilities near 0/1 indicate that the test statistic was lower/higher under the posterior compared to the prior unexpectedly often i.e. they contain the same information as a p-value for a two-tailed test and they also express if the test statistic was higher or lower than expected. p-values near 0 indicate that the test statistic was larger in magnitude under the posterior compared to the prior unexpectedly often.

	ACF		Periodogram		QQ	
#	min	min loc	max	max loc	max	min
1	0.501	0.481	0.543	0.497	0.223	0.776
2	0.501	0.479	0.723	0.500	0.858	0.192
3	0.959	0.898	0.734	0.229	0.368	0.792
4	0.564	0.486	0.393	0.371	0.790	0.812
5	0.605	0.465	0.409	0.455	0.204	0.732
6	0.516	0.477	0.412	0.396	0.477	0.674
7	0.456	0.510	0.461	0.480	0.498	0.561
8	0.584	0.638	0.585	0.526	0.012	0.697

Table 2: Model checking statistics for each component. Cumulative probabilities for minimum of autocorrelation function (ACF) and its location. Cumulative probabilities for maximum of periodogram and its location. p-values for maximum and minimum deviations of QQ-plot from straight line.

The nature of any observed discrepancies is now described and plotted and hypotheses are given for the patterns in the data that may not be captured by the model.

4.1 Moderately statistically significant discrepancies

4.1.1 Component 8 : Uncorrelated noise. This function applies from 1643 until 1716

The following discrepancies between the prior and posterior distributions for this component have been detected.

- The qq plot has an unexpectedly large positive deviation from equality ($x = y$). This discrepancy has an estimated p-value of 0.012.

The positive deviation in the qq-plot can indicate heavy positive tails if it occurs at the right of the plot or light negative tails if it occurs as the left.

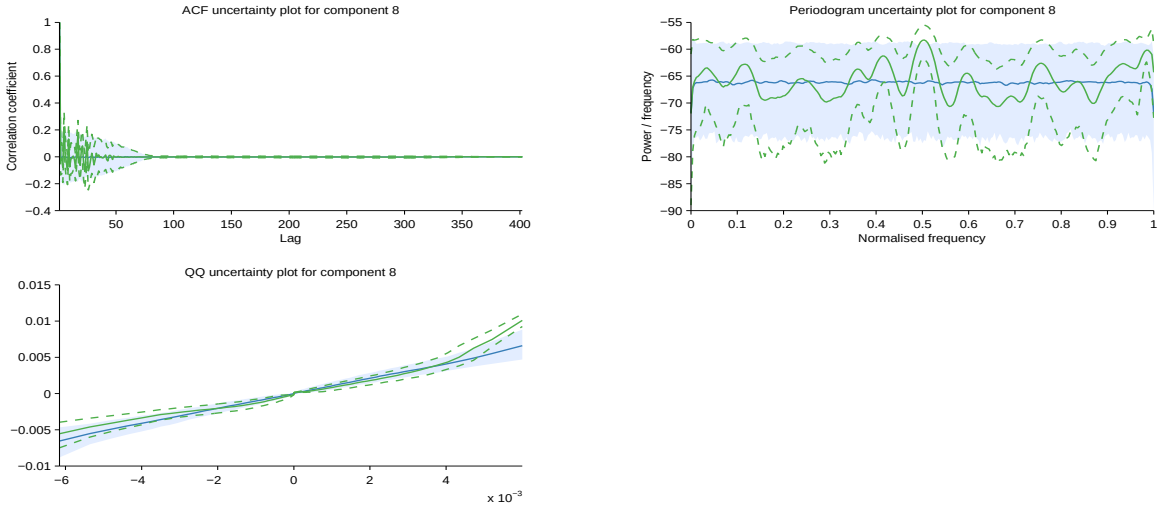


Figure 26: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 8. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2 Model checking plots for components without statistically significant discrepancies

4.2.1 Component 1 : A constant

No discrepancies between the prior and posterior of this component have been detected

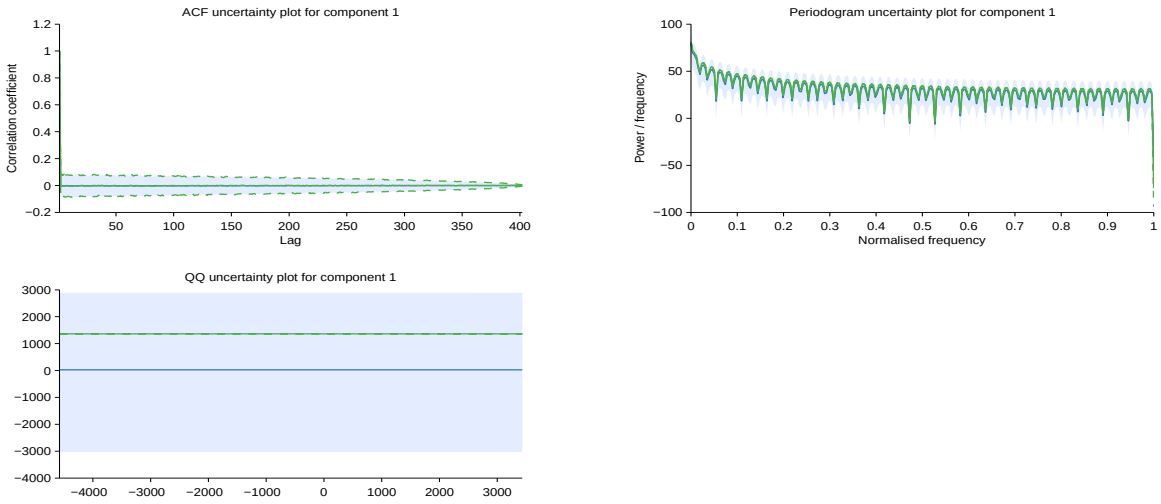


Figure 27: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 1. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2.2 Component 2 : A constant. This function applies from 1643 until 1716

No discrepancies between the prior and posterior of this component have been detected

