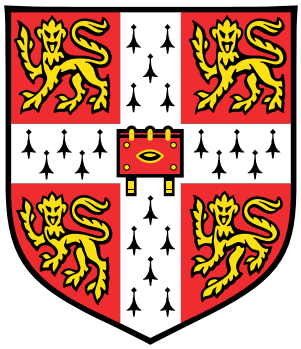
Structured Gaussian Process Models



David Kristjanson Duvenaud

Department of Engineering

University of Cambridge

This dissertation is submitted for the degree of

Doctor of Philosophy

Pembroke College

June 2014

I would like to dedicate this thesis to my loving parents ...

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 including appendices, bibliography, footnotes, tables and equations and has less than 150 figures.

David Kristjanson Duvenaud
June 2014

Acknowledgements

I'd like to thank my advisor, Carl Edward Rasmussen. It was wonderful having an advisor who had spent many years thinking deeply about the business of modeling. "The devil is in the details". I'd also like to thank Zoubin Ghahramani for providing much encouragement.

Thanks to Michael Osborne, whose thesis was an inspiration. Discussions with Roman Garnett made the research process enjoyable. Sinead Williamson and Peter Orbanz provided valuable advice in my early years in the lab. Philipp Hennig mentored me during my time in Tübingen.

I'd like to thank Andrew McHutchon, Konstantina Palla, Alex Davies and Neil Houlsby for making the lab feel like a family. I'd like to thank James Lloyd for all the endless discussions that really helped refine my understanding of a lot of things, and for keeping a level head even in the depths of deadline death-marches. Christian Steinruecken constantly reminded me that amazing things are hidden all around.

Abstract

This is where you write your abstract ...

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "I only work on intractable nonparametrics - Gaussian processes don't count."
>
> Sinead Williamson, personal communication

1.1 Regression

The general problem of regression consists of learning a function f mapping from some input space $\mathcal{X}$ to some output space $\mathcal{Y}$. We would like an expressive language which can represent both simple parametric forms of f such as linear, polynomial, etc. and also complex nonparametric functions specified in terms of properties such as smoothness, periodicity, etc. Fortunately, Gaussian processes (GPs) provide a very general and analytically tractable way of capturing both simple and complex functions.

1.2 Gaussian process models

Gaussian processes are a flexible and tractable prior over functions, useful for solving regression and classification tasksRasmussen and Williams (2006). The kind of structure which can be captured by a GP model is mainly determined by its *kernel*: the covariance function. One of the main difficulties in specifying a Gaussian process model is in choosing a kernel which can represent the structure present in the data. For small to medium-sized datasets, the kernel has a large impact on modeling efficacy.

Gaussian processes are distributions over functions such that any finite subset of function evaluations, $(f(x_1), f(x_2), \ldots f(x_N))$, have a joint Gaussian distribution (Rasmussen and Williams, 2006). A GP is completely specified by its mean function, $\mu(x) = \mathbb{E}(f(x))$

and kernel (or covariance) function $k(x, x') = \text{Cov}(f(x), f(x'))$. It is common practice to assume zero mean, since marginalizing over an unknown mean function can be equivalently expressed as a zero-mean GP with a new kernel. The structure of the kernel captures high-level properties of the unknown function, f, which in turn determines how the model generalizes or extrapolates to new data. We can therefore define a language of regression models by specifying a language of kernels.

1.2.1 Useful properties of Gaussian process models

- **Tractable inference** Given a kernel function, the posterior distribution can be computed exactly in closed form. This is a rare property for nonparametric models to have.

- **Expressivity** by choosing different covariance functions, we can express a very wide range of modeling assumptions.

- **Integration over hypotheses** the fact that a GP posterior lets us exactly integrate over a wide range of hypotheses means that overfitting is less of an issue than in comparable model classes - for example, neural nets.

- **Marginal likelihood** A side benefit of being able to integrate over all hyoptheses is that we compute the *marginal likelihood* of the data given the model. This gives us a principled way of comparing different Gaussian process models.

- **Closed-form posterior** The posterior predictive distribution of a GP is another GP. This means that GPs can easily be composed with other models or decision procedures. For example, (⋆) Carl's reinforcement learning work.

Figure 1.1 shows a Gaussian process posterior. Typically, it's rendered with the mean and +- 2SD, but there's nothing special about mean.

1.2.2 Why assume zero-mean?

It is common practice to assume zero mean, since marginalizing over an unknown mean function can be equivalently expressed as a zero-mean GP with a new kernel.

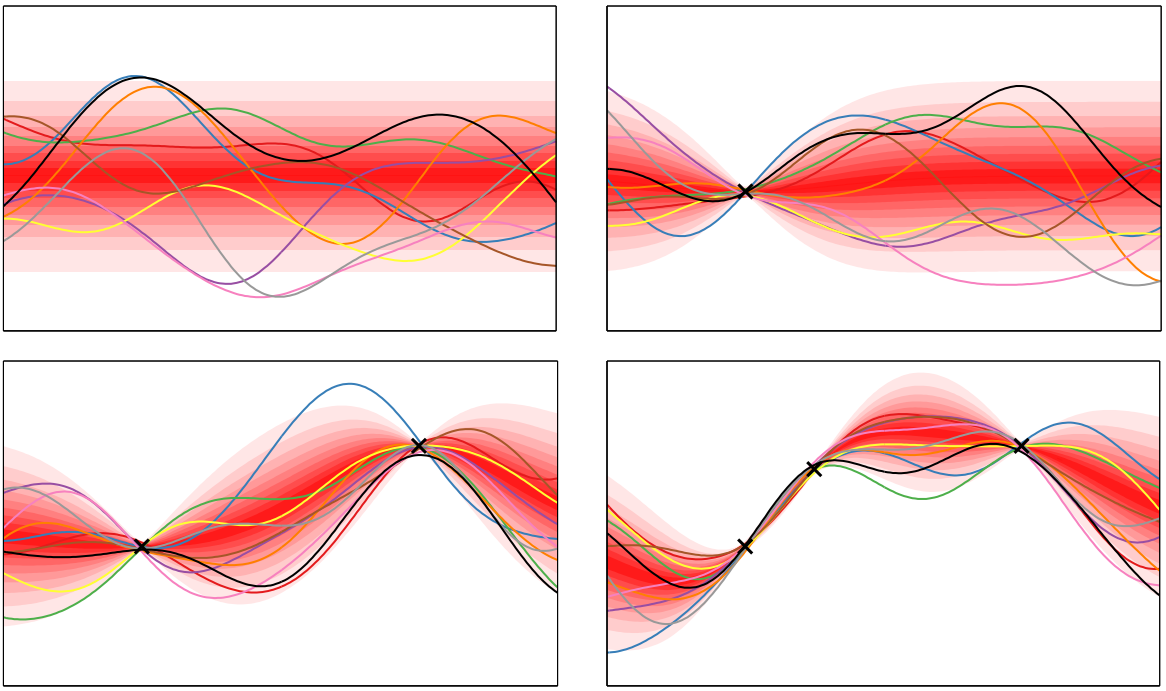


Fig. 1.1 A visual representation of a one-dimensional Gaussian process posterior. Red isocountours show the marginal density at each input location. Coloured lines are samples from the posterior.

1.3 Latent Variable Models

Besides being useful for modeling functions, a simple extension allows GPs to be useful for general density modeling.

Unfortunately, this extension causes many of the useful properties of the GP not to hold.

The GP-LVM can also be thought of as a method for modeling the covariance matrix between all rows of Y using a number of parameters which grows linearly with N.

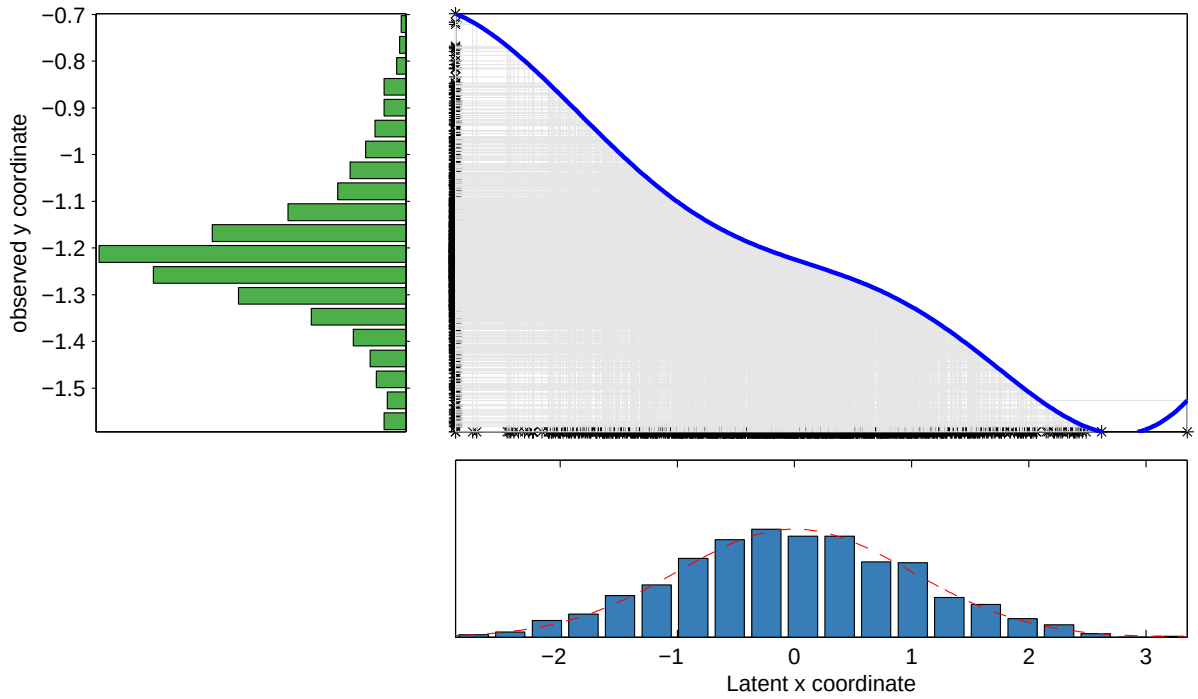


Fig. 1.2 A visual representation of the Gaussian process latent variable model. Bottom: density and samples from a 1D Gaussian, specifying the distribution $p(\mathbf{X})$ in the latent space. Top Right: A function drawn from a GP prior. Left: A nonparametric density defined by warping the latent density through the function drawn from a GP prior.

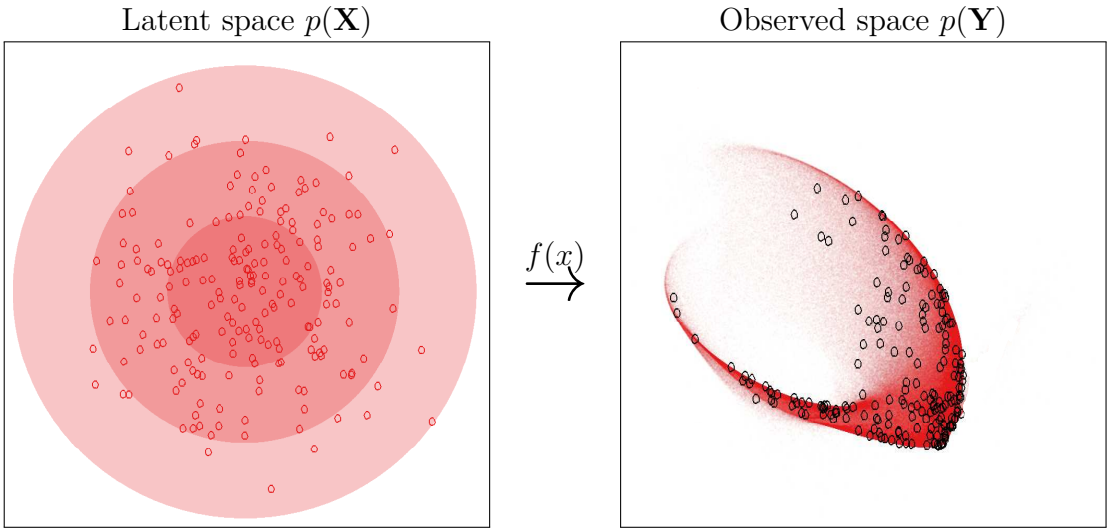


Fig. 1.3 A visual representation of the Gaussian process latent variable model. Left: Isocontours and samples from a 2D Gaussian, specifying the distribution $p(\mathbf{X})$ in the latent space. Right: Density and samples from a nonparametric density defined by warping the latent density through a function drawn from a GP prior.

Chapter 2

Expressing Structure through Kernels

Kernels specify similarity between function values of two objects, not between similarity of objects

Gaussian process 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 generalization properties of the model. In this section, we review the ways in which kernel families[1]can be composed to express diverse priors over functions.

There has been significant work on constructing GP kernels and analyzing their properties, summarized in Chapter 4 of Rasmussen and Williams (2006). Commonly used kernels families include the squared exponential (SE), periodic (Per), linear (Lin), and rational quadratic (RQ) (see Figure 2.1 and the appendix).

Kernel definitions For scalar-valued inputs, the squared exponential (SE), periodic (Per), linear (Lin), and rational quadratic (RQ) kernels are defined as follows:

$$
\begin{aligned}
k_{\mathrm{SE}}(x, x') &= \sigma^2 \exp\left(-\tfrac{(x-x')^2}{2\ell^2}\right) \\
k_{\mathrm{Per}}(x, x') &= \sigma^2 \exp\left(-\tfrac{2\sin^2(\pi(x-x')/p)}{\ell^2}\right) \\
k_{\mathrm{Lin}}(x, x') &= \sigma_b^2 + \sigma_v^2(x-\ell)(x'-\ell) \\
k_{\mathrm{RQ}}(x, x') &= \sigma^2\left(1 + \tfrac{(x-x')^2}{2\alpha\ell^2}\right)^{-\alpha}
\end{aligned}
$$

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

The elements of this language are a set of base kernels capturing different function properties, and a set of composition rules which combine kernels to yield other valid kernels. Our base kernels are white noise (WN), constant (C), linear (Lin), squared exponential (SE) and periodic (Per), which on their own encode for uncorrelated noise, constant functions, linear functions, smooth functions and periodic functions respectively[2]. The composition rules are addition and multiplication:

$$k_1 + k_2 = k_1(x, x') + k_2(x, x') \tag{2.1}$$

$$k_1 \times k_2 = k_1(x, x') \times k_2(x, x') \tag{2.2}$$

Combining kernels using these operations can yield kernels encoding for richer structures such as approximate periodicity (SE $\times$ Per) or smooth functions with linear trends (SE + Lin).

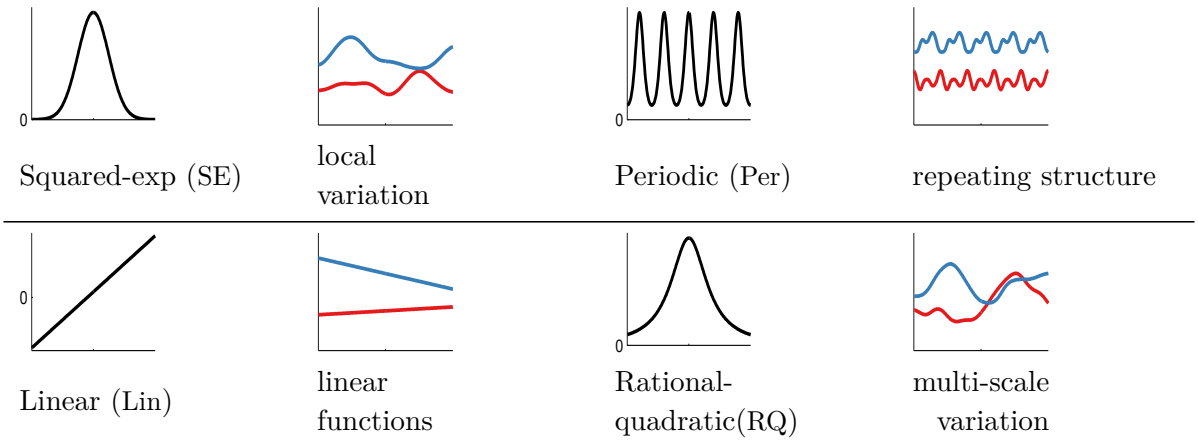


Fig. 2.1 Examples of structures expressible by base kernels. Left and third columns: base kernels $k(\cdot, 0)$. Second and fourth columns: draws from a GP with each repective kernel. The x-axis has the same range on all plots.

Composing Kernels Positive semidefinite kernels (i.e. those which define valid covariance functions) are closed under addition and multiplication. This allows one to create richly structured and interpretable kernels from well understood base components.

All of the base kernels we use are one-dimensional; kernels over multidimensional inputs are constructed by adding and multiplying kernels over individual dimensions. These dimensions are represented using subscripts, e.g. SE_2 represents an SE kernel over the second dimension of x.

[2]Definitions of kernels are in the supplementary material.

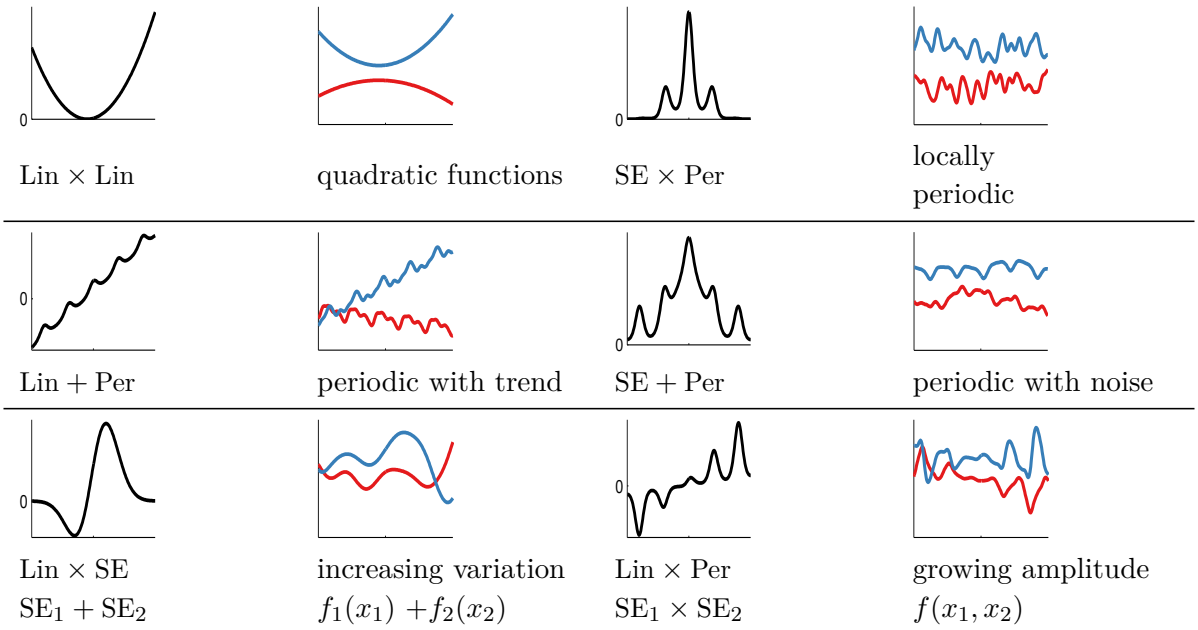


Fig. 2.2 Examples of one-dimensionalstructures expressible by composite kernels. Left column and third columns: composite kernels $k(\cdot, 0)$. Plots have same meaning as in figure 2.1.