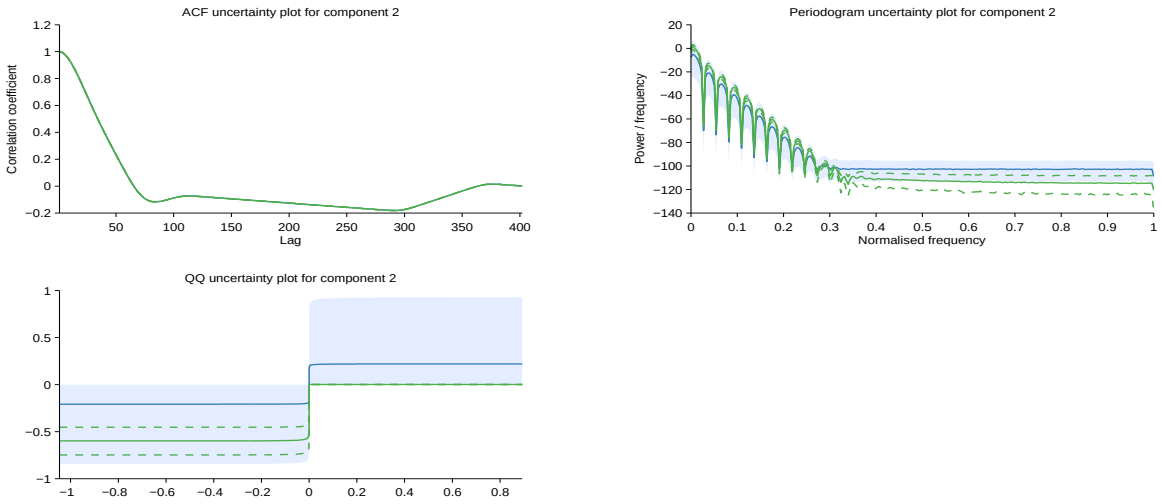


Figure 28: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 2. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2.3 Component 3 : A smooth function. This function applies until 1643 and from 1716 onwards

No discrepancies between the prior and posterior of this component have been detected

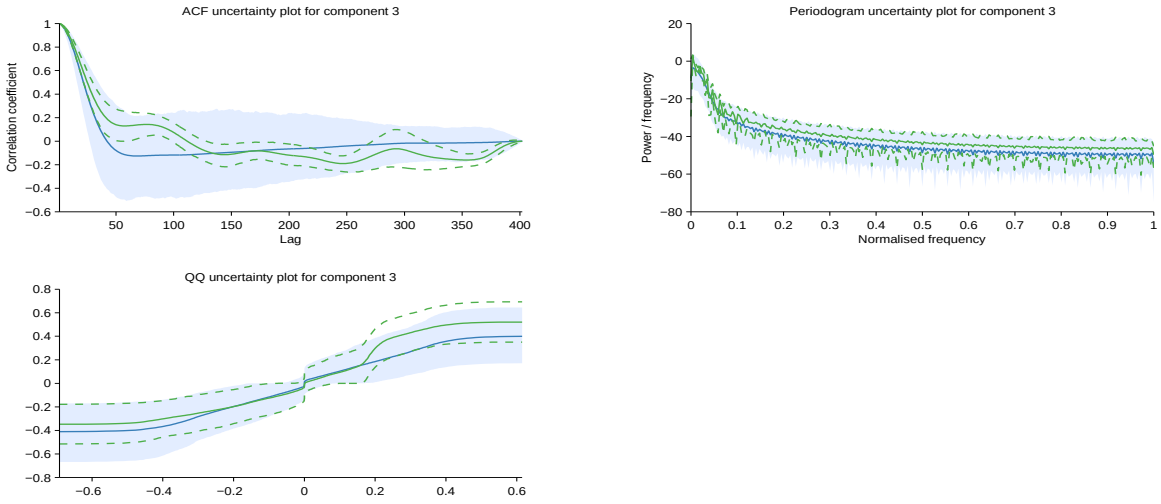


Figure 29: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 3. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2.4 Component 4 : An approximately periodic function with a period of 10.8 years. This function applies until 1643 and from 1716 onwards

No discrepancies between the prior and posterior of this component have been detected

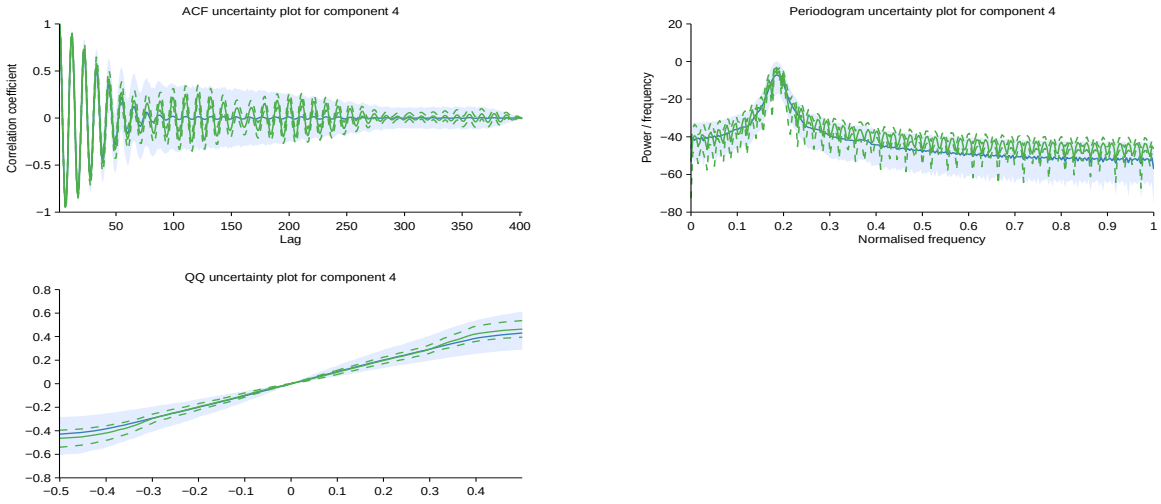


Figure 30: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 4. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2.5 Component 5 : A rapidly varying smooth function. This function applies until 1643 and from 1716 onwards

No discrepancies between the prior and posterior of this component have been detected

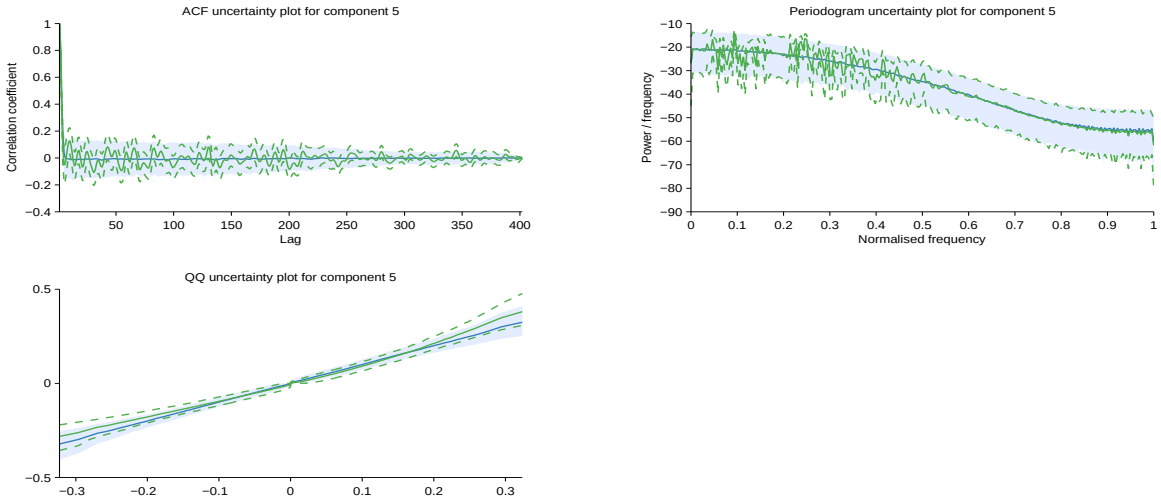


Figure 31: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 5. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2.6 Component 6 : Uncorrelated noise with standard deviation increasing linearly away from 1837. This function applies until 1643 and from 1716 onwards