2.1 Structure through additivity

By summing kernels, we can model the data as a sum of independent functions, possibly representing different structures. Suppose functions f_1, f_2 are draw from independent GP priors, $f_1 \sim \mathcal{GP}(\mu_1, k_1)$, $f_2 \sim \mathcal{GP}(\mu_2, k_2)$. Then

$$f := f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2). \tag{2.3}$$

From a kernel point of view,
if $k_1(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^\mathsf{T}\phi_1(\mathbf{x}')$ and $k_2 = \phi_2(\mathbf{x})^\mathsf{T}\phi_2(\mathbf{x}')$ then

$$k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^\mathsf{T}\phi_1(\mathbf{x}') + \phi_2(\mathbf{x})^\mathsf{T}\phi_2(\mathbf{x}') \tag{2.4}$$

In other words, the features of k_1+k_2 are the concatenation of the two sets of features.

In time series models, sums of 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 4 of figure 2.3, respectively.

A theme throughout this thesis is exploring the idea that a lot of the expressivity of GP models comes from the fact that these models can be combined and decomposed additively.

2.1.1 Derivation of Component Marginal Variance

In this section, we derive the posterior marginal variance and covariance of the additive components of a GP. These formulas let us plot the marginal variance of each component separately. These formulas can also be used to examine the posterior covariance between pairs of components.

Let us assume that our function $\mathbf{f}$ is a sum of two functions, $\mathbf{f}_1$ and $\mathbf{f}_2$, where $\mathbf{f} = \mathbf{f}_1 + \mathbf{f}_2$. If $\mathbf{f}_1$ and $\mathbf{f}_2$ are a priori independent, and $\mathbf{f}_1 \sim \mathrm{GP}(\mu_1, k_1)$ and $\mathbf{f}_2 \sim \mathrm{GP}(\mu_2, k_2)$, then

$$\begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_1^\star \\ \mathbf{f}_2 \\ \mathbf{f}_2^\star \\ \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_1^\star \\ \mu_2 \\ \mu_2^\star \\ \mu_1 + \mu_2 \\ \mu_1^\star + \mu_2^\star \end{bmatrix}, \begin{bmatrix} \mathbf{k}_1 & \mathbf{k}_1^\star & 0 & 0 & \mathbf{k}_1 & \mathbf{k}_1^\star \\ \mathbf{k}_1^\star & \mathbf{k}_1^{\star\star} & 0 & 0 & \mathbf{k}_1^\star & \mathbf{k}_1^{\star\star} \\ 0 & 0 & \mathbf{k}_2 & \mathbf{k}_2^\star & \mathbf{k}_2 & \mathbf{k}_2^\star \\ 0 & 0 & \mathbf{k}_2^\star & \mathbf{k}_2^{\star\star} & \mathbf{k}_2^\star & \mathbf{k}_2^{\star\star} \\ \mathbf{k}_1 & \mathbf{k}_1^\star & \mathbf{k}_2 & \mathbf{k}_2^\star & \mathbf{k}_1 + \mathbf{k}_2 & \mathbf{k}_1^\star + \mathbf{k}_2^\star \\ \mathbf{k}_1^\star & \mathbf{k}_1^{\star\star} & \mathbf{k}_2^\star & \mathbf{k}_2^{\star\star} & \mathbf{k}_1^\star + \mathbf{k}_2^\star & \mathbf{k}_1^{\star\star} + \mathbf{k}_2^{\star\star} \end{bmatrix} \right) \tag{2.5}$$

where $\mathbf{k}_1 = k_1(\mathbf{X}, \mathbf{X})$ and $\mathbf{k}_1^\star = k_1(\mathbf{X}^\star, \mathbf{X})$.

By the formula for Gaussian conditionals:

$$\mathbf{x}_A | \mathbf{x}_B \sim \mathcal{N}(\mu_A + \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}(\mathbf{x}_B - \mu_B), \boldsymbol{\Sigma}_{AA} - \boldsymbol{\Sigma}_{AB}\boldsymbol{\Sigma}_{BB}^{-1}\boldsymbol{\Sigma}_{BA}), \tag{2.6}$$

we get that the conditional variance of a Gaussian conditioned on its sum with another Gaussian is given by

$$\begin{aligned} \mathbf{f}_1(\mathbf{x}^\star) | \mathbf{f}(\mathbf{x}) \sim \mathcal{N}\Big(& \mu_1(\mathbf{x}^\star) + \mathbf{k}_1(\mathbf{x}^\star, \mathbf{x}) \left[\mathbf{K}_1(\mathbf{x}, \mathbf{x}) + \mathbf{K}_2(\mathbf{x}, \mathbf{x})\right]^{-1} (\mathbf{f}(\mathbf{x}) - \mu_1(\mathbf{x}) - \mu_2(\mathbf{x})), \\ & \mathbf{k}_1(\mathbf{x}^\star, \mathbf{x}^\star) - \mathbf{k}_1(\mathbf{x}^\star, \mathbf{x}) \left[\mathbf{K}_1(\mathbf{x}, \mathbf{x}) + \mathbf{K}_2(\mathbf{x}, \mathbf{x})\right]^{-1} \mathbf{k}_1(\mathbf{x}, \mathbf{x}^\star) \Big). \end{aligned} \tag{2.7}$$

These formulae express the posterior model uncertainty about different components of the signal, integrating over the possible configurations of the other components.

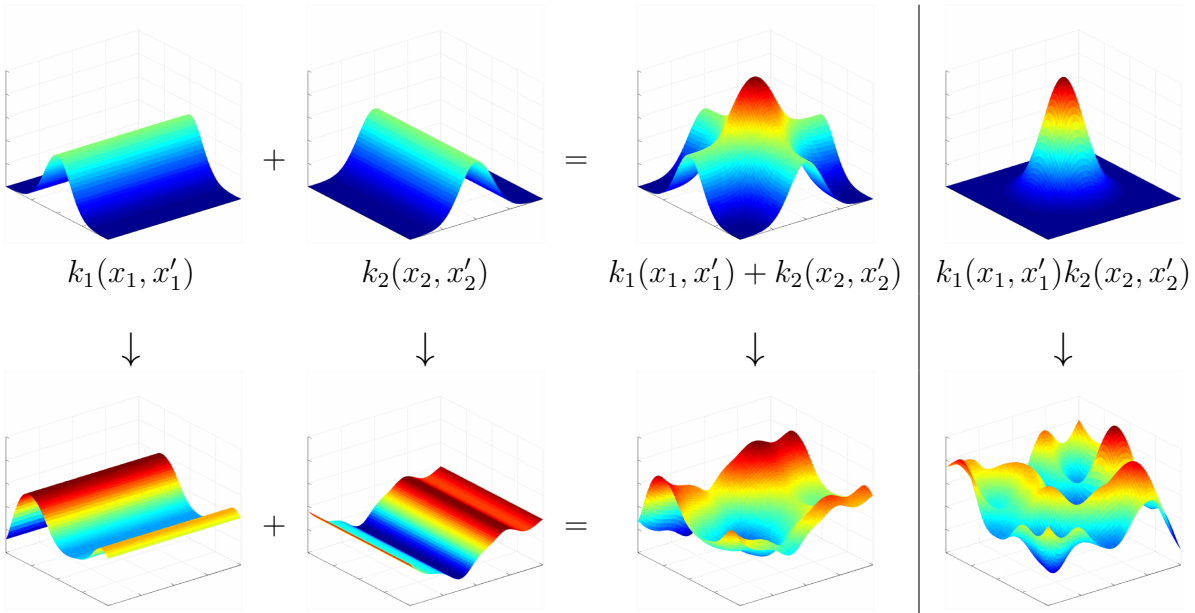


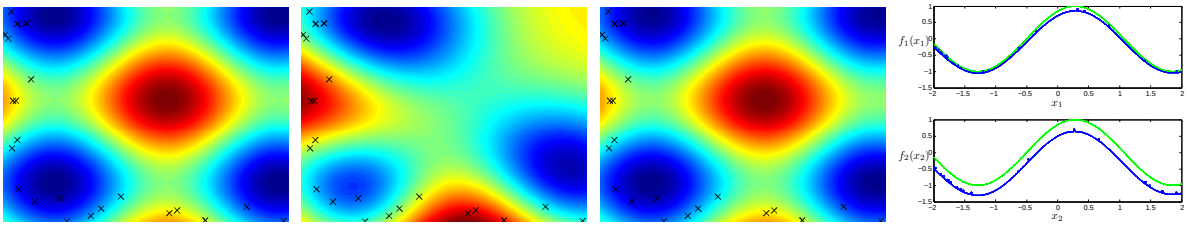
Fig. 2.3 Top left: An additive kernel is simply a sum of kernels. Bottom left: A draw from an additive kernel corresponds to a sum of draws from GPs with the corresponding kernels. Top right: a product kernel is a product of kernels. Bottom right: A draw from a product kernel does not correspond to a product of draws from the corresponding kernels.

2.1.2 Additivity in multiple dimensions

Figure 2.3 compares, for two-dimensional functions, a first-order additive kernel with a second-order kernel.

Long-range extrapolation through additivity

Because additive kernels can discover non-local structure in data, they are exceptionally well-suited to problems where local interpolation fails. Figure 2.4 shows a dataset which



True Function & data locations | Squared-exp GP posterior mean | Additive GP posterior mean | Additive GP 1st-order functions

Fig. 2.4 Long-range inference in functions with additive structure.

demonstrates this feature of additive GPs, consisting of data drawn from a sum of two axis-aligned sine functions. The training set is restricted to a small, L-shaped area; the test set contains a peak far from the training set locations. The additive GP recovered both of the original sine functions (shown in green), and inferred correctly that most of the variance in the function comes from first-order interactions. The ability of additive GPs to discover long-range structure suggests that this model may be well-suited to deal with covariate-shift problems.

2.2 Structure through Multiplication

Multiplying kernels allows us to account for interactions between different input dimensions or different notions of similarity. For instance, in multidimensional data, the multiplicative kernel $\mathrm{SE}_1 \times \mathrm{SE}_3$ represents a smoothly varying function of dimensions 1 and 3 which is not constrained to be additive. In univariate data, multiplying a kernel by SE gives a way of converting global structure to local structure. For example, Per corresponds to globally periodic structure, whereas Per $\times$ SE corresponds to locally periodic structure, as shown in row 1 of figure 2.3.

Many architectures for learning complex functions, such as convolutional networks LeCun et al. (1989) and sum-product networks Poon and Domingos (2011), include units which compute AND-like and OR-like operations. Composite kernels can be viewed in this way too. 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.

From a kernel point of view,
if $k_1(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^\mathsf{T}\phi_1(\mathbf{x}')$ and $k_2 = \phi_2(\mathbf{x})^\mathsf{T}\phi_2(\mathbf{x}')$ then

$$k_1(\mathbf{x}, \mathbf{x}') \times k_2(\mathbf{x}, \mathbf{x}') = [\phi_1(\mathbf{x}) + \phi_2(\mathbf{x})]^\mathsf{T} [\phi_1(\mathbf{x}') + \phi_2(\mathbf{x}')] \tag{2.8}$$

In other words, the features of $k_1 \times k_2$ are the cartesian product (all possible combinations) of the original sets of features.

2.2.1 Multiplying with constant functions

If we wish to model a function that's been multiplied by some fixed function $a(x)$, this can be easily achieved by multiplying the kernel by $a(\mathbf{x})a(\mathbf{x}')$.

This is why it is common practice to assume zero mean, since marginalizing over an unknown mean function can be equivalently expressed as a zero-mean GP with a new kernel.

2.2.2 Signal versus noise

In most derivations of Gaussian processes, the model is given as $y = f + \epsilon$, where $\epsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2_{\text{noise}})$.

However, ϵ can equivalently be thought of as another Gaussian process, and so this model can be written as one in which $f \sim \mathcal{GP}(0, k + \delta)$. The lack of distinction between the noise model and the model of the signal raises the larger question: Which part of a model represents signal, and which represents noise?

Our answer is: *it depends on what you want to do with the model.* For example: often, we don't care about the short-term variations in a function, and only in the long-term

trend. However, in many other cases, we with to de-trend our data to see more clearly how much a particular part of the signal deviated from normal.

Student's t processes

One shortcoming of the

2.3 Expressing Symmetries

When modeling functions, encoding known symmetries greatly aids learning and prediction. We demonstrate that in nonparametric regression, many types of symmetry can be enforced through operations on the covariance function. These symmetries can be composed to produce nonparametric priors on functions whose domains have interesting topological structure such as spheres, torii, and Möbius strips. We demonstrate that marginal likeihood an be used to automatically search over such structures.

Joint work with David Reshef, Roger Grosse, Joshua B. Tenenbaum

It is well-known that the properties of the functions we wish to model can be expressed mainly through the covariance function Rasmussen and Williams (2006).

In this section, we give recipes for expressing several classes of symmetries. Later, we will show how these can be combined to produce more interesting structures.

Periodicity Given D dimensions, we can enforce rotational symmetry on any subset of the dimensions:

$$f(x) = f(x_i + k\tau_i) \quad \forall k \in \mathbb{Z} \tag{2.9}$$

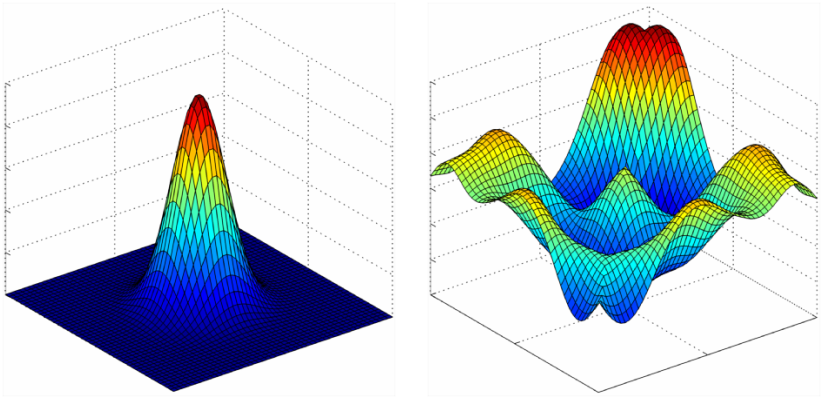
by the applying a kernel between pairs transformed coordinates $\sin(x), \cos(x)$:

$$k_{\text{periodic}}(x, x') = k(\sin(x), \cos(x), \sin(x'), \cos(x')) \tag{2.10}$$

We can also apply rotational symmetry repeatedy to a single dimension.

Reflective Symmetry along an axis we can enforce the symmetry

$$f(x) = f(-x) \tag{2.11}$$



Additive method min method

Fig. 2.5 An illustration of two methods of introducing symmetry: The additive method or the min method. The additive method has half the marginal variance away from $y = x$, but the min method introduces a non-differentiable seam along $y = x$.

by the kernel transform

$$k_{\text{symm arg1}}(x, x') = k(x, x') + k(x, -x') + k(-x, x') + k(-x, -x') \tag{2.12}$$

Reflective Symmetry along a diagonal We can enforce symmetry between any two dimensions:

$$f(x, y) = f(y, x) \tag{2.13}$$

by two methods: In the additive method, we transform the kernel by:

$$k_{\text{reflect add}}(x, y, x', y') = k(x, y, x', y') + k(x, y, y', x') + k(y, x, x', y') + k(y, x, y', x') \tag{2.14}$$

or by

$$k_{\text{reflect min}}(x, y, x', y') = k(\min(x, y), \max(x, y), \min(x', y'), \max(x', y')) \tag{2.15}$$

however, the second method will in general lead to non-differentiability along $x = y$. Figure 2.5 shows the difference.

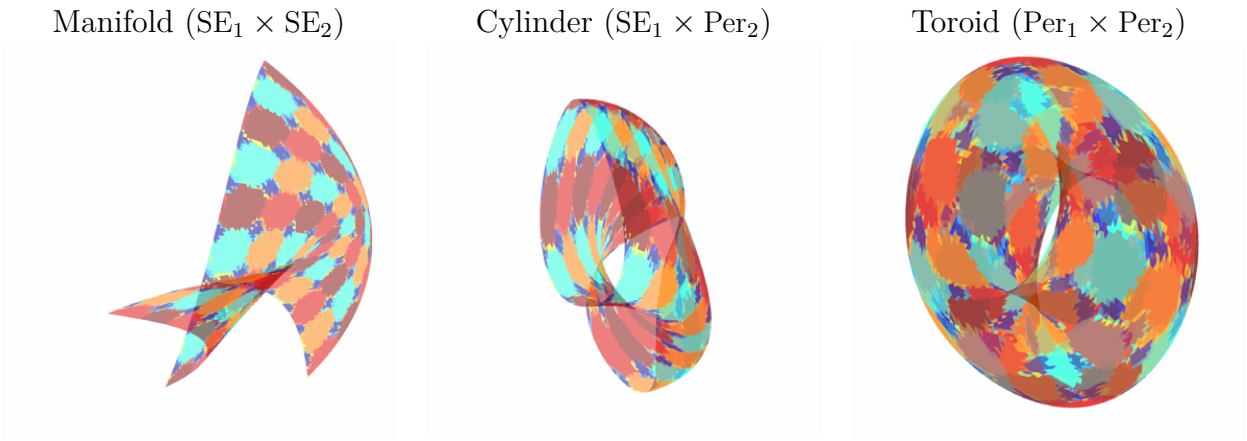


Fig. 2.6 Generating 2D manifolds with different topological structures. By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, the surfaces created have the corresponding topologies, ignoring self-intersections.

2.3.1 Parametric embeddings

In general, we can always enforce the symmetries obeyed by a given surface by finding a parametric embedding to that surface. However, it is not clear how to do this in general without introducing unnecessary

2.4 How to generate 3D shapes with a given topology

First create a mesh in 2d. Then draw 3 independent functions from a GP prior with the relevant symmetries encoded in the kernel. Then, map the 2d points making up the mesh through those 3 functions to get the 3D coordinates of each point on the mesh.

This is similar in spirit to the GP-LVM model Lawrence (2005), which learns an embedding of the data into a low-dimensional space, and constructs a fixed kernel structure over that space.

Function on Möbius strip Möbius manifold Sudanese Möbius strip

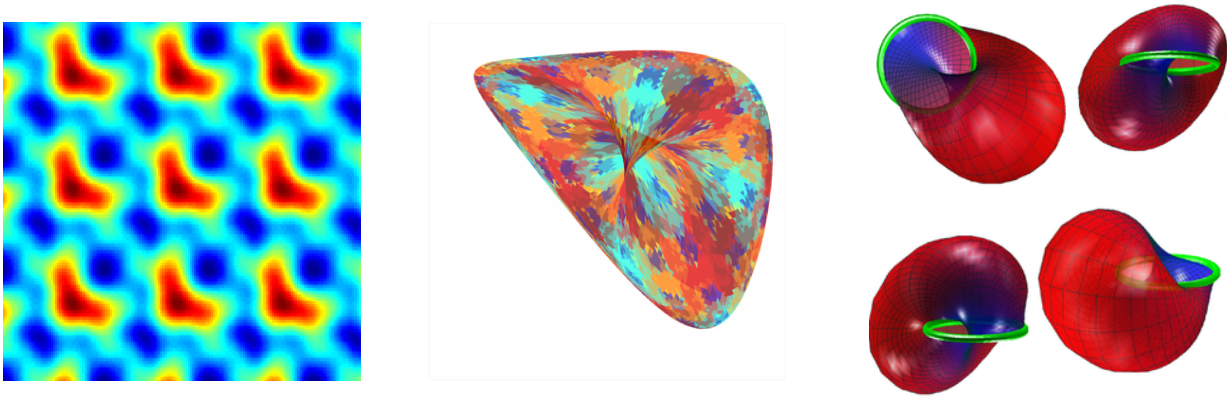


Fig. 2.7 Generating Möbius strips. By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, the surfaces created have topology corresponding to a Möbius strip. TODO: Talk about Sudanese representation.

2.4.1 Möbius strips

A prior on functions on Möbius strips can be achieved by enforcing the symmetries:

$$f(x, y) = f(x, y + \tau_y) \tag{2.16}$$

$$f(x, y) = f(x + \tau_x, y) \tag{2.17}$$

$$f(x, y) = f(y, x) \tag{2.18}$$

If we imagine moving along the edge of a Möbius strip, that is equivalent to moving along a diagonal in the function generated. Figure 2.7 shows this. The second example is doesn't resemble a typical Möbius strip because the edge of the mobius strip is in a geometric circle. This kind of embedding is resembles the Sudanese Möbius strip [cite].

Another classic example of a function living on a Mobius strip is the auditory quality of 2-note intervals. The harmony of a pair of notes is periodic (over octaves) for each note, and the

2.5 Discrete data

Kernels can be defined over all types of data structures: Text, images, matrices, and even kernels . Coming up with a kernel on a new type of data used to be an easy way to get a NIPS paper.

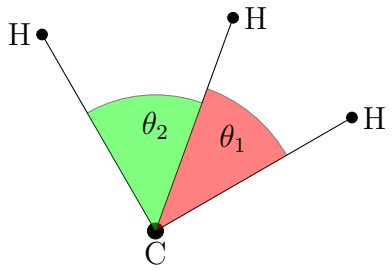


Fig. 2.8 An example of a function expressing the same symmetries as a Möbius strip in two of its arguments. The energy of a molecular configuration $f(\theta_1, \theta_2)$ depends only on the relative angles between atoms, and because each atom is indistiguishable, is invariant to permuting the atoms.

2.5.1 Categorical variables

There is a simple way to do GP regression over categorical variables. Simply represent your categorical variable as a by a one-of-k encoding. This means that if your number ranges from 1 to 5, represent that as 5 different data dimensions, only one of which is on at a time.

Then, simply put a product of SE kernels on those dimensions. This is the same as putting one SE ARD kernel on all of them. The lengthscale hyperparameter will now encode whether, when that coding is active, the rest of the function changes. If you notice that the estimated lengthscales for your categorical variables is short, your model is saying that it's not sharing any information between data of different categories.

2.6 Examples

2.6.1 Computing molecular energies

Figure 2.8 gives one example of a function which obeys the same symmetries as a Möbius strip, in some subsets of its arguments.

2.6.2 Translation invariance in images

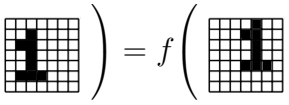
Most models of images are invariant to spatial translations [cite convolution nets]. Similarily, most models of sounds are also invariant to translation through time.

Note that this sort of translational invariance is completely distinct from the stationarity properties of kernels used in Gaussian process priors. A stationary kernel implies

that the prior is invariant to translations of the entire training and test set.

We are discussing here a discretized input space (into pixels or the audio equivalent), where the input vectors have one dimension for every pixel. We are interested in creating priors on functions that are invariant to shifting a signal along its pixels:

$$f\left(\quad\right) = f\left(\quad\right) \tag{2.19}$$



Translational invariance in this setting is equivalent to symmetries between dimensions in the input space.

This prior can be achieved in one dimension by using the following kernel transformation:

$$k\left((x_1, x_2, \ldots, x_D), (x'_1, x'_2, \ldots, x'_D)\right) = \sum_{i=1}^{D} \prod_{j=1}^{D} k(x_j, x'_{i+j \bmod D}) \tag{2.20}$$

Edge effects can be handled either by wrapping the image around, or by padding it with zeros.

Convolution The resulting kernel could be called a *discrete convolution kernel*. For an image with R, C rows and columns, it can also be written as:

$$k_{\text{conv}}\left((x_{11}, x_{12}, \ldots, x_{RC}), (x'_{11}, x'_{12}, \ldots, x'_{RC})\right) = \sum_{i=-L}^{L} \sum_{j=-L}^{L} k\left(\mathbf{x}, T_{ij}(\mathbf{x}')\right) \tag{2.21}$$

where $T_{ij}(\mathbf{x})$ is the operator which replaces each x_{mn} with $x_{m+i,n+j}$. Thus we are simply defining the covariance between two images to be the sum of all covariances between all relative translations of the two images. We can also normalize the kernel by pre-multiplying it with $\sqrt{k_{\text{conv}}(\mathbf{x}, \mathbf{x}) k_{\text{conv}}(\mathbf{x}', \mathbf{x}')}$.

Is there a pathology of the additive construction that appears in the limit?

2.6.3 Max-pooling

What we'd really like to do is a max-pooling operation. However, in general, a kernel which is the max of other kernels is not PSD [put counterexample here?]. Is the max over co-ordinate switching PSD?

2.7 Related Work

Invariances in Gaussian processes Ginsbourger et al. (2013) show that, for Gaussian processes, with probability one, $f(\mathbf{x}) = f(T(\mathbf{x}))$ if and only if $k(x, x') = k(x, T(x'))$.

2.8 Deep kernels

? showed that kernel machines have limited generalization ability when they use a local kernel such as the squared-exp. However, many interesting non-local kernels can be constructed which allow non-trivial extrapolation. For example, periodic kernels can be viewed as a 2-layer-deep kernel, in which the first layer maps $x \to [\sin(x), \cos(x)]$, and the second layer maps through basis functions corresponding to the SE kernel.

Can we construct other useful kernels by composing fixed feature maps several times, creating deep kernels? Cho (2012) constructed kernels of this form, repeatedly applying multiple layers of feature mappings. We can compose the feature mapping of two kernels:

$$k_1(\mathbf{x}, \mathbf{x}') = \mathbf{h}_1(\mathbf{x})^\mathsf{T} \mathbf{h}_1(\mathbf{x}') \tag{2.22}$$

$$k_2(\mathbf{x}, \mathbf{x}') = \mathbf{h}_2(\mathbf{x})^\mathsf{T} \mathbf{h}_2(\mathbf{x}') \tag{2.23}$$

$$(k_1 \circ k_2)(\mathbf{x}, \mathbf{x}') = k_2(\mathbf{h}_1(\mathbf{x}), \mathbf{h}_1(\mathbf{x}')) \tag{2.24}$$

$$= [\mathbf{h}_2(\mathbf{h}_1(\mathbf{x}))]^\mathsf{T} \mathbf{h}_2(\mathbf{h}_1(\mathbf{x}')) \tag{2.25}$$

Composing the squared-exp kernel with any implicit mapping $\mathbf{h}(\mathbf{x})$ has a simple closed form:

$$\begin{aligned} k_{L+1}(\mathbf{x}, \mathbf{x}') &= k_{SE}(\mathbf{h}(\mathbf{x}), \mathbf{h}(\mathbf{x}')) = \\ &= \exp\left(-\frac{1}{2}||\mathbf{h}(\mathbf{x}) - \mathbf{h}(\mathbf{x}')||_2^2\right) \\ &= \exp\left(-\frac{1}{2}\left[\mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}) - 2\mathbf{h}(\mathbf{x})^\mathsf{T}\mathbf{h}(\mathbf{x}') + \mathbf{h}(\mathbf{x}')^\mathsf{T}\mathbf{h}(\mathbf{x}')\right]\right) \\ &= \exp\left(-\frac{1}{2}\left[k_L(\mathbf{x}, \mathbf{x}) - 2k_L(\mathbf{x}, \mathbf{x}') + k_L(\mathbf{x}', \mathbf{x}')\right]\right) \end{aligned} \tag{2.26}$$

Thus, we can express k_{L+1} exactly in terms of k_L.

Infinitely deep kernels What happens when we repeat this composition of feature maps many times, starting with the squared-exp kernel? In the infinite limit, this recursion converges to $k(\mathbf{x}, \mathbf{x}') = 1$ for all pairs of inputs, which corresponds to a prior

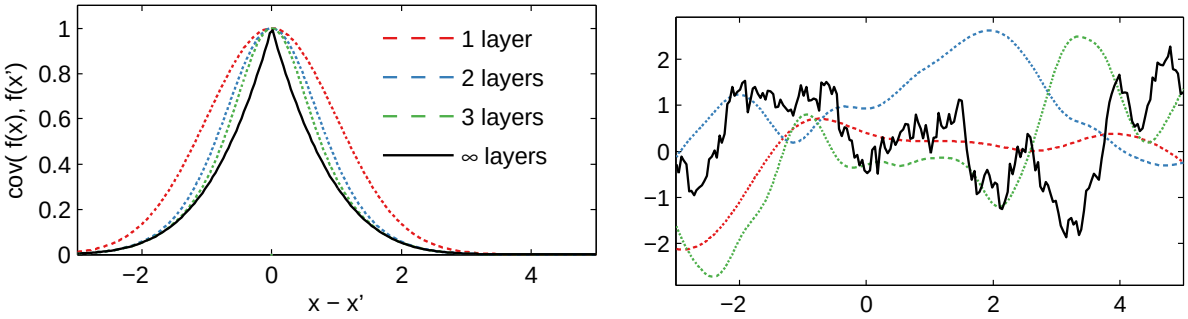


Fig. 2.9 Left: Input-connected deep kernels. By connecting the inputs $\mathbf{x}$ to each layer, the kernel can still depend on its input even after arbitrarily many layers of computation. Right: GP draws using deep input-connected kernels.

on constant functions $f(\mathbf{x}) = c$.

A non-degenerate construction As before, we can overcome this degeneracy by connecting the inputs $\mathbf{x}$ to each layer. To do so, we simply augment the feature vector $\mathbf{h}_L(\mathbf{x})$ with $\mathbf{x}$ at each layer:

$$\begin{aligned} k_{L+1}(\mathbf{x}, \mathbf{x}') &= \exp\left(-\frac{1}{2}\left|\left|\begin{bmatrix}\mathbf{h}_L(\mathbf{x}) \\ \mathbf{x}\end{bmatrix} - \begin{bmatrix}\mathbf{h}_L(\mathbf{x}') \\ \mathbf{x}'\end{bmatrix}\right|\right|_2^2\right) \\ &= \exp\left(-\frac{1}{2}\Big[k_L(\mathbf{x},\mathbf{x}) - 2k_L(\mathbf{x},\mathbf{x}') + k_L(\mathbf{x}',\mathbf{x}') - ||\mathbf{x}-\mathbf{x}'||_2^2\Big]\right) \end{aligned} \tag{2.27}$$

For the SE kernel, this repeated mapping satisfies

$$k_\infty(\mathbf{x}, \mathbf{x}') - \log\left(k_\infty(\mathbf{x}, \mathbf{x}')\right) = 1 + \frac{1}{2}||\mathbf{x} - \mathbf{x}'||_2^2 \tag{2.28}$$

The solution to this recurrence has no closed form, but has a similar shape to the Ornstein-Uhlenbeck covariance $k_{\text{OU}}(x, x') = \exp(-|x - x'|)$ with lighter tails. Samples from a GP prior with this kernel are not differentiable, and are locally fractal.

2.8.1 When are deep kernels useful models?

Kernels correspond to fixed feature maps, and so kernel learning is an example of implicit representation learning. Such feature maps can capture rich structure (Duvenaud et al., 2013), and can enable many types of generalization, such as translation and rotation invariance in images (Kondor, 2008). **?** used a deep neural network to learn feature

transforms for kernels, which learn invariances in an unsupervised manner. The relatively uninteresting properties of the kernels derived in this section simply reflect the fact that an arbitrary deep computation is not usually a useful representation, unless combined with learning.

2.9 Worked example: building a structured kernel for a time-series

2.9.1 Modeling multiple periodicities

2.9.2 Incoportating side-information

2.9.3 Incoportating discrete covariates

2.9.4 Breaking down the predictions, examining different parts of the model

Chapter 3

Automatically Building Structured Covariance Functions

> "It would be very nice to have a formal apparatus that gives us some 'optimal' way of recognizing unusual phenomena and inventing new classes of hypotheses that are most likely to contain the true one; but this remains an art for the creative human mind."
>
> E. T. Jaynes, 1985

Despite its importance, choosing the structural form of the kernel in nonparametric regression remains a black art. We define a space of kernel structures which are built compositionally by adding and multiplying a small number of base kernels. We present a method for searching over this space of structures which mirrors the scientific discovery process. The learned structures can often decompose functions into interpretable components and enable long-range extrapolation on time-series datasets. Our structure search method outperforms many widely used kernels and kernel combination methods on a variety of prediction tasks.

This paper presents the beginnings of an automatic statistician, focusing on regression problems. Our system explores an open-ended space of possible statistical models to discover a good explanation of the data, and then produces a detailed report with figures and natural-language text.

Our approach treats unknown functions nonparametrically using Gaussian processes, which has two important consequences. First, Gaussian processes model functions in terms of high-level properties (e.g. smoothness, trends, periodicity, changepoints). Taken together with the compositional structure of our language of models, this allows us to

automatically describe functions through a decomposition into additive parts. Second, the use of flexible nonparametric models and a rich language for composing them in an open-ended manner also results in state-of-the-art extrapolation performance evaluated over 13 real time series data sets from various domains.

3.0.5 Attribution

Joint work with James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum and Zoubin Ghahramani.

3.1 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 generalization.

Most existing techniques pose kernel learning as a (possibly high-dimensional) parameter estimation problem. Examples include learning hyperparameters (Rasmussen and Williams, 2006), linear combinations of fixed kernels Bach (2009), and mappings from the input space to an embedding space Salakhutdinov and Hinton (2008).

However, to apply existing kernel learning algorithms, the user must specify the parametric form of the kernel, and this can require considerable expertise, as well as trial and error.

To make kernel learning more generally applicable, we reframe the kernel learning problem as one of structure discovery, and automate the choice of kernel form. In particular, we formulate a space of kernel structures defined compositionally in terms of sums and products of a small number of base kernel structures. This provides an expressive modeling language which concisely captures many widely used techniques for constructing kernels. We focus on Gaussian process regression, where the kernel specifies a covariance function, because the Bayesian framework is a convenient way to formalize structure discovery. 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 marginal

likelihood as the search criterion.

We found that our structure discovery 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 paper will help replace the current and often opaque art of kernel engineering with a more transparent science of automated kernel construction.

Automating the process of statistical modeling would have a tremendous impact on fields that currently rely on expert statisticians, machine learning researchers, and data scientists. While fitting simple models (such as linear regression) is largely automated by standard software packages, there has been little work on the automatic construction of flexible but interpretable models. What are the ingredients required for an artificial intelligence system to be able to perform statistical modeling automatically? In this paper we conjecture that the following ingredients may be useful for building an AI system for statistics, and we develop a working system which incorporates them:

- **An open-ended language of models** expressive enough to capture many of the modeling assumptions and model composition techniques applied by human statisticians to capture real-world phenomena

- **A search procedure** to efficiently explore the space of models spanned by the language

- **A principled method for evaluating models** in terms of their complexity and their degree of fit to the data

- **A procedure for automatically generating reports** which explain and visualize different factors underlying the data, make the chosen modeling assumptions explicit, and quantify how each component improves the predictive power of the model

In this paper, we introduce a system for modeling time-series data containing the above ingredients which we call the Automatic Bayesian Covariance Discovery (ABCD) system. The system defines an open-ended language of Gaussian process models via 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.

We show examples of automatically generated reports which highlight interpretable features discovered in a variety of data sets (e.g. figure 4.4). The supplementary material to this paper includes 13 complete reports automatically generated by ABCD.

Good statistical modeling requires not only interpretability but predictive accuracy. We compare ABCD against existing model construction techniques in terms of predictive performance at extrapolation, and we find state-of-the-art performance on 13 time series. In the remainder of this paper we describe the components of ABCD in detail.

3.2 A language of regression models

As discussed above, we can construct a wide variety of kernel structures compositionally by adding and multiplying a small number of base kernels. In particular, we consider the four base kernel families discussed in Section **??**: SE, Per, Lin, and RQ. Any algebraic expression combining these kernels using the operations $+$ and $\times$ defines a kernel family, whose parameters are the concatenation of the parameters for the base kernel families.