No discrepancies between the prior and posterior of this component have been detected

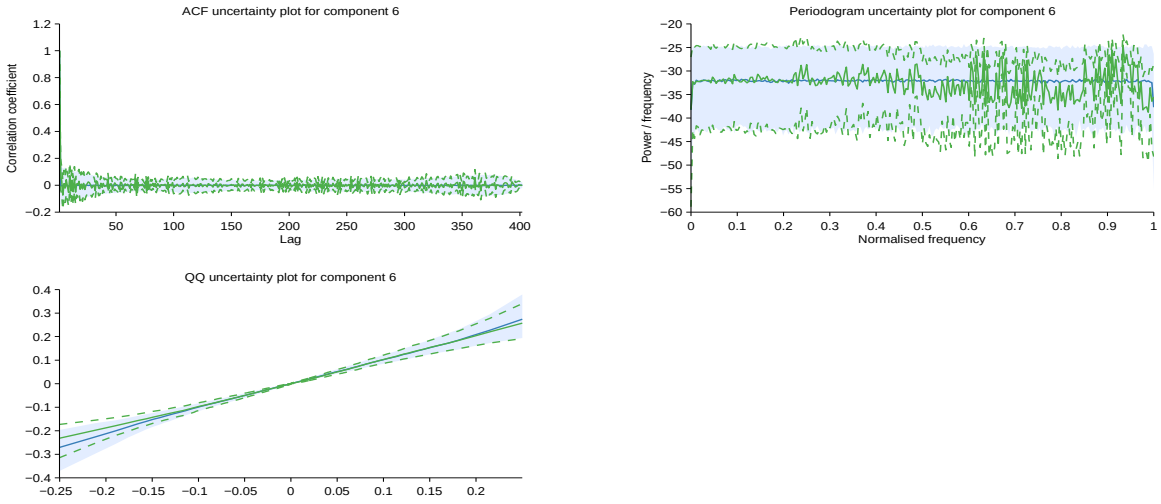


Figure 32: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 6. The green line and green dashed lines are the corresponding quantities under the posterior.

4.2.7 Component 7 : Uncorrelated noise with standard deviation increasing linearly away from 1952. This function applies until 1643 and from 1716 onwards

No discrepancies between the prior and posterior of this component have been detected

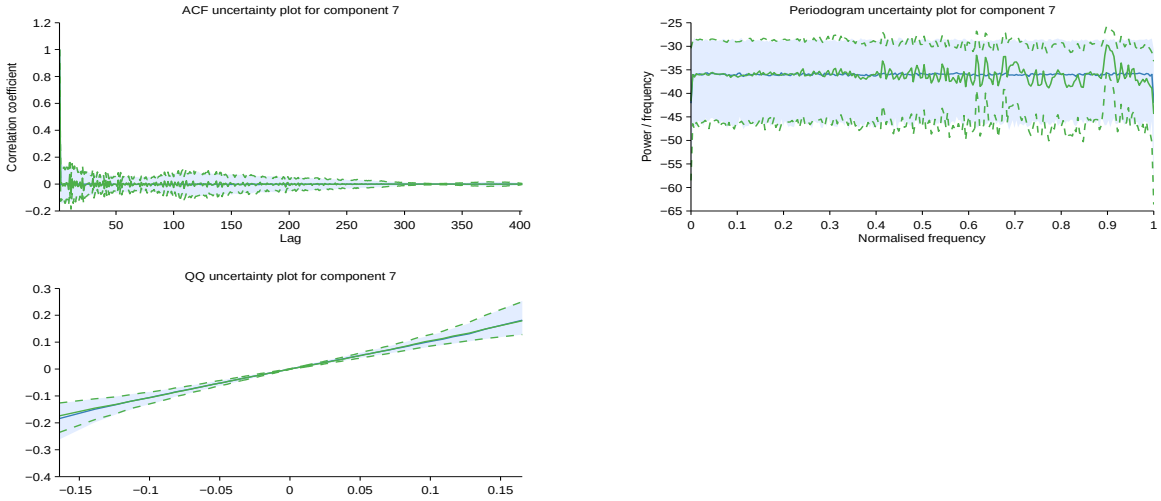


Figure 33: ACF (top left), periodogram (top right) and quantile-quantile (bottom left) uncertainty plots. The blue line and shading are the pointwise mean and 90% confidence interval of the plots under the prior distribution for component 7. The green line and green dashed lines are the corresponding quantities under the posterior.