Our search procedure begins by proposing all base kernel families applied to all input dimensions. We allow the following search operators over our set of expressions:

(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 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 context-free grammar 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. In particular,

- One can look for structure, e.g. periodicity, in the residuals of a model, and then extend the model to capture that structure. This corresponds to applying rule (1).
- One can start with structure, e.g. linearity, which is assumed to hold globally, but find that it only holds locally. This corresponds to applying rule (2) to obtain the structure shown in rows 1 and 3 of figure 2.3.
- 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.

Scoring kernel families Choosing kernel structures requires a criterion for evaluating structures. We choose marginal likelihood as our criterion, since it balances the fit and complexity of a model (Rasmussen and Ghahramani, 2001). Conditioned on kernel parameters, the marginal likelihood of a GP can be computed analytically. However, to evaluate a kernel family we must integrate over kernel parameters. We approximate this intractable integral with the Bayesian information criterion (Schwarz, 1978) after first optimizing to find the maximum-likelihood kernel parameters.

Unfortunately, optimizing over parameters is not a convex optimization 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 provide reasonable initializations: all of the parameters which were part of the previous kernel are initialized to their previous values. All parameters are then optimized using conjugate gradients, randomly restarting the newly introduced parameters. This procedure is not guaranteed to find the global optimum, but it implements the commonly used heuristic of iteratively modeling residuals.

The general problem of regression consists of learning a function f mapping from some input space $\mathcal{X}$ to some output space $\mathcal{Y}$. We would like an expressive language which can represent both simple parametric forms of f such as linear, polynomial, etc. and also complex nonparametric functions specified in terms of properties such as smoothness,

periodicity, etc. Fortunately, Gaussian processes (GPs) provide a very general and analytically tractable way of capturing both simple and complex functions.

Gaussian processes are distributions over functions such that any finite subset of function evaluations, $(f(x_1), f(x_2), \ldots f(x_N))$, have a joint Gaussian distribution (Rasmussen and Williams, 2006). A GP is completely specified by its mean function, $\mu(x) = \mathbb{E}(f(x))$ and kernel (or covariance) function $k(x, x') = \mathrm{Cov}(f(x), f(x'))$. It is common practice to assume zero mean, since marginalizing over an unknown mean function can be equivalently expressed as a zero-mean GP with a new kernel. The structure of the kernel captures high-level properties of the unknown function, f, which in turn determines how the model generalizes or extrapolates to new data. We can therefore define a language of regression models by specifying a language of kernels.

The elements of this language are a set of base kernels capturing different function properties, and a set of composition rules which combine kernels to yield other valid kernels. Our base kernels are white noise (WN), constant (C), linear (Lin), squared exponential (SE) and periodic (Per), which on their own encode for uncorrelated noise, constant functions, linear functions, smooth functions and periodic functions respectively[1]. The composition rules are addition and multiplication:

$$k_1 + k_2 = k_1(x, x') + k_2(x, x') \tag{3.1}$$

$$k_1 \times k_2 = k_1(x, x') \times k_2(x, x') \tag{3.2}$$

Combining kernels using these operations can yield kernels encoding for richer structures such as approximate periodicity (SE $\times$ Per) or smooth functions with linear trends (SE + Lin).

We have found that incorporating changepoints into the language is essential for realistic models of time series (e.g. figure 4.4). Changepoints can be defined through addition and multiplication with sigmoidal functions:

$$\mathrm{CP}(k_1, k_2) = k_1 \times \boldsymbol{\sigma} + k_2 \times \bar{\boldsymbol{\sigma}} \tag{3.3}$$

where $\boldsymbol{\sigma} = \sigma(x)\sigma(x')$ and $\bar{\boldsymbol{\sigma}} = (1 - \sigma(x))(1 - \sigma(x'))$. Changewindows $\mathrm{CW}(\cdot, \cdot)$ can be defined similarly by replacing $\sigma(x)$ with a product of two sigmoids.

We also expanded and reparametrised the set of base kernels so that they were more amenable to automatic description and to extend the number of common regression

[1] Definitions of kernels are in the supplementary material.

models included in the language. Table 3.1 lists common regression models that can be expressed by our language.

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	$\sum \mathrm{SE} + \sum \mathrm{Per} + \mathrm{WN}$
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 3.1 Common regression models expressible in our language. cos is a special case of our reparametrised Per.

3.3 Model Search and Evaluation

We explore the space of regression models using a greedy search. We use the same search operators, but also include additional operators to incorporate changepoints; a complete list is contained in the supplementary material.

After each model is proposed its kernel parameters are optimised by conjugate gradient descent. We evaluate each optimized model, M, using the Bayesian Information Criterion (BIC) (Schwarz, 1978):

$$\mathrm{BIC}(M) = -2 \log p(D \mid M) + p \log n \tag{3.4}$$

where p is the number of kernel parameters, $\log p(D|M)$ is the marginal likelihood of the data, D, and n is the number of data points. BIC trades off model fit and complexity and implements what is known as "Bayesian Occam's Razor" (e.g. MacKay, 2003; Rasmussen and Ghahramani, 2001).

3.4 Structure discovery in time series

To investigate our method's ability to discover structure, we ran the kernel search on several time-series.

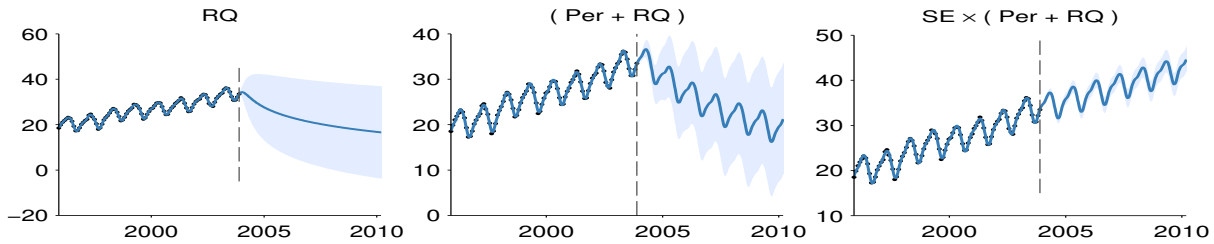


Fig. 3.1 Posterior mean and variance for different depths of kernel search. The dashed line marks the extent of the dataset. In the first column, the function is only modeled as a locally smooth function, and the extrapolation is poor. Next, a periodic component is added, and the extrapolation improves. At depth 3, the kernel can capture most of the relevant structure, and is able to extrapolate reasonably.

As discussed in section 2, a GP whose kernel is a sum of kernels can be viewed as a sum of functions drawn from component GPs. This provides another method of visualizing the learned structures. In particular, all kernels in our search space can be equivalently written as sums of products of base kernels by applying distributivity. For example,

$$\mathrm{SE} \times (\mathrm{RQ} + \mathrm{Lin}) = \mathrm{SE} \times \mathrm{RQ} + \mathrm{SE} \times \mathrm{Lin} \tag{3.5}$$

We visualize the decompositions into sums of components using the formulae given in the appendix. The search was run to depth 10, using the base kernels from Section **??**.

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

Figure 3.4 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 3.2 shows the final model chosen by our method, together with its decomposition into additive components. The final model exhibits both plausible extrapolation and interpretable components: a long-term trend, annual periodicity and medium-term 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.

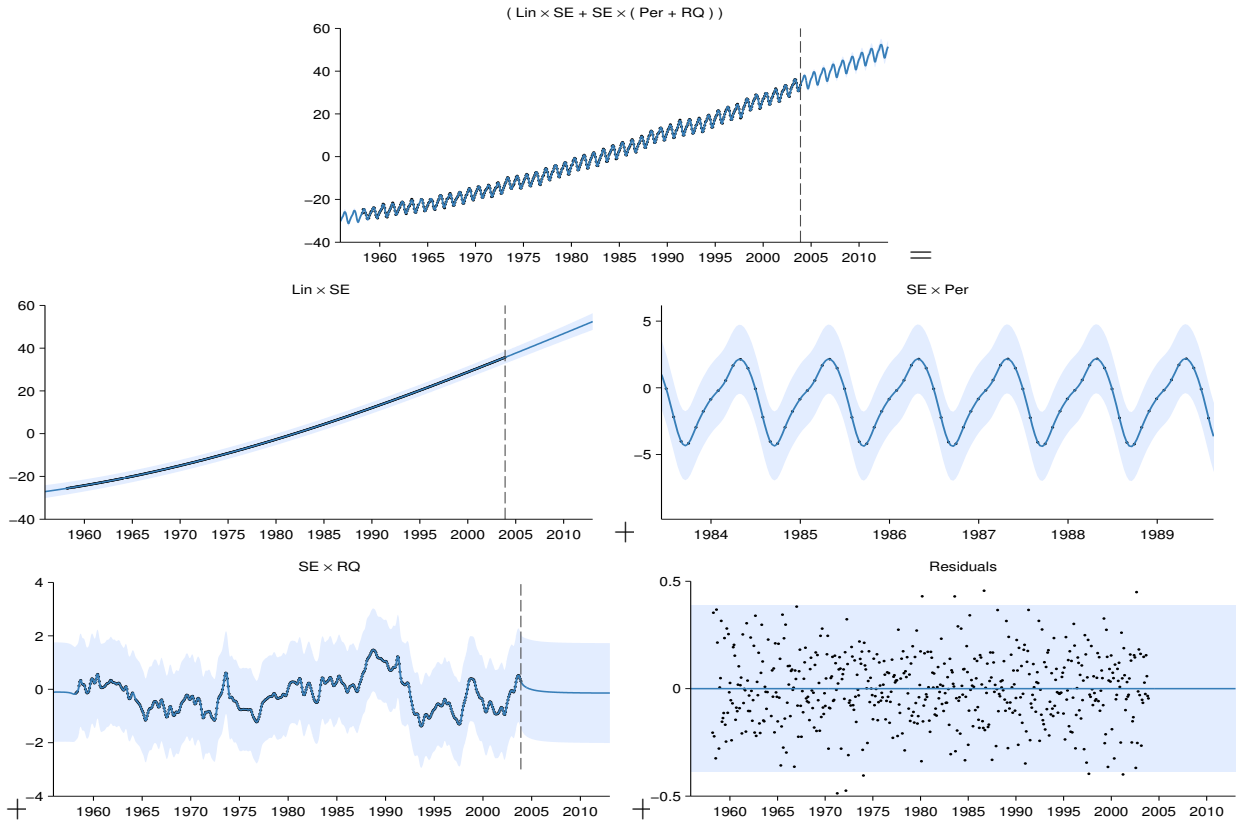


Fig. 3.2 First row: The posterior on the Mauna Loa dataset, after a search of depth 10. Subsequent rows show the automatic decomposition of the time series. The decompositions shows long-term, yearly periodic, medium-term anomaly components, and residuals, respectively. In the third row, the scale has been changed in order to clearly show the yearly periodic structure.

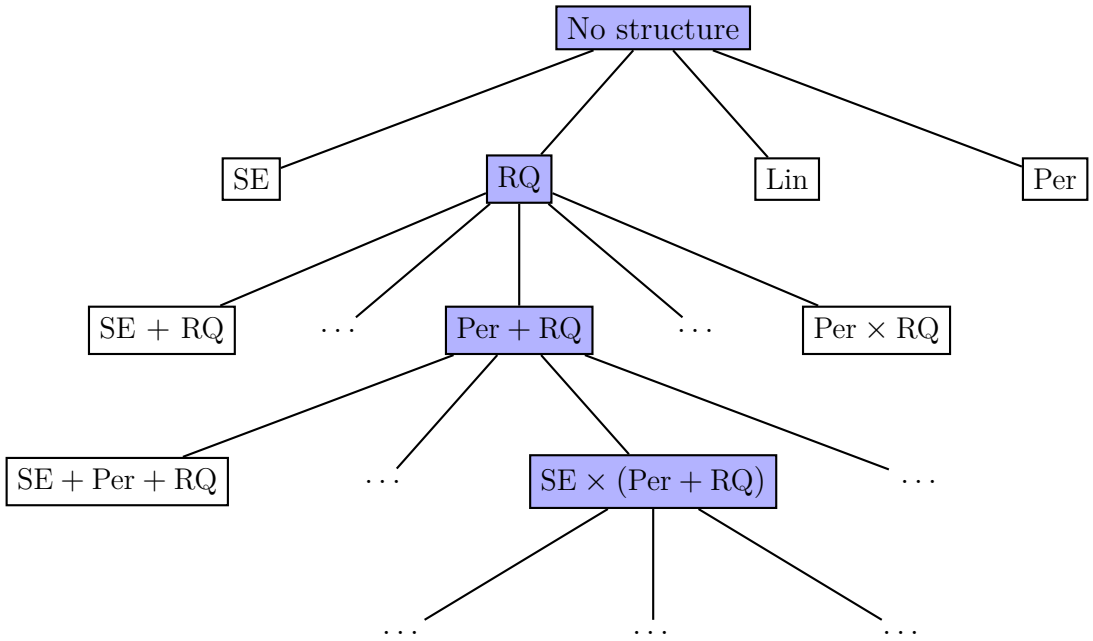


Fig. 3.3 An example of a search tree over kernel expressions. Figure 3.4 shows the model increasing in sophistication as the kernel expression grows.

Airline passenger data Figure 3.6 shows the decomposition produced by applying our method to monthly totals of international airline passengers (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 analyzed annual solar irradiation data from 1610 to 2011 (Lean et al., 1995). The posterior and residuals of the learned kernel are shown in figure 4.2. 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.

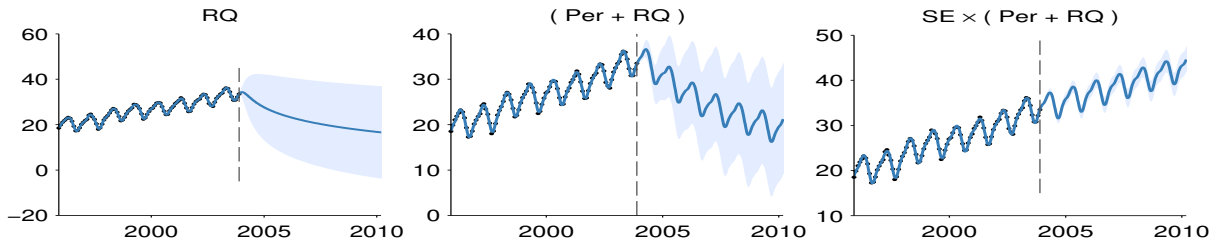


Fig. 3.4 Posterior mean and variance for different depths of kernel search. The dashed line marks the extent of the dataset. In the first column, the function is only modeled as a locally smooth function, and the extrapolation is poor. Next, a periodic component is added, and the extrapolation improves. At depth 3, the kernel can capture most of the relevant structure, and is able to extrapolate reasonably.

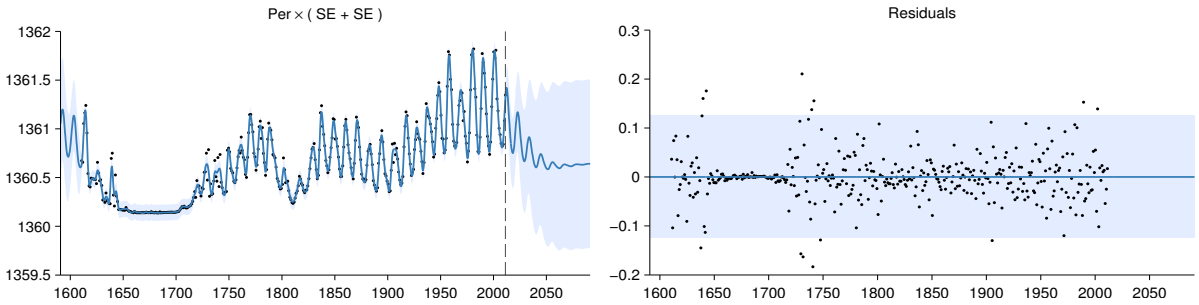


Fig. 3.5 Full posterior and residuals on the solar irradiance dataset.

3.5 Related work

3.5.1 Nonparametric regression in high dimensions

Nonparametric regression methods such as 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 generalize 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. Generalized 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 generalizes well. 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. 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). This model is a linear combinations of splines along each dimension, all pairs of dimensions, and possibly higher-order combinations. Because 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 an SE kernel). In our approach, this can be represented as a sum of SE and Lin.

3.5.2 Kernel learning

There is a large body of work attempting to construct a rich kernel through 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 hyperparameters, must be specified in advance.

Another approach to kernel learning is to learn an embedding of the data points. 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 generalizing 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 f(x).

Wilson and Adams (2013) derive kernels of the form SE $\times \cos(x - x')$, forming a basis for stationary kernels. These kernels share similarities with SE $\times$ Per but can express negative prior correlation, and could usefully be included in our grammar.

Diosan et al. (2007) and Bing et al. (2010) learn composite kernels for support vector machines and relevance vector machines, using genetic search algorithms. Our work

employs a Bayesian search criterion, and goes beyond this prior work by demonstrating the interpretability of the structure implied by composite kernels, and how such structure allows for extrapolation.

3.5.3 Structure discovery

There have been several attempts to uncover the structural form of a dataset by searching over a grammar of structures. For example, **?**, Todorovski and Dzeroski (1997) and Washio et al. (1999) attempt to learn parametric forms of equations to describe time series, or relations between quantities. Because we learn expressions describing the covariance structure rather than the functions themselves, we are able to capture structure which does 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 continous 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.

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 discover such structure automatically from data. Our work is tackling a similar problem, but in a supervised setting.

3.5.4 Building Kernel Functions

Rasmussen and Williams (2006) devote 4 pages to manually constructing a composite kernel to model a time series of carbon dioxode concentrations. In the supplementary material, we include a report automatically generated by ABCD for 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 (2012) and Lloyd (2013).

Bing et al. (2010); Diosan et al. (2007) and Kronberger and Kommenda (2013) search over a similar space of models as ABCD using genetic algorithms but do not interpret the resulting models.

3.5.5 Kernel Learning

Sparse spectrum GPs (Lázaro-Gredilla et al., 2010) approximate the spectral density of a stationary kernel function using delta functions which 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, 1959), that these kernels can approximate any stationary covariance function. Our language of kernels includes both of these kernel classes (see table 3.1).

There is a large body of work attempting to construct rich kernels through a weighted sum of base kernels called multiple kernel learning (MKL) (e.g. Bach et al., 2004). These approaches find the optimal solution in polynomial time but only if the component kernels and parameters are pre-specified. We compare to a Bayesian variant of MKL in section 3.6 which is expressed as a restriction of our language of kernels.

3.5.6 Equation learning

Todorovski and Dzeroski (1997), Washio et al. (1999) and **?** learn parametric forms of functions specifying time series, or relations between quantities. In contrast, ABCD learns a parametric form for the covariance, allowing it to model functions without a simple parametric form.

3.5.7 Searching over open-ended model spaces

This work was inspired by previous successes at searching over open-ended model spaces: matrix decompositions (Grosse et al., 2012) and graph structures (Kemp and Tenenbaum, 2008). In both cases, the model spaces were defined compositionally through a handful of components and operators, and models were selected using criteria which trade off model complexity and goodness of fit. Our work differs in that our procedure automatically interprets the chosen model, making the results accessible to non-experts.

3.5.8 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 video interpretation (Barbu et al., 2012) and automated theorem proving (Ganesalingam and Gowers, 2013).

3.6 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.

3.6.1 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, Accessed summer 2013); plots of the data can be found at the beginning of the reports in the supplementary material.

3.6.2 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, 2013; 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).

We use the default mean absolute error criterion when using Eureqa. All other algorithms can be expressed as restrictions of our modeling language (see table 3.1) so we perform inference using the same search methodology and selection criterion[2] 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 modeling 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., 2013). Constructing a language for this class of time-series model would be an interesting area for future research.

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

3.6.3 Interpretability versus accuracy

BIC trades off model fit and complexity by penalizing the number of parameters in a kernel expression. This can result in ABCD favoring 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 interpretable. We experimented with distributing all products over addition during the search, causing models with many additive components to be more heavily penalized by BIC. We call this procedure ABCD-interpretability, in contrast to the unrestricted version of the search, ABCD-accuracy.

3.6.4 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 3.7 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 outperforms 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 the supplementary material).

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. The results are similar to those for extrapolation and are included in the supplementary material.

3.7 Quantitative evaluation

In addition to the qualitative evaluation in section 3.4, we investigated quantitatively how our method performs on both extrapolation and interpolation tasks.

3.7.1 Extrapolation

We compared the extrapolation capabilities of our model against standard baselines[3]. 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 3.8 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 structure. 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.

3.7.2 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 [4] comparing 5 methods in terms of MSE and predictive likelihood. Our structure search was run up to depth 10, using the SE and RQ base kernel families.

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 table 5.2. Our method outperformed the next-best method in each test, although not substantially.

[3] In one dimension, the predictive means of all baseline methods in table 5.2 are identical to that of a GP with an SE kernel.

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

3.8 Conclusion

Towards the goal of automating statistical modeling 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 state-of-the-art compared to existing model construction techniques. We believe this procedure has the potential to make powerful statistical model-building techniques accessible to non-experts.

3.9 Validation on synthetic data

Table 3.2 Kernels chosen by our method on synthetic data generated using known kernel structures. D denotes the dimension of the functions being modeled. SNR indicates the signal-to-noise ratio. Dashes - indicate no structure.

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	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \ldots$ $(\mathrm{SE}_3 \times \mathrm{Lin}_3 \times \mathrm{Lin}_1 + \mathrm{SE}_4)$	$(\mathrm{SE}_1 + \mathrm{SE}_2) \times \ldots$ $\mathrm{SE}_3 \times \mathrm{SE}_4$	-

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 prior. We then added i.i.d. Gaussian noise to the functions, at various signal-to-noise ratios (SNR).

Table 3.2 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 modeling 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.

Table 3.3 Comparison of multidimensional regression performance. 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%.

	Mean Squared Error (MSE)					Negative Log-Likelihood				
Method	bach	concrete	puma	servo	housing	bach	concrete	puma	servo	housing
Linear Regression	1.031	0.404	0.641	0.523	0.289	2.430	1.403	1.881	1.678	1.052
GAM	1.259	0.149	0.598	0.281	0.161	1.708	0.467	1.195	0.800	0.457
HKL	**0.199**	0.147	0.346	0.199	0.151	-	-	-	-	-
GP SE-ARD	**0.045**	0.157	**0.317**	**0.126**	**0.092**	**−0.131**	0.398	**0.843**	0.429	0.207
Additive GP	**0.045**	**0.089**	**0.316**	**0.110**	0.102	**−0.131**	**0.114**	**0.841**	0.309	0.194
Search - SE, RQ	**0.044**	**0.087**	**0.315**	**0.102**	**0.082**	**−0.141**	**0.065**	**0.840**	0.265	**0.059**
Search - All kernels	**0.509**	**0.079**	**0.321**	**0.094**	**0.112**	**0.357**	**0.114**	**0.837**	**−0.427**	**0.151**

3.10 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 included in this space includes many standard regression models. We proposed a search procedure for this space of kernels which parallels the process of scientific discovery.

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 model-checking by humans. We believe that a data-driven approach to choosing kernel structures automatically can help make nonparametric regression and classification methods accessible to non-experts.

[5]

[5] All GP parameter optimisation was performed by automated calls to the GPML toolbox available at http://www.gaussianprocess.org/gpml/code/.

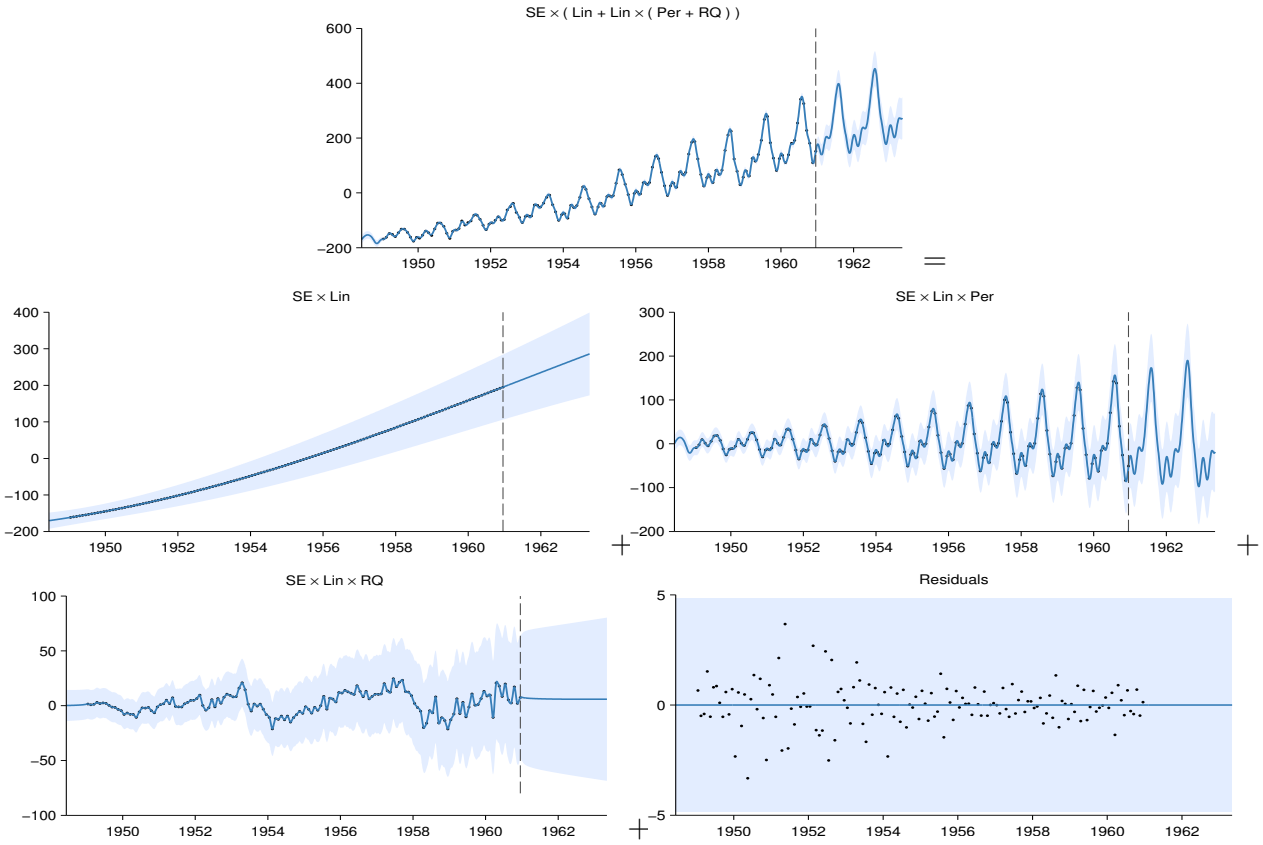


Fig. 3.6 First row: The airline dataset and posterior after a search of depth 10. Subsequent rows: Additive decomposition of posterior into long-term smooth trend, yearly variation, and short-term deviations. Due to the linear kernel, the marginal variance grows over time, making this a heteroskedastic model.

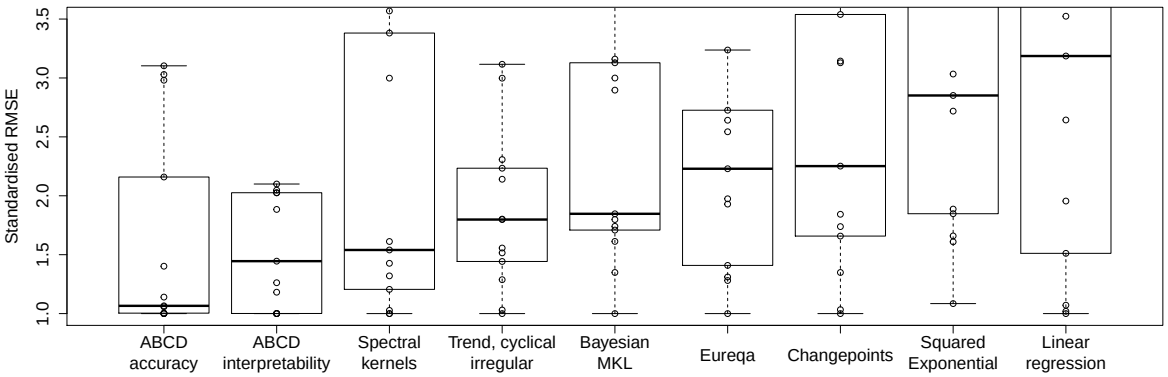


Fig. 3.7 Raw data, and box plot (showing median and quartiles) of standardised extrapolation RMSE (best performance = 1) on 13 time-series. The methods are ordered by median.

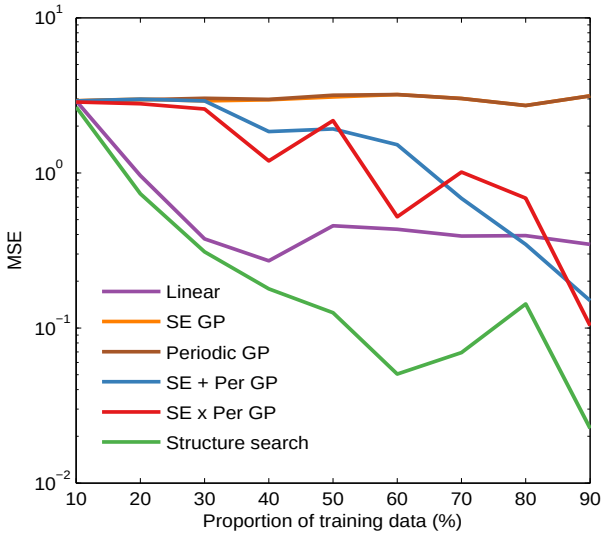


Fig. 3.8 Extrapolation performance on the airline dataset. We plot test-set MSE as a function of the fraction of the dataset used for training.

Chapter 4

Automatically Describing Structured Covariance Functions

This paper presents the beginnings of an automatic statistician, focusing on regression problems. Our system explores an open-ended space of possible statistical models to discover a good explanation of the data, and then produces a detailed report with figures and natural-language text.

Our approach treats unknown functions nonparametrically using Gaussian processes, which has two important consequences. First, Gaussian processes model functions in terms of high-level properties (e.g. smoothness, trends, periodicity, changepoints). Taken together with the compositional structure of our language of models, this allows us to automatically describe functions through a decomposition into additive parts. Second, the use of flexible nonparametric models and a rich language for composing them in an open-ended manner also results in state-of-the-art extrapolation performance evaluated over 13 real time series data sets from various domains.

4.0.1 Attribution

Joint work with James Robert Lloyd, Roger Grosse, Joshua B. Tenenbaum and Zoubin Ghahramani.

4.1 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 found 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.

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 Per) 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:

$$K \prod_m \mathrm{Lin}^{(m)} \prod_n \boldsymbol{\sigma}^{(n)}, \tag{4.1}$$

where K is one of WN, C, SE, $\prod_k \mathrm{Per}^{(k)}$ or $\mathrm{SE} \prod_k \mathrm{Per}^{(k)}$ 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.

Each kernel in a product modifies a model in a consistent way This allows us to describe the contribution of each kernel in a product as an adjective, or more generally as a modifier of a noun. We now describe how each kernel modifies a model and how this can be described in natural language:

- **Multiplication by SE** removes long range correlations from a model since $\mathrm{SE}(x, x')$ decreases monotonically to 0 as $|x - x'|$ increases. This can be described as making an existing model's correlation structure 'local' or 'approximate'.
- **Multiplication by Lin** is equivalent to multiplying the function being modeled 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 and can be described as e.g. 'with linearly increasing standard deviation'.

- **Multiplication by $\boldsymbol{\sigma}$** is equivalent to multiplying the function being modeled by a sigmoid which means that the function goes to zero before or after some point. This can be described as e.g. 'from [time]' or 'until [time]'.

- **Multiplication by Per** 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

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

  This can be loosely described as e.g. 'modulated by a periodic function with a period of [period] [units]'.

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 e.g. 'smooth function' for SE. 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 $\mathrm{SE} \times \mathrm{Per} \times \mathrm{Lin} \times \boldsymbol{\sigma}$ could be described as an

$$\underbrace{\mathrm{SE}}_{\text{approximately}} \times \underbrace{\mathrm{Per}}_{\text{periodic function}} \times \underbrace{\mathrm{Lin}}_{\text{with linearly growing amplitude}} \times \underbrace{\boldsymbol{\sigma}}_{\text{until 1700.}}$$

where Per has been selected as the head noun.

In principle, any assignment of kernels in a product to these different phrasal roles is possible, but in practice we found certain assignments to produce more interpretable phrases than others. The head noun is chosen according to the following ordering:

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

i.e. Per is always chosen as the head noun when present.

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.

4.1.1 Worked example

Suppose we start with a kernel of the form

$$\mathrm{SE} \times (\mathrm{WN} \times \mathrm{Lin} + \mathrm{CP}(\mathrm{C}, \mathrm{Per})).$$

This is converted to a sum of products:

$$\mathrm{SE} \times \mathrm{WN} \times \mathrm{Lin} + \mathrm{SE} \times \mathrm{C} \times \boldsymbol{\sigma} + \mathrm{SE} \times \mathrm{Per} \times \bar{\boldsymbol{\sigma}}.$$

which is simplified to

$$\mathrm{WN} \times \mathrm{Lin} + \mathrm{SE} \times \boldsymbol{\sigma} + \mathrm{SE} \times \mathrm{Per} \times \bar{\boldsymbol{\sigma}}.$$

To describe the first component, the head noun description for WN, 'uncorrelated noise', is concatenated with a modifier for Lin, 'with linearly increasing standard deviation'. 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]'.

Automating the process of statistical modeling would have a tremendous impact on fields that currently rely on expert statisticians, machine learning researchers, and data scientists. While fitting simple models (such as linear regression) is largely automated by standard software packages, there has been little work on the automatic construction of flexible but interpretable models. What are the ingredients required for an artificial intelligence system to be able to perform statistical modeling automatically? In this paper we conjecture that the following ingredients may be useful for building an AI system for statistics, and we develop a working system which incorporates them:

- **An open-ended language of models** expressive enough to capture many of the modeling assumptions and model composition techniques applied by human statisticians to capture real-world phenomena

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.

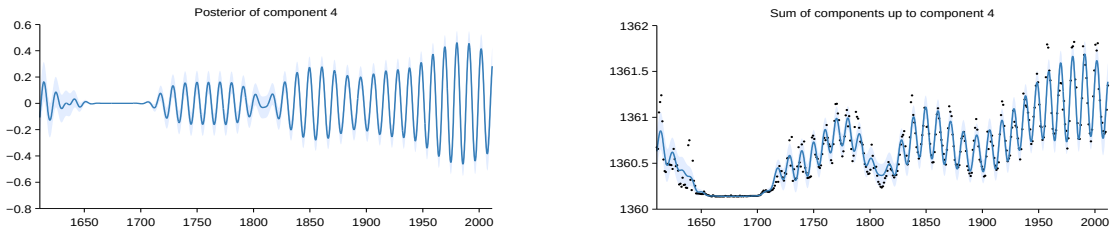


Figure 8: Pointwise posterior of component 4 (left) and the posterior of the cumulative sum of components with data (right)

Fig. 4.1 Extract from an automatically-generated report describing the model components discovered by automatic model search. 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).

- **A search procedure** to efficiently explore the space of models spanned by the language
- **A principled method for evaluating models** in terms of their complexity and their degree of fit to the data
- **A procedure for automatically generating reports** which explain and visualize different factors underlying the data, make the chosen modeling assumptions explicit, and quantify how each component improves the predictive power of the model

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.

We show examples of automatically generated reports which highlight interpretable features discovered in a variety of data sets (e.g. figure 4.4). The supplementary material to this paper includes 13 complete reports automatically generated by ABCD.

Good statistical modeling requires not only interpretability but predictive accuracy.

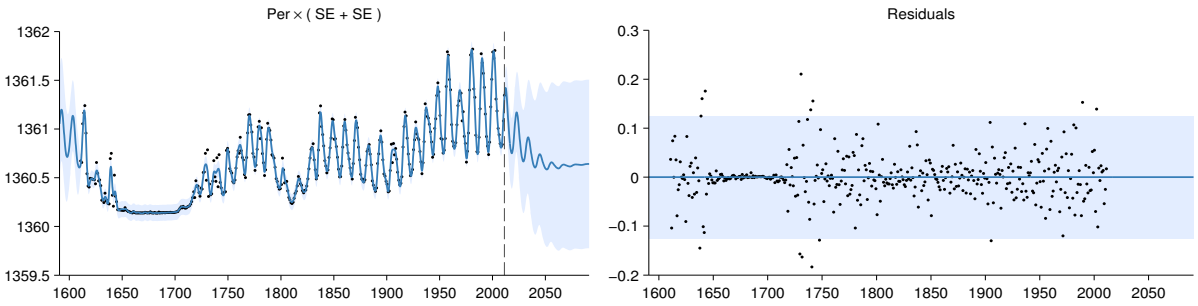


Fig. 4.2 Full posterior and residuals on the solar irradiance dataset.