Appendix B

Plots of data sets

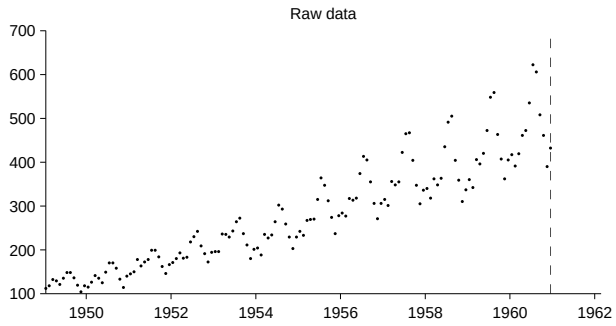


Figure B.1: Airline passengers

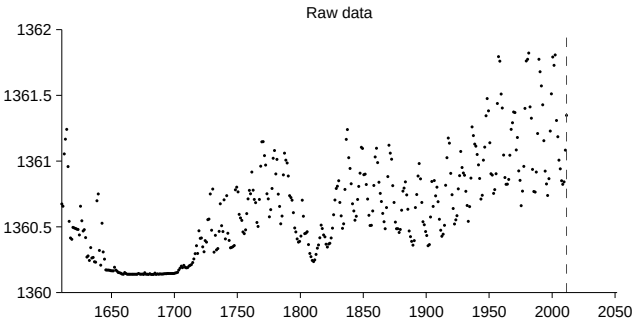


Figure B.2: Solar irradiance

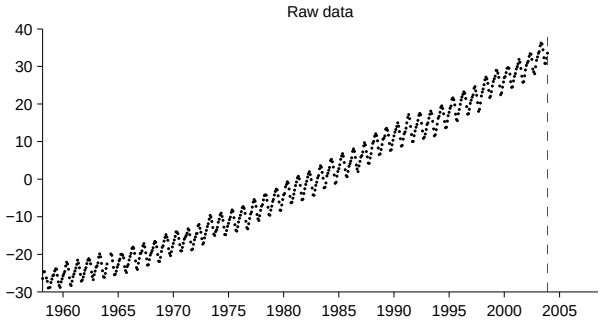


Figure B.3: Carbon dioxide levels at the Mauna Loa observatory

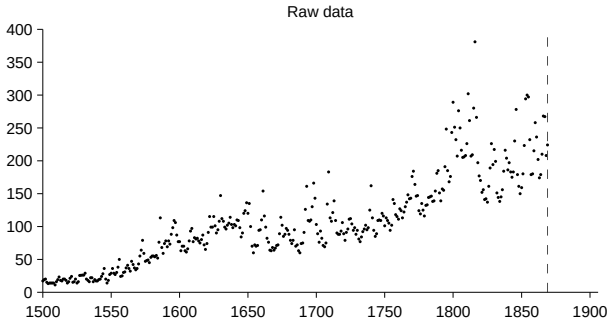


Figure B.4: Wheat prices

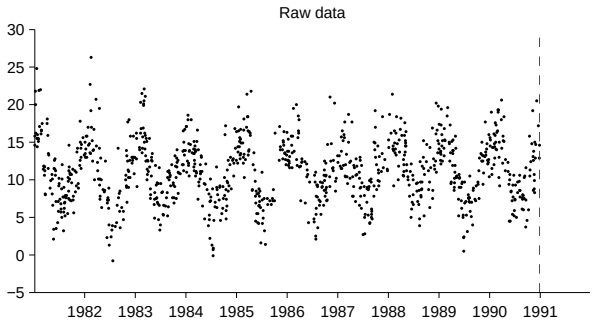


Figure B.5: Temperature in Melbourne, Australia

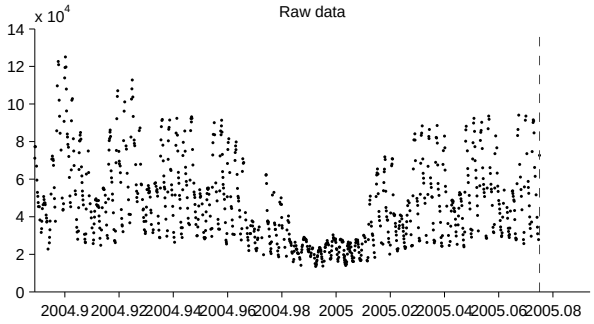


Figure B.6: Internet traffic

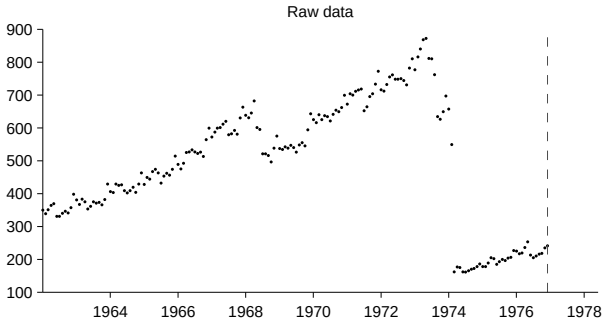


Figure B.7: Number of calls at a call centre

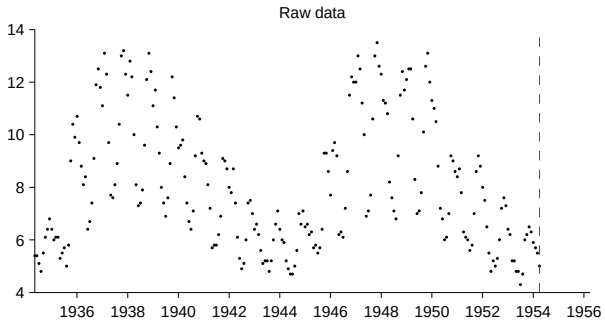


Figure B.8: Frequency at which the atmosphere becomes opaque to radio waves

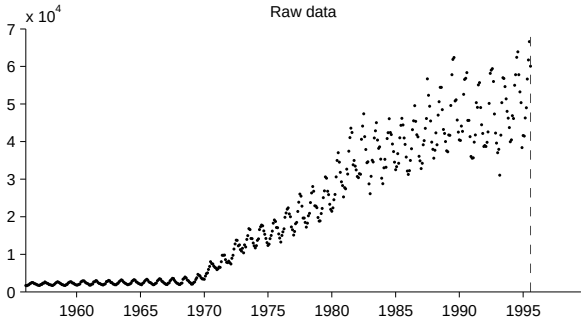


Figure B.9: Gas production

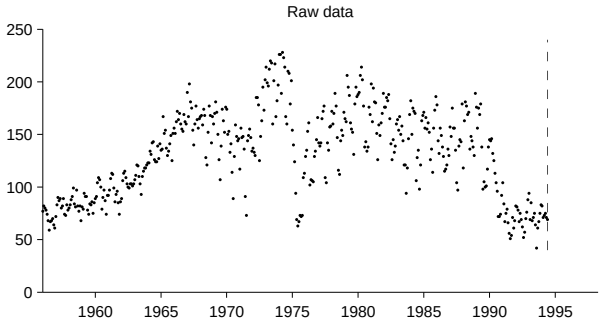


Figure B.10: Sulphuric acid production

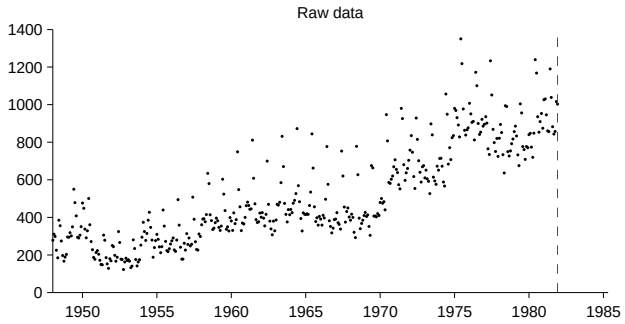


Figure B.11: Unemployment levels in the US

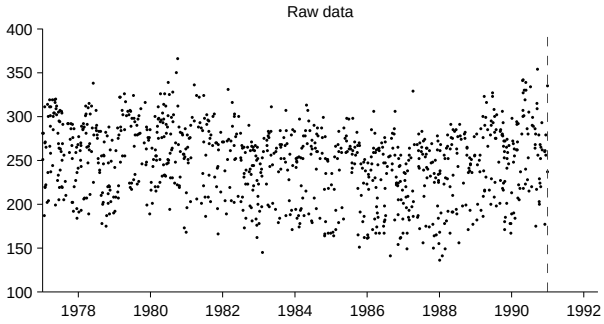


Figure B.12: Births in Quebec, Canada

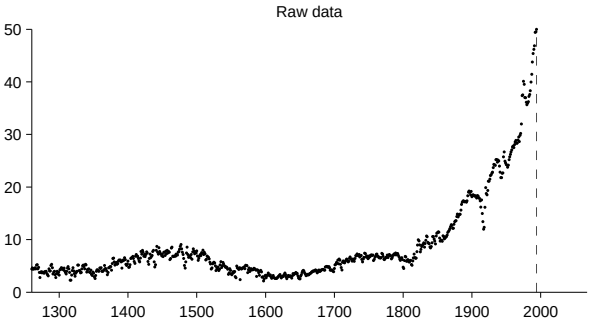


Figure B.13: Average wages in the UK

References

Evrim Acar, Tamara G Kolda, and Daniel M Dunlavy. All-at-once optimization for coupled matrix and tensor factorizations. In *Ninth workshop on mining and learning with graphs*, 2011. (page 47)