Airline passenger data Figure 3.6 shows the decomposition produced by applying our method to monthly totals of international airline passengers (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 analyzed annual solar irradiation data from 1610 to 2011 (Lean et al., 1995).

We demonstrate the ability of our procedure to discover and describe a variety of patterns on two time series. Full automatically-generated reports for 13 data sets are provided as supplementary material.

4.1.2 Summarizing 400 Years of Solar Activity

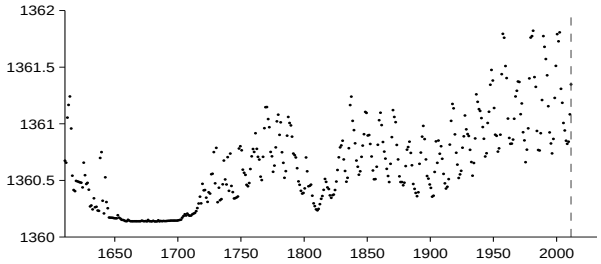


Fig. 4.3 Solar irradiance data.

We show excerpts from the report automatically generated on annual solar irradiation data from 1610 to 2011 (figure 4.3). 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 corresponds to the Maunder minimum, a period in which sunspots were extremely rare (Lean et al., 1995). ABCD clearly identifies these two features, as discussed below.

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.

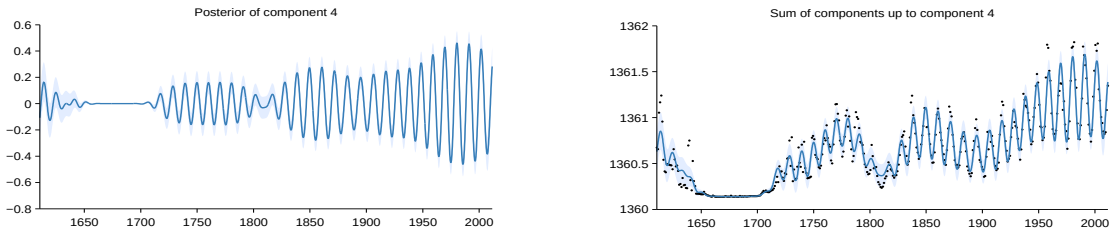


Figure 8: Pointwise posterior of component 4 (left) and the posterior of the cumulative sum of components with data (right)

Fig. 4.4 Extract from an automatically-generated report describing the model components discovered by automatic model search. 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).

Figure 4.5 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 visualized in figures 4.6 and 4.4, respectively. The third component corresponds to long-term trends, as visualized in figure 4.7.

4.1.3 Finding heteroscedasticity in air traffic data

Next, we present the analysis generated by our procedure on international airline passenger data (figure 4.8). The model constructed by ABCD has four components: Lin + SE × Per × Lin + SE + WN × Lin, with descriptions given in figure 4.9.

The second component (figure 4.10) is accurately described as approximately (SE) periodic (Per) with linearly growing amplitude (Lin). By multiplying a white noise kernel by a linear kernel, the model is able to express heteroscedasticity (figure 4.11).

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.

Fig. 4.5 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.

4.1.4 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, 2011) for the data sets shown above, using the default mean absolute error performance metric.

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.

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.

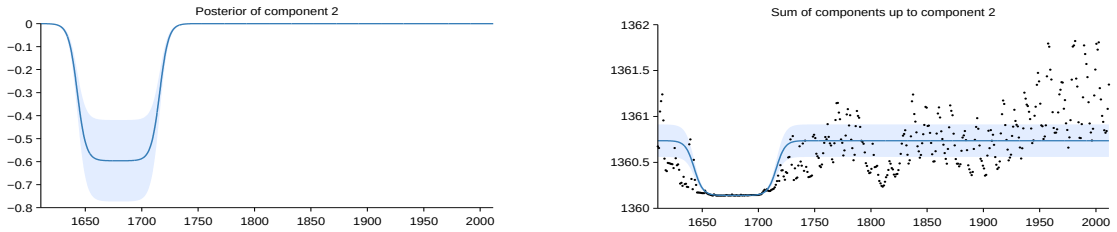


Figure 4: Pointwise posterior of component 2 (left) and the posterior of the cumulative sum of components with data (right)

Fig. 4.6 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.

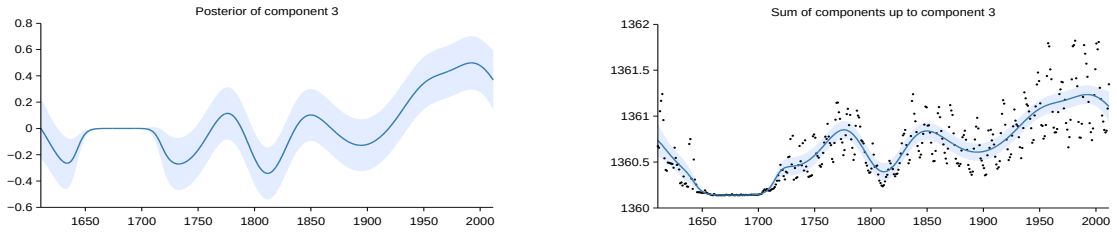


Figure 6: Pointwise posterior of component 3 (left) and the posterior of the cumulative sum of components with data (right)

Fig. 4.7 Characterizing 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.

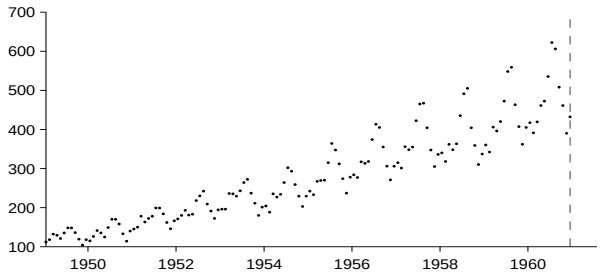


Fig. 4.8 International airline passenger monthly volume (e.g. Box et al., 2013).

4.2 Related work

4.2.1 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 video interpretation (Barbu et al., 2012) and automated theorem proving (Ganesalingam and Gowers, 2013).

4.2.2 Interpretability versus accuracy

BIC trades off model fit and complexity by penalizing the number of parameters in a kernel expression. This can result in ABCD favoring 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 interpretable. We experimented with distributing all products over addition during the search, causing models with many additive components to be more heavily penalized by BIC. We call this procedure ABCD-interpretability, in contrast to the unrestricted version of the search, ABCD-accuracy.

4.3 Conclusion

Towards the goal of automating statistical modeling 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

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

Fig. 4.9 Short descriptions and summary statistics for the four components of the airline model.

on time-series are state-of-the-art compared to existing model construction techniques. We believe this procedure has the potential to make powerful statistical model-building techniques accessible to non-experts.

4.4 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 included in this space includes many standard regression models. We proposed a search procedure for this space of kernels which parallels the process of scientific discovery.

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 model-checking by humans. We believe that a data-driven approach to choosing ker-

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.

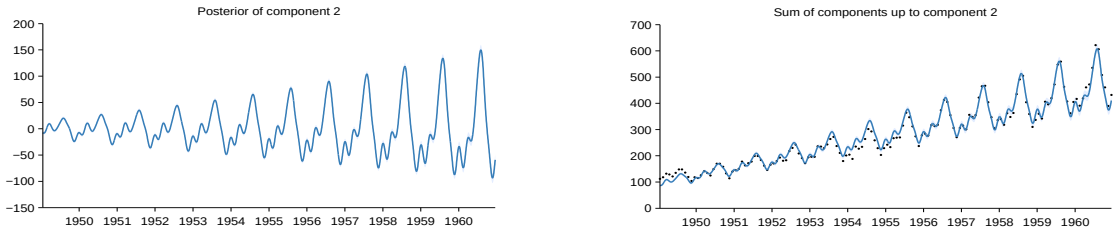


Figure 4: Pointwise posterior of component 2 (left) and the posterior of the cumulative sum of components with data (right)

Fig. 4.10 Capturing non-stationary periodicity in the airline data

nel structures automatically can help make nonparametric regression and classification methods accessible to non-experts.

[1]

[1] All GP parameter optimisation was performed by automated calls to the GPML toolbox available at http://www.gaussianprocess.org/gpml/code/.

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.

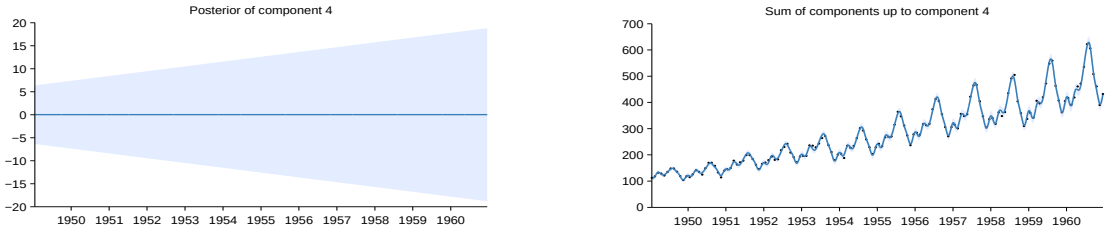


Figure 8: Pointwise posterior of component 4 (left) and the posterior of the cumulative sum of components with data (right)

Fig. 4.11 Modeling heteroscedasticity in the airline dataset.

Chapter 5

Dropout in Gaussian processes

5.1 Dropout in Gaussian processes

Dropout is a method for regularizing neural networks (**??**). Training with dropout entails randomly and independently "dropping" (setting to zero) some proportion p of features or inputs, in order to improve the robustness of the resulting network by reducing co-dependence between neurons. To maintain similar overall activation levels, weights are multiplied by $1/p$ at test time. Alternatively, feature activations are multiplied by $1/p$ during training. At test time, the set of models defined by all possible ways of dropping-out neurons is averaged over, usually in an approximate way.

? and **?** analyzed dropout in terms of the effective prior induced by this procedure in several models, such as linear and logistic regression. In this section, we examine the priors on functions that result from performing dropout in the one-hidden-layer neural network implicitly defined by a GP (equation (**??**)).

5.1.1 Dropout on feature activations

First, we examine the prior that results from randomly dropping features from $\mathbf{h}(\mathbf{x})$ with probability p. If these features have a weight distribution with finite moments $\mathbb{E}[\alpha_i] = \mu, \mathbb{V}[\alpha_i] = \sigma^2$, then the distribution of weights after each one has been dropped out with probability p is:

$$r_i \overset{\text{iid}}{\sim} \text{Ber}(p) \qquad \mathbb{E}[r_i\alpha_i] = p\mu, \mathbb{V}[r_i\alpha_i] = p^2\sigma^2 \; . \tag{5.1}$$

However, after we increase the remaining activations to maintain the same expected activation by multiplying them by $1/p$, the resulting moments are again:

$$\mathbb{E}\left[\frac{1}{p}r_i\alpha_i\right] = \frac{p}{p}\mu = \mu, \quad \mathbb{V}\left[\frac{1}{p}r_i\alpha_i\right] = \frac{p^2}{p^2}\sigma^2 = \sigma^2 \,. \tag{5.2}$$

Thus, dropping out features of an infinitely-wide MLP does not change the model at all, since no individual feature can have more than infinitesimal contribution to the network output.

5.1.2 Dropping out inputs

In a GP with kernel $k(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{D} k_d(\mathbf{x}_d, \mathbf{x}'_d)$, exact averaging over all possible ways of dropping out inputs with probability $1/2$ results in a mixture of GPs, each depending on only a subset of the inputs:

$$p\left(f(\mathbf{x})\right) = \frac{1}{2^D} \sum_{\mathbf{r}\in\{0,1\}^D} \mathrm{GP}\left(0, \prod_{d=1}^{D} k_d(\mathbf{x}_d, \mathbf{x}'_d)^{r_d}\right) \tag{5.3}$$

We present two results ways to gain intuition about this model.

First, if the kernel on each dimension has the form $k_d(\mathbf{x}_d, \mathbf{x}'_d) = g\left(\frac{\mathbf{x}_d - \mathbf{x}'_d}{w_d}\right)$, as does the SE kernel (**??**), then any input dimension can be dropped out by setting its lengthscale w_d to ∞. Thus, dropout on the inputs of a GP corresponds to a spike-and-slab prior on lengthscales, with each dimension independently having $w_d = \infty$ with probability $1/2$.

Another way to understand the resulting prior is to note that the dropout mixture (5.3) has the same covariance as

$$f(\mathbf{x}) \sim \mathrm{GP}\left(0, \frac{1}{2^{-2D}} \sum_{\mathbf{r}\in\{0,1\}^D} \prod_{d=1}^{D} k_d(\mathbf{x}_d, \mathbf{x}'_d)^{r_d}\right) \tag{5.4}$$

A GP whose covariance is a sum of kernels corresponds to a sum of functions, each distributed according to a GP. Therefore, (5.4) describes a sum of 2^D functions, each depending on a different subset of the inputs. This model class was studied by Duvenaud et al. (2011), who showed that exact inference in these models can be performed in $\mathcal{O}(N^2D^2 + N^3)$.

In this chapter, we introduce a Gaussian process model of functions which are *additive*. An additive function is one which decomposes into a sum of low-dimensional functions, each depending on only a subset of the input variables. Additive GPs general-

ize both Generalized Additive Models, and the standard GP models which use squared-exponential kernels. Hyperparameter learning in this model can be seen as Bayesian Hierarchical Kernel Learning (HKL). We introduce an expressive but tractable parameterization of the kernel function, which allows efficient evaluation of all input interaction terms, whose number is exponential in the input dimension. The additional structure discoverable by this model results in state-of-the-art predictive power in regression tasks, as well as increased interpretability.

5.2 Introduction

Most statistical regression models in use today are of the form: $g(y) = f(x_1) + f(x_2) + \cdots + f(x_D)$. Popular examples include logistic regression, linear regression, and Generalized Linear Models**?**. This family of functions, known as Generalized Additive Models (GAM)Hastie and Tibshirani (1990), are typically easy to fit and interpret. Some extensions of this family, such as smoothing-splines ANOVA Wahba (1990), add terms depending on more than one variable. However, such models generally become intractable and difficult to fit as the number of terms increases.

At the other end of the spectrum are kernel-based models, which typically allow the response to depend on all input variables simultaneously. These have the form: $y = f(x_1, x_2, \ldots, x_D)$. A popular example would be a Gaussian process model using a squared-exponential (or Gaussian) kernel. We denote this model as SE-GP. This model is much more flexible than the GAM, but its flexibility makes it difficult to generalize to new combinations of input variables.

In this paper, we introduce a Gaussian process model that generalizes both GAMs and the SE-GP. This is achieved through a kernel which allow additive interactions of all orders, ranging from first order interactions (as in a GAM) all the way to Dth-order interactions (as in a SE-GP). Although this kernel amounts to a sum over an exponential number of terms, we show how to compute this kernel efficiently, and introduce a parameterization which limits the number of hyperparameters to $O(D)$. A Gaussian process with this kernel function (an additive GP) constitutes a powerful model that allows one to automatically determine which orders of interaction are important. We show that this model can significantly improve modeling efficacy, and has major advantages for model interpretability. This model is also extremely simple to implement, and we provide example code.

We note that a similar breakthrough has recently been made, called Hierarchical

Kernel Learning (HKL)**?**. HKL explores a similar class of models, and sidesteps the possibly exponential number of interaction terms by cleverly selecting only a tractable subset. However, this method suffers considerably from the fact that cross-validation must be used to set hyperparameters. In addition, the machinery necessary to train these models is immense. Finally, on real datasets, HKL is outperformed by the standard SE-GP **?**.

We can see that a GP with a first-order additive kernel is an example of a GAM: Each function drawn from this model is a sum of orthogonal one-dimensional functions. Compared to functions drawn from the higher-order GP, draws from the first-order GP have more long-range structure.

We can expect many natural functions to depend only on sums of low-order interactions. For example, the price of a house or car will presumably be well approximated by a sum of prices of individual features, such as a sun-roof. Other parts of the price may depend jointly on a small set of features, such as the size and building materials of a house. Capturing these regularities will mean that a model can confidently extrapolate to unseen combinations of features.

5.3 Additive Kernels

We now give a precise definition of additive kernels. We first assign each dimension $i \in \{1 \dots D\}$ a one-dimensional *base kernel* $k_i(x_i, x_i')$. We then define the first order, second order and nth order additive kernel as:

$$k_{add_1}(\mathbf{x}, \mathbf{x}') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x_i') \tag{5.5}$$

$$k_{add_2}(\mathbf{x}, \mathbf{x}') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} k_i(x_i, x_i') k_j(x_j, x_j') \tag{5.6}$$

$$k_{add_n}(\mathbf{x}, \mathbf{x}') = \sigma_n^2 \sum_{1 \le i_1 < i_2 < \ldots < i_n \le D} \left[\prod_{d=1}^{n} k_{i_d}(x_{i_d}, x_{i_d}') \right] \tag{5.7}$$

where D is the dimension of our input space, and σ_n^2 is the variance assigned to all nth order interactions. The nth covariance function is a sum of $\binom{D}{n}$ terms. In particular, the Dth order additive covariance function has $\binom{D}{D} = 1$ term, a product of each dimension's covariance function:

$$k_{add_D}(\mathbf{x}, \mathbf{x}') = \sigma_D^2 \prod_{d=1}^{D} k_d(x_d, x_d') \tag{5.8}$$

In the case where each base kernel is a one-dimensional squared-exponential kernel, the Dth-order term corresponds to the multivariate squared-exponential kernel:

$$k_{add_D}(\mathbf{x}, \mathbf{x}') = \sigma_D^2 \prod_{d=1}^{D} k_d(x_d, x'_d) = \sigma_D^2 \prod_{d=1}^{D} \exp\left(-\frac{(x_d - x'_d)^2}{2l_d^2}\right) = \sigma_D^2 \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2l_d^2}\right) \tag{5.9}$$

also commonly known as the Gaussian kernel. The full additive kernel is a sum of the additive kernels of all orders.

5.3.1 Parameterization

The only design choice necessary in specifying an additive kernel is the selection of a one-dimensional base kernel for each input dimension. Any parameters (such as length-scales) of the base kernels can be learned as usual by maximizing the marginal likelihood of the training data.

In addition to the hyperparameters of each dimension-wise kernel, additive kernels are equipped with a set of D hyperparameters $\sigma_1^2 \dots \sigma_D^2$ controlling how much variance we assign to each order of interaction. These "order variance" hyperparameters have a useful interpretation: The dth order variance hyperparameter controls how much of the target function's variance comes from interactions of the dth order. Table 5.1 shows examples of normalized order variance hyperparameters learned on real datasets.

Table 5.1 Relative variance contribution of each order in the additive model, on different datasets. Here, the maximum order of interaction is set to 10, or smaller if the input dimension less than 10. Values are normalized to sum to 100.

Order of interaction	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
pima	0.1	0.1	0.1	0.3	1.5	**96.4**	1.4	0.0		
liver	0.0	0.2	**99.7**	0.1	0.0	0.0				
heart	**77.6**	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	22.0
concrete	**70.6**	13.3	13.8	2.3	0.0	0.0	0.0	0.0		
pumadyn-8nh	0.0	0.1	0.1	0.1	0.1	0.1	0.1	**99.5**		
servo	**58.7**	27.4	0.0	13.9						
housing	0.1	0.6	**80.6**	1.4	1.8	0.8	0.7	0.8	0.6	12.7

On different datasets, the dominant order of interaction estimated by the additive model varies widely. An additive GP with all of its variance coming from the 1st order

is equivalent to a GAM; an additive GP with all its variance coming from the Dth order is equivalent to a SE-GP.

Because the hyperparameters can specify which degrees of interaction are important, the additive GP is an extremely general model. If the function we are modeling is decomposable into a sum of low-dimensional functions, our model can discover this fact and exploit it (see Figure 2.4) . If this is not the case, the hyperparameters can specify a suitably flexible model.

5.3.2 Interpretability

As noted by Plate (1999), one of the chief advantages of additive models such as GAM is their interpretability. Plate also notes that by allowing high-order interactions as well as low-order interactions, one can trade off interpretability with predictive accuracy. In the case where the hyperparameters indicate that most of the variance in a function can be explained by low-order interactions, it is useful and easy to plot the corresponding low-order functions, as in Figure 5.1.

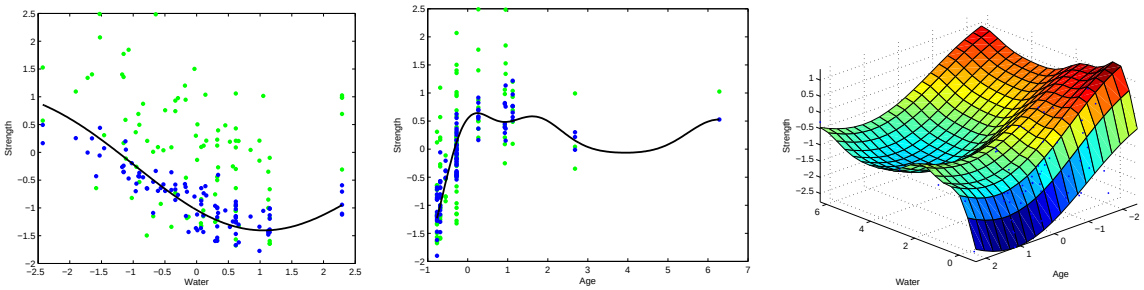


Fig. 5.1 Low-order functions on the concrete dataset. Left, Centre: By considering only first-order terms of the additive kernel, we recover a form of Generalized Additive Model, and can plot the corresponding 1-dimensional functions. Green points indicate the original data, blue points are data after the mean contribution from the other dimensions' first-order terms has been subtracted. The black line is the posterior mean of a GP with only one term in its kernel. Right: The posterior mean of a GP with only one second-order term in its kernel.

5.3.3 Efficient Evaluation of Additive Kernels

An additive kernel over D inputs with interactions up to order n has $O(2^n)$ terms. Naïvely summing over these terms quickly becomes intractable. In this section, we show how one can evaluate the sum over all terms in $O(D^2)$.

The nth order additive kernel corresponds to the nth *elementary symmetric polynomial*? ?, which we denote e_n. For example: if $\mathbf{x}$ has 4 input dimensions ($D = 4$), and if we let $z_i = k_i(x_i, x_i')$, then

$$
\begin{aligned}
k_{add_1}(\mathbf{x}, \mathbf{x}') &= e_1(z_1, z_2, z_3, z_4) = z_1 + z_2 + z_3 + z_4 \\
k_{add_2}(\mathbf{x}, \mathbf{x}') &= e_2(z_1, z_2, z_3, z_4) = z_1 z_2 + z_1 z_3 + z_1 z_4 + z_2 z_3 + z_2 z_4 + z_3 z_4 \\
k_{add_3}(\mathbf{x}, \mathbf{x}') &= e_3(z_1, z_2, z_3, z_4) = z_1 z_2 z_3 + z_1 z_2 z_4 + z_1 z_3 z_4 + z_2 z_3 z_4 \\
k_{add_4}(\mathbf{x}, \mathbf{x}') &= e_4(z_1, z_2, z_3, z_4) = z_1 z_2 z_3 z_4
\end{aligned}
$$

The Newton-Girard formulae give an efficient recursive form for computing these polynomials. If we define s_k to be the kth power sum: $s_k(z_1, z_2, \ldots, z_D) = \sum_{i=1}^{D} z_i^k$, then

$$
k_{add_n}(\mathbf{x}, \mathbf{x}') = e_n(z_1, \ldots, z_D) = \frac{1}{n} \sum_{k=1}^{n} (-1)^{(k-1)} e_{n-k}(z_1, \ldots, z_D) s_k(z_1, \ldots, z_D) \tag{5.10}
$$

Where $e_0 \triangleq 1$. The Newton-Girard formulae have time complexity $O(D^2)$, while computing a sum over an exponential number of terms.

Conveniently, we can use the same trick to efficiently compute all of the necessary derivatives of the additive kernel with respect to the base kernels. We merely need to remove the kernel of interest from each term of the polynomials:

$$
\frac{\partial k_{add_n}}{\partial z_j} = e_{n-1}(z_1, \ldots, z_{j-1}, z_{j+1}, \ldots z_D) \tag{5.11}
$$

This trick allows us to optimize the base kernel hyperparameters with respect to the marginal likelihood.

5.3.4 Computation

The computational cost of evaluating the Gram matrix of a product kernel (such as the SE kernel) is $O(N^2 D)$, while the cost of evaluating the Gram matrix of the additive kernel is $O(N^2 DR)$, where R is the maximum degree of interaction allowed (up to D). In higher dimensions, this can be a significant cost, even relative to the fixed $O(N^3)$ cost of inverting the Gram matrix. However, as our experiments show, typically only the first few orders of interaction are important for modeling a given function; hence if one is computationally limited, one can simply limit the maximum degree of interaction

without losing much accuracy.

Additive Gaussian processes are particularly appealing in practice because their use requires only the specification of the base kernel. All other aspects of GP inference remain the same. All of the experiments in this paper were performed using the standard GPML toolbox[1]; code to perform all experiments is available at the author's website.[2]

5.4 Related Work

PlatePlate (1999) constructs a form of additive GP, but using only the first-order and Dth order terms. This model is motivated by the desire to trade off the interpretability of first-order models, with the flexibility of full-order models. Our experiments show that often, the intermediate degrees of interaction contribute most of the variance.

A related functional ANOVA GP model**?** decomposes the *mean* function into a weighted sum of GPs. However, the effect of a particular degree of interaction cannot be quantified by that approach. Also, computationally, the Gibbs sampling approach used in **?** is disadvantageous.

Christoudias et al.Christoudias et al. (2009) previously showed how mixtures of kernels can be learnt by gradient descent in the Gaussian process framework. They call this *Bayesian localized multiple kernel learning*. However, their approach learns a mixture over a small, fixed set of kernels, while our method learns a mixture over all possible products of those kernels.

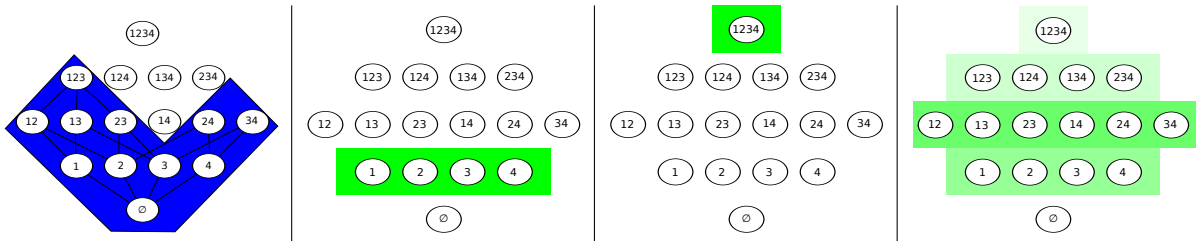
5.4.1 Hierarchical Kernel Learning

Bach**?** uses a regularized optimization framework to learn a weighted sum over an exponential number of kernels which can be computed in polynomial time. The subsets of kernels considered by this method are restricted to be a *hull* of kernels.[3] Given each dimension's kernel, and a pre-defined weighting over all terms, HKL performs model selection by searching over hulls of interaction terms. In **?**, Bach also fixes the relative weighting between orders of interaction with a single term α, computing the sum over

[1] Available at `http://www.gaussianprocess.org/gpml/code/`

[2] Example code available at: `http://mlg.eng.cam.ac.uk/duvenaud/`

[3] In the setting we are considering in this paper, a hull can be defined as a subset of all terms such that if term $\prod_{j \in J} k_j(\mathbf{x}, \mathbf{x}')$ is included in the subset, then so are all terms $\prod_{j \in J/i} k_j(\mathbf{x}, \mathbf{x}')$, for all $i \in J$. For details, see **?**.



HKL kernel | GP-GAM kernel | Squared-exp GP kernel | Additive GP kernel

Fig. 5.2 A comparison of different additive model classes. Nodes represent different interaction terms, ranging from first-order to fourth-order interactions. Far left: HKL can select a hull of interaction terms, but must use a pre-determined weighting over those terms. Far right: the additive GP model can weight each order of interaction seperately. Neither the HKL nor the additive model dominate one another in terms of flexibility, however the GP-GAM and the SE-GP are special cases of additive GPs.

all orders by:

$$k_a(\mathbf{x}, \mathbf{x}') = v_D^2 \prod_{d=1}^{D} (1 + \alpha k_d(x_d, x'_d)) \tag{5.12}$$

which has computational complexity $O(D)$. However, this formulation forces the weight of all nth order terms to be weighted by α^n.

Figure 5.2 contrasts the HKL hull-selection method with the Additive GP hyperparameter-learning method. Neither method dominates the other in flexibility. The main difficulty with the approach of **?** is that hyperparameters are hard to set other than by cross-validation. In contrast, our method optimizes the hyperparameters of each dimension's base kernel, as well as the relative weighting of each order of interaction.

5.4.2 ANOVA Procedures

Vapnik **?** introduces the support vector ANOVA decomposition, which has the same form as our additive kernel. However, they recommend approximating the sum over all D orders with only one term "of appropriate order", presumably because of the difficulty of setting the hyperparameters of an SVM. Stitson et al.**?** performed experiments which favourably compared the support vector ANOVA decomposition to polynomial and spline kernels. They too allowed only one order to be active, and set hyperparameters by cross-validation.

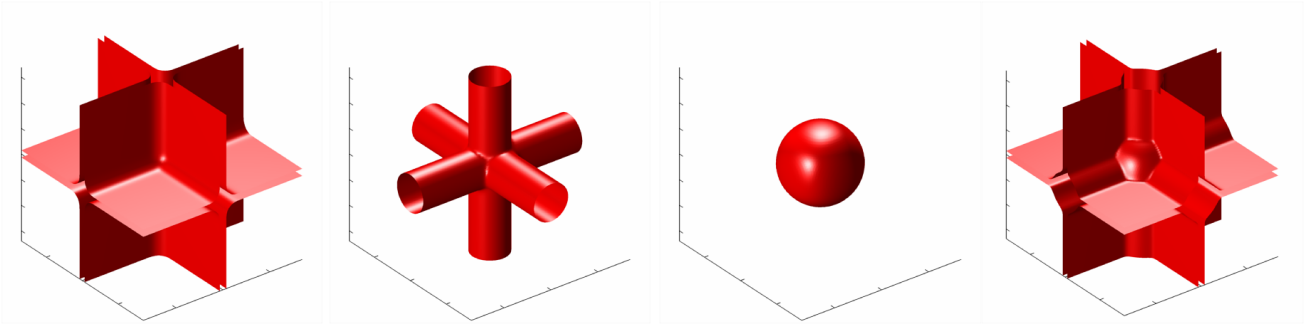
A closely related procedure from the statistics literature is smoothing-splines ANOVA (SS-ANOVA)Wahba (1990). An SS-ANOVA model is estimated as a weighted sum of

splines along each dimension, plus a sum of splines over all pairs of dimensions, all triplets, etc, with each individual interaction term having a separate weighting parameter. Because the number of terms to consider grows exponentially in the order, in practice, only terms of first and second order are usually considered. Learning in SS-ANOVA is usually done via penalized-maximum likelihood with a fixed sparsity hyperparameter.

In contrast to these procedures, our method can easily include all D orders of interaction, each weighted by a separate hyperparameter. As well, we can learn kernel hyperparameters individually per input dimension, allowing automatic relevance determination to operate.

5.4.3 Non-local Interactions

By far the most popular kernels for regression and classification tasks on continuous data are the squared exponential (Gaussian) kernel, and the Matérn kernels. These kernels depend only on the scaled Euclidean distance between two points, both having the form: $k(\mathbf{x}, \mathbf{x}') = f(\sum_{d=1}^{D} (x_d - x'_d)^2 / l_d^2)$. Bengio et al.**?** argue that models based on squared-exponential kernels are particularily susceptible to the *curse of dimensionality*. They emphasize that the locality of the kernels means that these models cannot capture non-local structure. They argue that many functions that we care about have such structure. Methods based solely on local kernels will require training examples at all combinations of relevant inputs.



1st order interactions $k_1 + k_2 + k_3$ | 2nd order interactions $k_1k_2 + k_2k_3 + k_1k_3$ | 3rd order interactions $k_1k_2k_3$ (Squared-exp kernel) | All interactions (Additive kernel)

Fig. 5.3 Isocontours of additive kernels in 3 dimensions. The third-order kernel only considers nearby points relevant, while the lower-order kernels allow the output to depend on distant points, as long as they share one or more input value.

Additive kernels have a much more complex structure, and allow extrapolation based

on distant parts of the input space, without spreading the mass of the kernel over the whole space. For example, additive kernels of the second order allow strong non-local interactions between any points which are similar in any two input dimensions. Figure 5.3 provides a geometric comparison between squared-exponential kernels and additive kernels in 3 dimensions.

5.5 Experiments

5.5.1 Experimental Setup

On a diverse collection of datasets, we compared five different models. In the results tables below, GP Additive refers to a GP using the additive kernel with squared-exp base kernels. For speed, we limited the maximum order of interaction in the additive kernels to 10. GP-GAM denotes an additive GP model with only first-order interactions. GP Squared-Exp is a GP model with a squared-exponential ARD kernel. HKL[4] was run using the all-subsets kernel, which corresponds to the same set of kernels as considered by the additive GP with a squared-exp base kernel.

For all GP models, we fit hyperparameters by the standard method of maximizing training-set marginal likelihood, using L-BFGS **?** for 500 iterations, allowing five random restarts. In addition to learning kernel hyperparameters, we fit a constant mean function to the data. In the classification experiments, GP inference was done using Expectation Propagation **?**.

5.5.2 Bach Synthetic Dataset

In addition to standard UCI repository datasets, we generated a synthetic dataset following the same recipe as **?**: From a covariance matrix drawn from a Wishart distribution with 1024 degrees of freedom, we select 8 variables. We then construct the non-linear function $f(X) = \sum_{i=1}^{4} \sum_{j=1+1}^{4} X_i X_j + \epsilon$, which sums all 2-way products of the first 4 variables, and adds Gaussian noise ϵ. This dataset is one which can be predicted well by a kernel which is a sum of two-way interactions over the first 4 variables, ignoring the extra 4 noisy copies.

This dataset was designed by **?** to demonstrate the advantages of HKL over GP-ARD.

[4]Code for HKL available at `http://www.di.ens.fr/~fbach/hkl/`

If the dataset is large enough, HKL can construct a hull around only those subsets of cross terms that are optimal for predicting the output. GP-ARD, in contrast, can only learn to ignore the noisy copy variables, but cannot learn to ignore the higher-term interactions between the predictive variables. However, a GP with an additive kernel can learn both to ignore irrelevant variables, and to ignore certain orders of interaction. In this example, the additive GP is able to recover the relevant structure.

5.5.3 Results

Tables 5.2, 5.3, 5.4 and 5.5 show mean performance across 10 train-test splits. Because HKL does not specify a noise model, it could not be included in the likelihood comparisons.

Table 5.2 Regression Mean Squared Error

Method	bach	concrete	pumadyn-8nh	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**

Table 5.3 Regression Negative Log Likelihood

Method	bach	concrete	pumadyn-8nh	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**

The model with best performance on each dataset is in bold, along with all other models that were not significantly different under a paired t-test. The additive model never performs significantly worse than any other model, and sometimes performs significantly better than all other models. The difference between all methods is larger in the case of regression experiments. The performance of HKL is consistent with the results in **?**, performing competitively but slightly worse than SE-GP.

Table 5.4 Classification percent error

Method	breast	pima	sonar	ionosphere	liver	heart
Logistic Regression	7.611	24.392	26.786	16.810	45.060	**16.082**
GP GAM	**5.189**	**22.419**	**15.786**	**8.524**	**29.842**	**16.839**
HKL	**5.377**	24.261	**21.000**	9.119	**27.270**	**18.975**
GP Squared-exp	**4.734**	**23.722**	**16.357**	**6.833**	**31.237**	**20.642**
GP Additive	**5.566**	**23.076**	**15.714**	**7.976**	**30.060**	**18.496**

Table 5.5 Classification negative log-likelihood

Method	breast	pima	sonar	ionosphere	liver	heart
Logistic Regression	0.247	0.560	4.609	0.878	0.864	0.575
GP GAM	**0.163**	**0.461**	**0.377**	**0.312**	**0.569**	**0.393**
GP Squared-exp	**0.146**	0.478	**0.425**	**0.236**	**0.601**	0.480
GP Additive	**0.150**	**0.466**	**0.409**	**0.295**	**0.588**	**0.415**

The additive GP performed best on datasets well-explained by low orders of interaction, and approximately as well as the SE-GP model on datasets which were well explained by high orders of interaction (see table 5.1). Because the additive GP is a superset of both the GP-GAM model and the SE-GP model, instances where the additive GP performs slightly worse are presumably due to over-fitting, or due to the hyperparameter optimization becoming stuck in a local maximum. Additive GP performance can be expected to benefit from integrating out the kernel hyperparameters.