Evrim Acar, George E. Plopper, and Bülent Yener. Coupled analysis of in vitro and histology tissue samples to quantify structure-function relationship. *PLoS One*, 7(3): e32227, January 2012. (page 47)

Evrim Acar, Morten Arendt Rasmussen, Francesco Savorani, Tormod Næs, and Rasmus Bro. Understanding data fusion within the framework of coupled matrix and tensor factorizations. *Chemometrics Intellig. Lab. Syst.*, June 2013. (page 47)

Ryan Prescott Adams, George E Dahl, and Iain Murray. Incorporating side information in probabilistic matrix factorization with gaussian processes. In *Proceedings of the Twenty-Sixth Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 1–9, 2010. (page 46)

Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, September 2008. (pages 9, 10, and 16)

David J. Aldous. Representations for partially exchangeable arrays of random variables. *J. Multivar. Anal.*, 11(4):581–598, 1981. (pages 10, 26, 30, and 47)

David J. Aldous. More uses of exchangeability: Representations of complex random structures. In *Probability and Mathematical Genetics: Papers in Honour of Sir John Kingman*, 2010. (pages 14 and 30)

Marie Louise Max Andersen, Morten Arendt Rasmussen, Sven Pörksen, Jannet Svensson, Jennifer Vikre-Jørgensen, Jane Thomsen, Niels Thomas Hertel, Jesper Johannesen, Flemming Pociot, Jacob Sten Petersen, Lars Hansen, Henrik Bindesbøl Mortensen, and Lotte Brøndum Nielsen. Complex Multi-Block analysis identifies new immunologic and genetic disease progression patterns associated with the residual β-cell function 1 year after diagnosis of type 1 diabetes. *PLoS One*, 8(6):e64632, January 2013. (page 47)

Tim Austin. Exchangeable random arrays. Technical report, 2012. (page 30)

F Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 105–112, 2009. (page 56)

Francis R Bach, Gert R G Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the twenty-first international conference on Machine learning*, page 6. ACM, 4 July 2004. (page 89)

Adrian Barbu and Nathan Lay. An introduction to artificial prediction markets for classification. *J. Mach. Learn. Res.*, 13:2177–2204, 2012. (page 88)

M J Bayarri and J O Berger. Quantifying surprise in the data and model verification. *Bayes. Stat.*, 1999. (page 130)

M J Bayarri and M E Castellanos. Bayesian checking of the second levels of hierarchical models. *Stat. Sci.*, 22(3):322–343, August 2007. (page 130)

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. R. Stat. Soc. Series B Stat. Methodol.* (page 128)

P J Bickel. A distribution free version of the smirnov two sample test in the p-variate case. *Ann. Math. Stat.*, 40(1):1–23, 1 February 1969. (page 118)

W Bing, Z Wen-qiong, C Ling, and L Jia-hong. A GP-based kernel construction and optimization method for RVM. In *International Conference on Computer and Automation Engineering (ICCAE)*, volume 4, pages 419–423, 2010. (page 57)

C M Bishop. Pattern recognition and machine learning. 2006. (page 63)

David M Blei. Build, compute, critique, repeat: Data analysis with latent variable models. *Annual Review of Statistics and Its Application*, 1(1):203–232, 2014. (page 116)

S Bochner, M Tenenbaum, and H Pollard. *Lectures on Fourier Integrals*. Annals of mathematics studies. Princeton University Press, 1959. (pages 57 and 76)

Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–57, 15 July 2006. (page 116)

G E P Box, G M Jenkins, and G C Reinsel. *Time series analysis: forecasting and control*. 1976. (pages 59, 88, and 90)

George E P Box. Sampling and Bayes' inference in scientific modelling and robustness. *J. R. Stat. Soc. Ser. A*, 143(4):383–430, 1980. (pages 115 and 117)

Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 1 October 2001. (page 108)

Francois Caron and Emily B Fox. Bayesian nonparametric models of sparse and exchangeable random graphs. *arXiv preprint 1401.1137*, 2014. (pages 15 and 45)

David S. Choi and Patrick J. Wolfe. Co-clustering separately exchangeable network data. *Ann. Stat.*, December 2013. (pages 30 and 47)

M Christoudias, R Urtasun, and T Darrell. Bayesian localized multiple kernel learning. *Technical report, EECS Department, University of California, Berkeley*, 2009. (page 56)

Kacper Chwialkowski, Dino Sejdinovic, and Arthur Gretton. A wild bootstrap for degenerate kernel tests. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. (page 131)

R D Cook and S Weisberg. Residuals and influence in regression. *Mon. on Stat. and App. Prob.*,, 1982. (page 115)

Mary Kathryn Cowles and Bradley P Carlin. Markov chain monte carlo convergence diagnostics: A comparative review. *J. Am. Stat. Assoc.*, 91(434):883–904, 1 June 1996. (page 131)

Richard Threlkeld Cox. Probability, frequency and reasonable expectation. *Am. J. Phys.*, 14:1–13, January 1946. (page 2)

Deep learning tutorial authors. Deep learning tutorial - http://www.deeplearning.net/tutorial/, 2014. (page 124)

Persi Diaconis and Svante Janson. Graph limits and exchangeable random graphs. *arXiv preprint arXiv:0712.2749*, pages 1–26, 2007. (page 30)

L Diosan, A Rogozan, and J P Pecuchet. Evolving kernel functions for SVMs by genetic programming. In *Machine Learning and Applications, 2007*, pages 19–24, 2007. (page 57)

Daniele Durante and David Dunson. Bayesian logistic gaussian process models for dynamic networks. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, pages 194–201, 2014. (page 46)

D Duvenaud, H Nickisch, and C E Rasmussen. Additive gaussian processes. In *Advances in Neural Information Processing Systems*, 2011. (pages 56 and 61)

David Duvenaud. *Automatic Model Construction with Gaussian Processes*. PhD thesis, University of Cambridge, 2014. (pages 19 and 57)

David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, June 2013. (pages 49, 74, 75, and 76)

Beyza Ermis, Evrim Acar, and A. Taylan Cemgil. Link prediction via generalized coupled tensor factorisation. *arXiv preprint arXiv:1208.6231*, 2012. (page 47)