5.5.4 Future Work

Since the non-local structure capturable by additive kernels is necessarily axis-aligned, we can naturally consider that combining the hyperparameter optimization with an initial rotation of the input space might allow us to recover non-axis aligned additivity in functions.

Note that we are free to choose a different covariance function along each dimension.

5.6 Conclusion

We present additive Gaussian processes: a simple family of models which generalizes two widely-used classes of models. Additive GPs also introduce a tractable new type of structure into the GP framework. Our experiments indicate that such additive structure is present in real datasets, allowing our model to perform better than standard GP models. In the case where no such structure exists, our model can recover arbitrarily flexible models, as well.

In addition to improving modeling efficacy, the additive GP also improves model interpretability: the order variance hyperparameters indicate which sorts of structure are present in our model.

Compared to HKL, which is the only other tractable procedure able to capture the same types of structure, our method benefits from being able to learn individual kernel hyperparameters, as well as the weightings of different orders of interaction. Our experiments show that additive GPs are a state-of-the-art regression model.

Appendix A

Appendix for Grammars

A.1 Kernels

A.1.1 Base kernels

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

$$\mathrm{WN}(x, x') = \sigma^2 \delta_{x,x'} \tag{A.1}$$

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

$$\mathrm{Lin}(x, x') = \sigma^2 (x - \ell)(x' - \ell) \tag{A.3}$$

$$\mathrm{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) \tag{A.4}$$

$$\mathrm{Per}(x, x') = \sigma^2 \frac{\exp\left(\frac{\cos\frac{2\pi(x-x')}{p}}{\ell^2}\right) - I_0\left(\frac{1}{\ell^2}\right)}{\exp\left(\frac{1}{\ell^2}\right) - I_0\left(\frac{1}{\ell^2}\right)} \tag{A.5}$$

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.

A.1.2 Changepoints and changewindows

The changepoint, $\mathrm{CP}(\cdot, \cdot)$ operator is defined as follows:

$$\begin{aligned} \mathrm{CP}(k_1, k_2)(x, x') = {} & \sigma(x) k_1(x, x') \sigma(x') \\ & + (1 - \sigma(x)) k_2(x, x') (1 - \sigma(x')) \end{aligned} \tag{A.6}$$

where $\sigma(x) = 0.5 \times (1 + \tanh(\frac{\ell - x}{s}))$. This can also be written as

$$\mathrm{CP}(k_1, k_2) = \boldsymbol{\sigma} k_1 + \bar{\boldsymbol{\sigma}} k_2 \tag{A.7}$$

where $\boldsymbol{\sigma}(x, x') = \sigma(x)\sigma(x')$ and $\bar{\boldsymbol{\sigma}}(x, x') = (1 - \sigma(x))(1 - \sigma(x'))$.

Changewindow, $\mathrm{CW}(\cdot, \cdot)$, operators are defined similarly by replacing the sigmoid, $\sigma(x)$, with a product of two sigmoids.

A.1.3 Properties of the periodic kernel

A simple application of l'Hôpital's rule shows that

$$\mathrm{Per}(x, x') \to \sigma^2 \cos\left(\frac{2\pi(x - x')}{p}\right) \quad \text{as} \quad \ell \to \infty. \tag{A.8}$$

This limiting form is written as the cosine kernel (cos).

A.2 Model construction / search

A.2.1 Overview

The model construction phase of ABCD starts with the kernel equal to the noise kernel, WN. New kernel expressions are generated by applying search operators to the current kernel. When new base kernels are proposed by the search operators, their parameters are randomly initialised with several restarts. Parameters are then optimized by conjugate gradients to maximise the likelihood of the data conditioned on the kernel parameters. The kernels are then scored by the Bayesian information criterion and the top scoring kernel is selected as the new kernel. The search then proceeds by applying the search operators to the new kernel i.e. this is a greedy search algorithm.

In all experiments, 10 random restarts were used for parameter initialisation and the search was run to a depth of 10.

A.2.2 Search operators

ABCD is based on a search algorithm which used the following search operators

$$\mathcal{S} \rightarrow \mathcal{S} + \mathcal{B} \tag{A.9}$$
$$\mathcal{S} \rightarrow \mathcal{S} \times \mathcal{B} \tag{A.10}$$
$$\mathcal{B} \rightarrow \mathcal{B}' \tag{A.11}$$

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

To accommodate changepoint/window operators we introduce the following additional operators

$$\mathcal{S} \rightarrow \mathrm{CP}(\mathcal{S}, \mathcal{S}) \tag{A.12}$$
$$\mathcal{S} \rightarrow \mathrm{CW}(\mathcal{S}, \mathcal{S}) \tag{A.13}$$
$$\mathcal{S} \rightarrow \mathrm{CW}(\mathcal{S}, \mathrm{C}) \tag{A.14}$$
$$\mathcal{S} \rightarrow \mathrm{CW}(\mathrm{C}, \mathcal{S}) \tag{A.15}$$

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

$$\mathcal{S} \rightarrow \mathcal{S} \times (\mathcal{B} + \mathrm{C}) \tag{A.16}$$
$$\mathcal{S} \rightarrow \mathcal{B} \tag{A.17}$$
$$\mathcal{S} + \mathcal{S}' \rightarrow \mathcal{S} \tag{A.18}$$
$$\mathcal{S} \times \mathcal{S}' \rightarrow \mathcal{S} \tag{A.19}$$

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

A.3 Predictive accuracy

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 A.1 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.

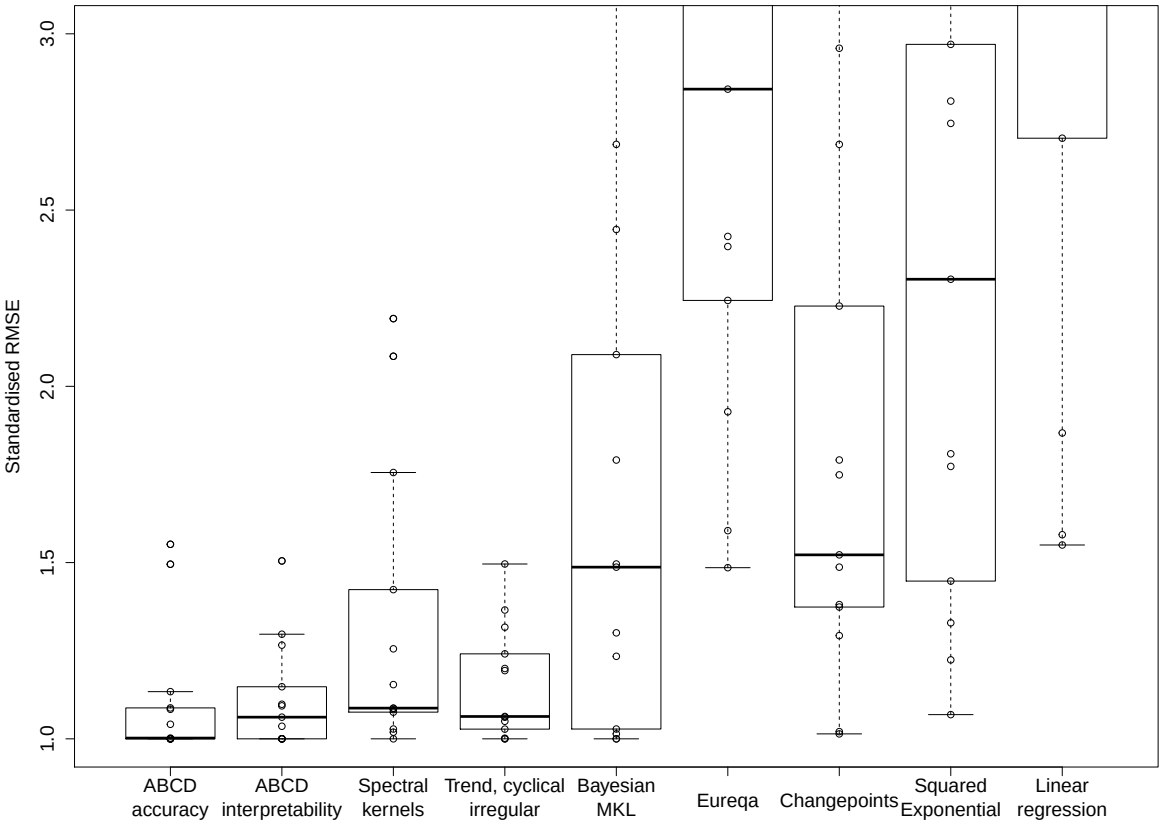


Fig. A.1 Box plot of standardised RMSE (best performance = 1) on 13 interpolation tasks.

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 expressing the changepoints). This results in worse interpolations at the locations of the

change points, suggesting that a more robust modeling 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.

A.3.1 Tabels of standardised RMSEs

See table A.1 for raw interpolation results and table A.2 for raw extrapolation results. The rows follow the order of the datasets in the rest of the supplementary material. The following abbreviations are used: ABCD-accuracy (ABCD-acc), ABCD-interpretability ((ABCD-int), Spectral kernels (SP), Trend-cyclical-irregular (TCI), Bayesian MKL (MKL), Eureqa (EL), Changepoints (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 A.1 Interpolation standardised RMSEs

A.4 Guide to the automatically generated reports

Additional supplementary material to this paper is 13 reports automatically generated by ABCD. A link to these reports will be maintained at http://mlg.eng.cam.ac.uk/lloyd/. We recommend that you read the report for '01-airline' first and review the reports that

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 A.2 Extrapolation standardised RMSEs

follow afterwards more briefly. '02-solar' is discussed in the main text. '03-mauna' analyses a dataset mentioned in the related work. '04-wheat' demonstrates changepoints being used to capture heteroscedasticity. '05-temperature' extracts an exactly periodic pattern from noisy data. '07-call-centre' demonstrates a large discontinuity being modeled by a changepoint. '10-sulphuric' combines many changepoints to create a highly structured model of the data. '12-births' discovers multiple periodic components.

A.5 Discussion

A.5.1 Why haven't structured kernels been built for SVMs?

Because without marginal likelihood to tell you which structure is present in your data, it's not clear how to choose which kernel to use without cross-validation.

A.6 Ingredients of an automatic statistician

Gelman (2013) asks "How can an artificial intelligence do statistics? ... It needs not just an inference engine, but also a way to construct new models and a way to check models. Currently, those steps are performed by humans, but the AI would have to do it itself."

In this section, we discuss in more detail the elements we believe are required to build an artificial intelligence that can do statistics.

1. An open-ended language of models Many statistical procedures consider all models in a class of fixed size - for example, graphical model construction algorithms [(1)] search over connectivity graphs for a given set of nodes. While these methods can be powerful, human statisticians are capable of deriving novel model classes when required by the modelling task. An automatic search through an open-ended class of models can achieve some of this flexibility, growing the complexity of the model to fit the task at hand, and possibly combining existing structures in novel ways.

(1) Cite

2. Searching through model space An open-ended space of models cannot be searched exhaustively. Just as human researchers iteratively refine their models, search procedures can propose new search directions based on the results of previous model fits. Because any search in an open-ended space must start with relatively simple models before moving on to more complex ones, any model search in an open-ended space will likely resemble a model-building procedure.

3. Model comparison and checking model fit [(2)] An automatic statistician should be able to question the models it has constructed, and formal procedures from model checking provide a way for it to do this. Gelman and Shalizi (2012) review the literature on model checking. (!) In this work, we use approximate marginal likelihood to compare models, penalizing complexity using the Bayesian Information Criterion as a heuristic.

(2) JL: Two section haps - or are they pletely intertwined?

4. Describing models Part of the value of statistical models comes from enabling humans to understand a dataset or a phenomenon. Furthermore, a clear description of the statistical structure found in a dataset helps a user to notice when the dataset has errors, the wrong question was asked, the model-building procedure failed to capture known structure, a relevant piece of data or constraint is missing, or when a novel statistical structure has been found.

In this work, we demonstrate that the properties of Gaussian processes allow for a modular description generation procedure. Whether or not such modularity and interpretability is present in other open-ended model classes is an open question.

References

F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *Advances in Neural Information Processing Systems*, pages 105–112. 2009. (pages 22 and 32)

Francis R Bach, Gert RG 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, 2004. (pages 34 and 35)

A Barbu, A Bridge, Z Burchill, D Coroian, S Dickinson, S Fidler, A Michaux, S Mussman, S Narayanaswamy, D Salvi, L Schmidt, J Shangguan, JM Siskind, J Waggoner, S Wang, J Wei, Y Yin, and Z Zhang. Video in sentences out. In *Conference on Uncertainty in Artificial Intelligence*, 2012. (pages 34 and 51)

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. (pages 32 and 33)

Salomon Bochner. *Lectures on Fourier integrals*, volume 42. Princeton University Press, 1959. (page 34)

George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time series analysis: forecasting and control*. Wiley. com, 2013. (pages 35 and 51)

G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time series analysis: forecasting and control*. 1976. (pages 30 and 47)

Youngmin Cho. *Kernel methods for deep learning*. PhD thesis, University of California, San Diego, 2012. (page 18)

M. Christoudias, R. Urtasun, and T. Darrell. Bayesian localized multiple kernel learning. *Technical report, EECS Department, University of California, Berkeley*, 2009. (pages 32 and 62)

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. IEEE, 2007. (pages 32 and 33)

David Duvenaud, Hannes Nickisch, and Carl Edward Rasmussen. Additive Gaussian processes. In *Advances in Neural Information Processing Systems 24*, pages 226–234, Granada, Spain, 2011. (pages 32, 37, and 56)

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. (page 19)

E.B. Fox and D.B. Dunson. Multiresolution Gaussian Processes. In *Neural Information Processing Systems 25*. MIT Press, 2013. (page 35)

M. Ganesalingam and W. T. Gowers. A fully automatic problem solver with human-style output. *CoRR*, abs/1309.4501, 2013. (pages 34 and 51)

Roman Garnett, Michael A Osborne, Steven Reece, Alex Rogers, and Stephen J Roberts. Sequential bayesian prediction in the presence of changepoints and faults. *The Computer Journal*, 53(9):1430–1446, 2010. (page 35)

Andrew Gelman. Why waste time philosophizing?, 2013. URL `http://andrewgelman.com/2013/02/11/why-waste-time-philosophizing/`. (page 74)

Andrew Gelman and Cosma Rohilla Shalizi. Philosophy and the practice of bayesian statistics. *British Journal of Mathematical and Statistical Psychology*, 2012. (page 75)

D. Ginsbourger, O. Roustant, and N. Durrande. Invariances of random fields paths, with applications in gaussian process regression. Technical Report arXiv:1308.1359 [math.ST], August 2013. (page 18)

R.B. Grosse, R. Salakhutdinov, W.T. Freeman, and J.B. Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Uncertainty in Artificial Intelligence*, 2012. (pages 22, 33, and 34)

C. Gu. *Smoothing spline ANOVA models*. Springer Verlag, 2002. ISBN 0387953531. (page 32)

T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990. (pages 7 and 57)

Rob J. Hyndman. Time series data library, Accessed summer 2013. URL `http://data.is/TSDLdemo`. (page 35)

E. T. Jaynes. Highly informative priors. In *Proceedings of the Second International Meeting on Bayesian Statistics*, 1985. (page 21)

C. Kemp and J.B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008. (pages 33 and 34)

Edgar Klenske. *Nonparametric System Identification and Control for Periodic Error Correction in Telescopes*. PhD thesis, University of Stuttgart, 2012. (page 33)

Imre Risi Kondor. *Group theoretical methods in machine learning*. PhD thesis, Columbia University, 2008. (page 19)

Gabriel Kronberger and Michael Kommenda. Evolution of covariance functions for gaussian process regression using genetic programming. *arXiv preprint arXiv:1305.3794*, 2013. (page 33)

N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005. (pages 14 and 32)

Miguel Lázaro-Gredilla, Joaquin Quiñonero-Candela, Carl Edward Rasmussen, and Aníbal R Figueiras-Vidal. Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 99:1865–1881, 2010. (page 34)

J. Lean, J. Beer, and R. Bradley. Reconstruction of solar irradiance since 1610: Implications for climate change. *Geophysical Research Letters*, 22(23):3195–3198, 1995. (pages 30, 46, 47, and 48)

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 Computation*, 1:541–551, 1989. (page 11)

Douglas A Lind, William G Marchal, Samuel Adam Wathen, and Business Week Magazine. *Basic statistics for business and economics*. McGraw-Hill/Irwin Boston, 2006. (page 35)

James Robert Lloyd. GEFCom2012 hierarchical load forecasting: Gradient boosting machines and gaussian processes. *International Journal of Forecasting*, 2013. (page 33)

David JC MacKay. *Information theory, inference, and learning algorithms*. Cambridge university press, 2003. (page 27)

Nutonian. Eureqa, 2011. URL `http://www.nutonian.com/`. (pages 35 and 49)

T.A. Plate. Accuracy versus interpretability in flexible modeling: Implementing a tradeoff using Gaussian process models. *Behaviormetrika*, 26:29–50, 1999. ISSN 0385-7417. (pages 32, 60, and 62)

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Conference on Uncertainty in AI*, pages 689–690. IEEE, 2011. (page 11)

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. *Advances in neural information processing systems*, pages 294–300, 2001. (pages 25 and 27)

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*, volume 38. The MIT Press, Cambridge, MA, USA, 2006. (pages 1, 5, 12, 22, 26, 28, and 33)

D. Ruppert, M.P. Wand, and R.J. Carroll. *Semiparametric regression*, volume 12. Cambridge University Press, 2003. (page 32)

Yunus Saatçi, Ryan D Turner, and Carl E Rasmussen. Gaussian process change point models. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 927–934, 2010. (page 35)

Ruslan Salakhutdinov and Geoffrey Hinton. Using deep belief nets to learn covariance kernels for Gaussian processes. *Advances in Neural information processing systems*, 20:1249–1256, 2008. (pages 22 and 32)

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. (pages 25 and 27)

L. Todorovski and S. Dzeroski. Declarative bias in equation discovery. In *International Conference on Machine Learning*, pages 376–384, 1997. (pages 22, 33, and 34)

G. Wahba. *Spline models for observational data*. Society for Industrial Mathematics, 1990. ISBN 0898712440. (pages 32, 57, and 63)

T. Washio, H. Motoda, Y. Niwa, et al. 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. (pages 33 and 34)

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process covariance kernels for pattern discovery and extrapolation. *arXiv: 1302.4245*, June 2013. (pages 32, 34, and 35)