F Feroz, M P Hobson, E Cameron, and A N Pettitt. Importance nested sampling and the MultiNest algorithm. *arXiv preprint 1306.2144*, 10 June 2013. (page 63)

Emily B Fox and David B Dunson. Multiresolution gaussian processes. In *Advances in Neural Information Processing Systems*, pages 737–745, 2012. (page 89)

Jerome H. Friedman and Lawrence C. Rafsky. Multivariate generalizations of the Wald-Wolfowitz and smirnov Two-Sample tests. *Ann. Stat.*, 7(4):697–717, 1 July 1979. (page 118)

N Friedman, L Getoor, D Koller, and A Pfeffer. Learning probabilistic relational models. *IJCAI*, 1999. (page 43)

M Ganesalingam and W T Gowers. A fully automatic problem solver with human-style output. *arXiv preprint 1309.4501*, 2013. (page 88)

Sheng Gao, Ludovic Denoyer, and Patrick Gallinari. Link pattern prediction with tensor decomposition in multi-relational networks. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 2011. (page 47)

R Garnett, M. a. Osborne, S Reece, a. Rogers, and S J Roberts. Sequential bayesian prediction in the presence of changepoints and faults. *Comput. J.*, 53(9):1430–1446, February 2010. (page 89)

A E Gelfand, D K Dey, and H Chang. Model determination using predictive distributions with implementation via sampling-based methods. Technical Report 462, Stanford Uni CA Dept Stat, 1992. (page 130)

A Gelman, J B Carlin, H S Stern, D B Dunson, A Vehtari, and D B Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. (pages 116, 130, and 131)

Andrew Gelman. A Bayesian formulation of exploratory data analysis and goodness-of-fit testing. *Int. Stat. Rev.*, 2003. (page 130)

Andrew Gelman. Understanding posterior p-values. *Elec. J. Stat.*, 2013. (page 130)

Andrew Gelman, Xiao-Li Meng, and Hal Stern. Posterior predictive assessment of model fitness via realized discrepancies. *Stat. Sin.*, 6:733–807, 1996. (page 131)

Sean Gerrish and David M Blei. Black box variational inference. *arXiv preprint arXiv:1401.0118*. (page 63)

John Geweke. Getting it right. *J. Am. Stat. Assoc.*, 99(467):799–804, September 2004. (page 131)

Z Ghahramani and C E Rasmussen. Bayesian monte carlo. *Adv. Neural Inf. Process. Syst.*, 2002. (page 64)

Noah D Goodman, Vikash K Mansinghka, Daniel M Roy, Keith Bonawitz, and Joshua B Tenenbaum. Church : a language for generative models. In *Conf. on Unc. in Art. Int. (UAI)*, 2008. (page 116)

T Green et al. Prediction api: Every app a smart app. *Google Developers Blog, Apr*, 21, 2011. (page 88)

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Berhard Schölkopf, and Alexander Smola. A kernel method for the two-sample problem. *Journal of Machine Learning Research*, 1:1–10, 2008. (pages 116, 119, and 120)

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two sample test. *Journal of Machine Learning Research*, 13: 723–773, 2012. (page 132)

Roger B Grosse, Ruslan Salakhutdinov, William T Freeman, and Joshua B Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Conf. on Unc. in Art. Int. (UAI)*, 2012. (pages 50, 54, 57, and 116)

C Gu. *Smoothing spline ANOVA models*. Springer Verlag, 2002. (page 56)

Irwin Guttman. The use of the concept of a future observation in goodness-of-fit problems. *J. R. Stat. Soc. Series B Stat. Methodol.*, 29(1):83–100, 1967. (page 117)

T J Hastie and R J Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990. (page 52)

Trevor Hastie, Rob Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning*. Springer, 2009. (pages 64, 102, and 104)

Philipp Hennig and Martin Kiefel. Quasi-Newton methods: A new direction. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012. (page 64)

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint 1309.6835*, 26 September 2013. (page 63)

Geoffrey E Hinton. To recognize shapes, first learn to generate images. *Prog. Brain Res.*, 165:535–547, 2007. (pages 123 and 124)

Geoffrey E Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, 2006. (pages 123 and 126)

Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Stat. Sci.*, 14(4):382–401, 1999. (pages 64 and 106)

Peter D. Hoff. Modeling homophily and stochastic equivalence in symmetric relational data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 657–664, 2007. (pages 10, 16, 21, 30, 34, 41, 46, and 47)

Peter D. Hoff, Adrian E Raftery, and Mark S Handcock. Latent space approaches to social network analysis. *J. Am. Stat. Assoc.*, 97(460):1090–1098, December 2002. (pages 9 and 47)

Thomas Hoffman, Jan Puzicha, and Michael Jordan. Learning from dyadic data. In *Advances in Neural Information Processing Systems*, 1998. (page 47)

Tao Hong, Pierre Pinson, and Shu Fan. Global energy forecasting competition 2012. *Int. J. Forecast.*, 30(2):357–363, April 2014. (pages 98 and 109)

D N Hoover. Row-column exchangeability and a generalized model for probability. In *Exchangeability in Probability and Statistics*, pages 281–291, 1982. (page 30)

Douglas N Hoover. Relations on probability spaces and arrays of random variables. Technical report, Institute for Advanced Study, Princeton, 1979. (pages 10, 26, 30, and 47)

Harold Hotelling. A generalized t-test and measure of multivariate dispersion. In *Proc. 2nd Berkeley Symp. Math. Stat. and Prob.* The Regents of the University of California, 1951. (page 118)

Rob J Hyndman. Time series data library. `http://data.is/TSDLdemo`. Accessed: 2013. (page 89)

Tomoharu Iwata, David Duvenaud, and Zoubin Ghahramani. Warped mixtures for nonparametric cluster shapes. In *Conf. on Unc. in Art. Int. (UAI)*. arxiv.org, 2013. (page 123)

Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003. (pages 1, 2, and 44)

Yu-Ru Lin Jimeng, Sun Paul, Castro Ravi, Konuru Hari, and Sundaram Aisling. MetaFac : Community discovery via relational hypergraph factorization. pages 527–535, 2009. (page 47)

Michael I Jordan. On statistics, computation and scalability. *Bernoulli*, 19(4):1378–1390, September 2013. (pages 2 and 133)

Olav Kallenberg. Symmetries on random arrays and set-indexed processes. *J. Theoret. Probab.*, 5(4):727–765, October 1992. (pages 10 and 26)

Olav Kallenberg. Multivariate sampling and the estimation problem for exchangeable arrays. *Journal of Theoretical Probability*, 12(3):859–883, 1999. (pages 29, 30, 36, 39, and 47)

Olav Kallenberg. *Foundations of Modern Probability*. Springer, 2002. (page 11)

Olav Kallenberg. *Probabilistic symmetries and invariance principles*. Springer, 2005. (page 3)

Andrej Karpathy and Li Fei-Fei. Deep Visual-Semantic alignments for generating image descriptions. 2014. (page 88)

Charles Kemp and Joshua B Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, August 2008. (page 57)

Charles Kemp, J B Tenenbaum, T L Griffiths, T Yamada, and N Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 2006. (pages 16 and 47)

Edgar D Klenske, Melanie N Zeilinger, Bernhard Schölkopf, and Philipp Hennig. Non-parametric dynamics estimation for time periodic systems. In *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 486–493. IEEE, 2013. (page 57)

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, August 2009. (page 47)

Gabriel Kronberger and Michael Kommenda. Evolution of covariance functions for gaussian process regression using genetic programming. *arXiv:1305.3794*, 2013. (pages 57 and 74)

Neil D. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Mach. Learn. Res.*, 6:1783–1816, 2005. (pages 15, 21, and 56)

Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with gaussian processes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1–8. ACM Press, 2009. (pages 10, 15, and 47)

Miguel Lázaro-Gredilla and Joaquin Quiñonero-Candela. Sparse spectrum gaussian process regression. 11:1865–1881, 2010. (page 57)

J Lean, J Beer, and R Bradley. Reconstruction of solar irradiance since 1610: Implications for climate change. *Geophys. Res. Lett.*, 22(23):3195–3198, 1995. (pages 59, 85, and 87)

Y LeCun, B Boser, J S Denker, D Henderson, R E Howard, W Hubbard, and L D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1:541–551, 1989. (page 53)

D A Lind, W G Marchal, S A Wathen, and B W Magazine. Basic statistics for business and economics. 2006. (pages 89 and 127)

Christoph Lippert, Stefan Hagen Weber, Yi Huang, Volker Tresp, Matthias Schubert, and Hans-Peter Kriegel. Relation prediction in Multi-Relational domains using matrix factorization. (Siso), 2008. (page 47)

J R Lloyd, P Orbanz, Z Ghahramani, and D M Roy. Exchangeable databases and their functional representation. In *Frontiers of Network Analysis: Methods, Models, and Applications at NIPS*, 2013. (page 29)

James Robert Lloyd. GEFCom2012 hierarchical load forecasting: Gradient boosting machines and gaussian processes. In *International Journal of Forecasting*, 2013. (pages 57 and 97)

James Robert Lloyd, Peter Orbanz, Daniel M. Roy, and Zoubin Ghahramani. Random function priors for exchangeable graphs and arrays. In *Advances in Neural Information Processing Systems (NIPS)*, 2012. (pages 9, 30, 34, 41, 47, and 63)

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Automatic construction and Natural-Language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, July 2014. (pages 71 and 116)

L Lovász. *Large Networks and Graph Limits*. American Mathematical Society colloquium publications. American Mathematical Society, 2012. (pages 15 and 45)

David J MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. (page 54)

E C Marshall and D J Spiegelhalter. Identifying outliers in bayesian hierarchical models: a simulation-based approach. *Bayesian Anal.*, 2(2):409–444, June 2007. (page 130)

G McLachlan and D Peel. *Finite Mixture Models*. Wiley series in probability and statistics: Applied probability and statistics. Wiley, 2004. (page 122)

Edward Meeds, Zoubin Ghahramani, Radford M. Neal, and Sam T. Roweis. Modeling dyadic data with binary latent factors. In *Advances in Neural Information Processing Systems (NIPS)*. Citeseer, 2007. (pages 16, 46, and 47)

A Menon. Link prediction via matrix factorization. *Machine Learning and Knowledge Discovery*, pages 437–452, 2011. (page 45)

Michel Meulders, Andrew Gelman, Iven Van Mechelen, and Paul De Boeck. Generalizing the probability matrix decomposition model: an example of bayesian model checking and model expansion. In *Assumptions, Robustness, and Estimation Methods in Multivariate Modeling*. Citeseer, 1998. (page 116)

B Milch, B Marthi, S Russel, D Sontag, D L Ong, and A Kolobov. BLOG: Probabilistic models with unknown objects. In *Proc. Int. Joint Conf. on Artificial Intelligence*, 2005. (page 116)

Kurt T. Miller, Thomas L. Griffiths, and Michael I. Jordan. Nonparametric latent feature models for link prediction. *Adv. Neural Inf. Process. Syst.*, pages 1276–1284, 2009. (pages 16, 21, 46, and 47)

Alfred Müller. Integral probability metrics and their generating classes of functions. *Adv. Appl. Probab.*, 29(2):429–443, 1 June 1997. (page 119)

Iain Murray and Ryan Prescott Adams. Slice sampling covariance hyperparameters of latent gaussian models. In *Advances in Neural Information Processing Systems (NIPS)*, 2010. (page 63)

Iain Murray, Ryan Prescott Adams, and David J C Mackay. Elliptical slice sampling. *J. Mach. Learn. Res.*, 9:541–548, 2010. (page 21)

Radford M Neal. Slice sampling. *Ann. Stat.*, 31(3):705–741, 1 June 2003. (page 20)

Maximilian Nickel. A Three-Way model for collective learning on Multi-Relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011. (page 47)

Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic blockstructures. *J. Am. Stat. Assoc.*, 96(455):1077–1087, 2001. (page 47)

Nutonian. Eureqa, 2011. (pages 85 and 89)

A O'Hagan. Bayes-Hermite quadrature. *J. Stat. Plan. Inference*, 29:245–260, 1991. (page 64)

A O'Hagan. HSSS model criticism. *Highly Structured Stochastic Systems*, pages 423–444, 2003. (pages 115, 131, and 132)

Michael A. Osborne, Roman Garnett, and Stephen J. Roberts. Gaussian processes for global optimization. In *3rd International Conference on Learning and Intelligent Optimization (LION3)*, 2009. (page 109)

Konstantina Palla, David A Knowles, and Zoubin Ghahramani. An infinite latent attribute model for network data. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012. (pages 16 and 47)

E Parzen. On estimation of a probability density function and mode. *Ann. Math. Stat.*, 1962. (page 120)

D Peel and G J McLachlan. Robust mixture modelling using the t distribution. *Stat. Comput.*, 10(4):339–348, 1 October 2000. (page 123)

T A Plate. Accuracy versus interpretability in flexible modeling: Implementing a tradeoff using gaussian process models. *Behaviormetrika*, 26:29–50, 1999. (page 56)

H Poon and P Domingos. Sum-product networks: a new deep architecture. In *Conference on Uncertainty in AI*, 2011. (page 53)

K Popper. *The logic of scientific discovery*. Routledge, 2005. (page 131)

J Quiñonero Candela and C E Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, 2005. (pages 20 and 110)

C E Rasmussen and Z Ghahramani. Occam's razor. In *Advances in Neural Information Processing Systems*, 2001. (page 54)

C E Rasmussen and C K Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, USA, 2006. (pages 3, 51, 54, 57, 58, 59, 63, 74, 116, and 120)

James M Robins, Aad van der Vaart, and Valerie Venture. Asymptotic distribution of p-values in composite null models. *J. Am. Stat. Assoc.*, 95(452):1143–1156, 2000. (page 130)

Murray Rosenblatt. Remarks on some nonparametric estimates of a density function. *Ann. Math. Stat.*, 27(3):832–837, September 1956. (page 120)

Daniel M. Roy and Yee Whye Teh. The mondrian process. In *Advances in Neural Information Processing Systems (NIPS)*. Citeseer, 2009. (pages 16, 18, 30, 34, 41, and 47)

Donald B Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *Ann. Stat.*, 12(4):1151–1172, 1984. (page 117)

D Ruppert, M P Wand, and R J Carroll. *Semiparametric regression*. Cambridge University Press, 2003. (page 56)

Yunis Saatçi, Richard D Turner, and Carl E Rasmussen. Gaussian process change point models. In *International Conference on Machine Learning*. machinelearning.wustl.edu, 2010. (page 89)

Yunus Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011. (page 16)

Ruslan Salakhutdinov and Geoffrey Hinton. Using deep belief nets to learn covariance kernels for gaussian processes. *Adv. Neural Inf. Process. Syst.*, 20:1249–1256, 2008. (pages 46 and 56)

Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1–8, 2008. (pages 9, 10, and 21)

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, April 2009. (page 57)

G Schwarz. Estimating the dimension of a model. *Ann. Stat.*, 6(2):461–464, 1978. (pages 55 and 63)

Qiuzi Shangguan, Liang Hu, Jian Cao, and Guandong Xu. Book recommendation based on joint multi-relational model. In *2012 Second International Conference on Cloud and Green Computing*, pages 523–530. Ieee, November 2012. (page 47)

B W Silverman. Some aspects of the spline smoothing approach to Non-Parametric regression curve fitting. *J. R. Stat. Soc. Series B Stat. Methodol.*, 47(1):1–52, 1 January 1985. (page 20)

Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization categories and subject descriptors. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008a. (page 47)

Ajit P Singh and Geoffrey J Gordon. A unified view of matrix factorization models. In *Machine Learning and Knowledge Discovery in Databases*, pages 358–373, 2008b. (page 47)

Ajit P Singh and Geoffrey J Gordon. A bayesian matrix factorization model for relational data. *arXiv preprint arXiv:1203.3517*, 2012. (page 47)

John Skilling. Nested sampling for general bayesian computation. *Bayesian Anal.*, 1(4): 833–859, December 2006. (page 63)

Alex J Smola and Peter Bartlett. Sparse greedy gaussian process regression. In *Advances in Neural Information Processing Systems 13*, 2001. (page 20)

Edward Snelson and Zoubin Ghahramani. Local and global sparse gaussian process approximations. *Artificial Intelligence and Statistics*, 2007. (page 112)

J Snoek, H Larochelle, and R P Adams. Practical bayesian optimization of machine learning algorithms. *arXiv preprint arXiv:1206.2944*, 2012. (pages 63, 88, and 109)

Stan Development Team. Stan: A C++ library for probability and sampling, version 2.2, 2014. (page 116)

M L Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer New York, 1999. (page 95)

Stephen M Stigler. Do robust estimators work with real data? *Ann. Stat.*, 5(6):1055–1098, November 1977. (page 120)

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-Thaw bayesian optimization. *arXiv preprint 1406.3896*, 16 June 2014. (pages 64 and 65)

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, New York, NY, USA, 2013. ACM. (pages 88 and 116)

Michalis K. Titsias and Neil D. Lawrence. Efficient sampling for gaussian process inference using control variables. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1681–1688, 2008. (page 21)

L Todorovski and S Dzeroski. Declarative bias in equation discovery. In *International Conference on Machine Learning*, pages 376–384, 1997. (pages 50 and 57)

J Ullman and J Widom. *A First Course in Database Systems*. Prentice Hall, 2002. (page 31)

Aad van der Vaart and Harry van Zanten. Information rates of nonparametric gaussian process methods. *J. Mach. Learn. Res.*, 12:2095–2119, July 2011. (page 95)

Jarno Vanhatalo and Aki Vehtari. Modelling local and global phenomena with sparse gaussian processes. *arXiv preprint 1206.3290*, 13 June 2012. (page 112)

Aki Vehtari and Janne Ojanen. A survey of Bayesian predictive methods for model assessment, selection and comparison. *Stat. Surv.*, 6:142–228, 2012. (page 130)

G Wahba. *Spline models for observational data*. Society for Industrial Mathematics, 1990. (page 56)

Grace Wahba. An introduction to (smoothing spline) ANOVA models in RKHS with examples in geographical data, medicine, atmospheric science and machine learning. 19 October 2004. (page 56)

Grace Wahba, Xiwu Lin, Fangyu Gao, Dong Xiang, Ronald Klein, and Barbara Klein. The Bias-Variance tradeoff and the randomized GACV. In M J Kearns, S A Solla, and D A Cohn, editors, *Advances in Neural Information Processing Systems 11*, pages 620–626. MIT Press, 1999. (page 20)

Chunyi Wang and Radford M Neal. Gaussian process regression with heteroscedastic or Non-Gaussian residuals. 26 December 2012. (page 43)

Yuchung J. Wang and George Y. Wong. Stochastic blockmodels for directed graphs. *J. Am. Stat. Assoc.*, 82(397):8–19, 1987. (page 47)

T Washio, H Motoda, Y Niwa, and Others. Discovering admissible model equations from observed data based on scale-types and identity constraints. In *International Joint Conference On Artifical Intelligence*, volume 16, pages 772–779, 1999. (page 57)

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process covariance kernels for pattern discovery and extrapolation. In *Proc. Int. Conf. Machine Learn.*, 2013. (pages 57, 76, 89, and 127)

Patrick J. Wolfe and Sofia C. Olhede. Nonparametric graphon estimation. *arXiv preprint arXiv:1309.5936*, pages 1–52, 2013. (pages 15, 30, 45, and 47)

Zenglin Xu, Feng Yan, and Yuan Qi. Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012. (pages 9, 16, and 47)

Zhao Xu, Volker Tresp, and Kai Yu. Infinite hidden relational models. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2006. (pages 16 and 47)

Feng Yan, Zenglin Xu, and Yuan Qi. Sparse matrix-variate gaussian process blockmodels for network modeling. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence (UAI)*, 2011. (pages 16 and 47)

Dawei Yin, Shengbo Guo, Boris Chidlovskii, Brian D Davison, Cedric Archambeau, and Guillaume Bouchard. Connecting comments and tags : Improved modeling of social tagging systems categories and subject descriptors. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 547–556, 2013. (page 47)

Kai Yu and Wei Chu. Gaussian process models for link analysis and transfer learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2008. (page 47)

Shandian Zhe, Youngja Park, and Ian Molloy. DinTucker : Scaling up gaussian process models on multidimensional arrays with billions of elements. *arXiv preprint 1311.2663*, 2013. (page 16)