Chapter 1

Structure through Kernels

This chapter is meant as a tutorial of the different ways one can express structure in a prior over functions.

We shall see that a few Gaussian identities will pop up again and again to help us reason about the covariance function affects the models we'll construct.

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 1.1 and the appendix).

[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.

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

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

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:

$$k_1 + k_2 = k_1(x, x') + k_2(x, x') \tag{1.1}$$

$$k_1 \times k_2 = k_1(x, x') \times k_2(x, x') \tag{1.2}$$

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).

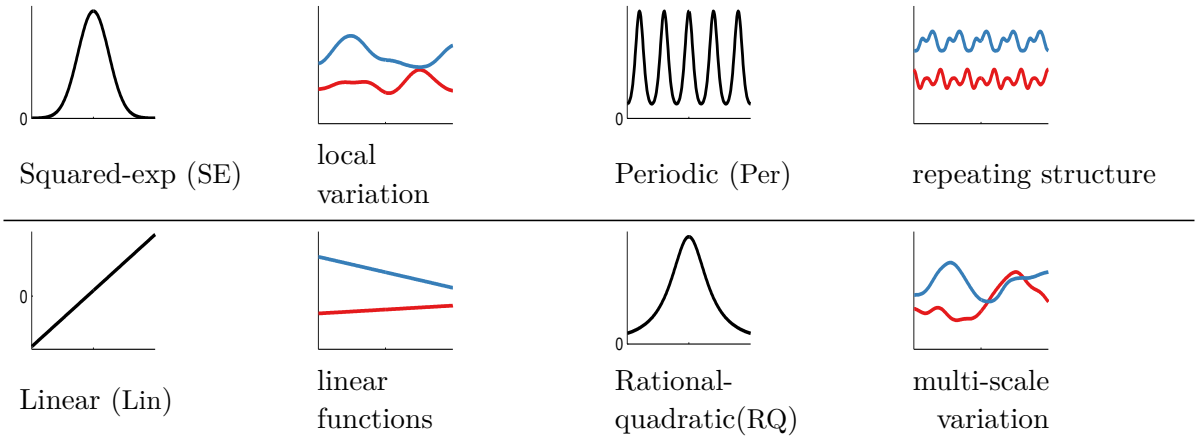


Fig. 1.1 Examples of structures expressible by base kernels. Left and third columns: base kernels $k(\cdot, 0)$. Second and fourth columns: draws from a GP with each repective kernel. The x-axis has the same range on all plots.

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

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.

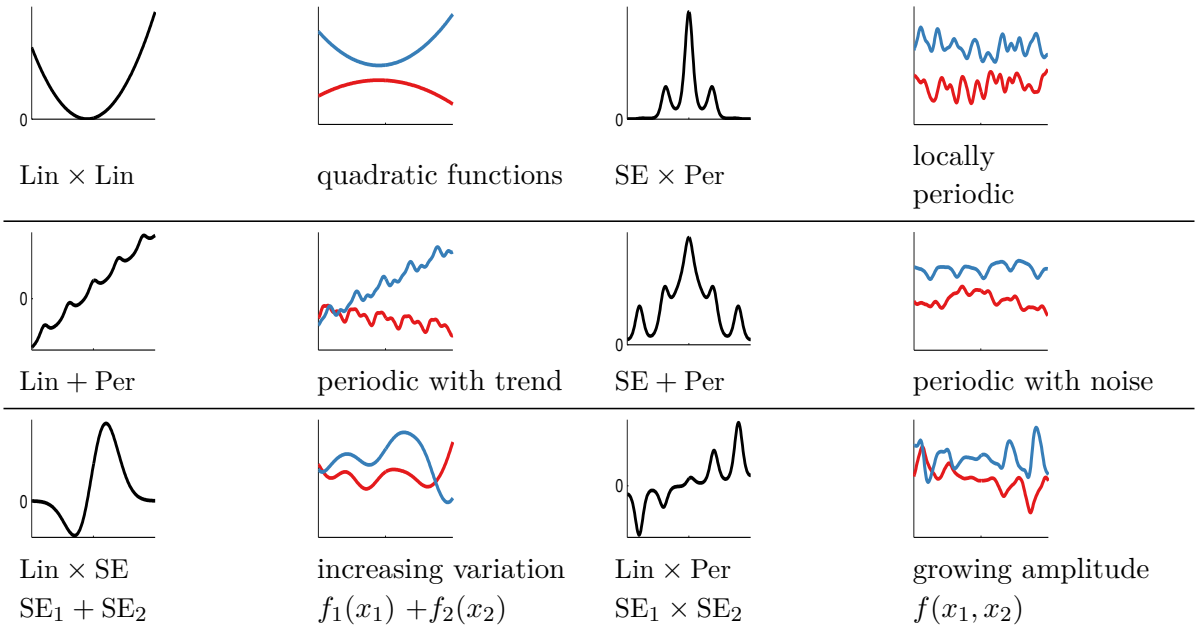


Fig. 1.2 Examples of one-dimensionalstructures expressible by composite kernels. Left column and third columns: composite kernels $k(\cdot, 0)$. Plots have same meaning as in figure 1.1.

1.1 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

$$f := f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2). \tag{1.3}$$

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

$$k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') = \phi_1(\mathbf{x})^\mathsf{T}\phi_1(\mathbf{x}') + \phi_2(\mathbf{x})^\mathsf{T}\phi_2(\mathbf{x}') \tag{1.4}$$

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 1.5, 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.

1.1.1 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's examine the joint prior over the sum of two functions. We'll distinguish between $f(\mathbf{x})$ (the function values at the training locations) and $f(\mathbf{x}^\star)$ (the function values at some other, query locations) so that it's clear which matrices to use to extrapolate.

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

$$\begin{bmatrix} \mathbf{f}_1(\mathbf{x}) \\ \mathbf{f}_1(\mathbf{x}^\star) \\ \mathbf{f}_2(\mathbf{x}) \\ \mathbf{f}_2(\mathbf{x}^\star) \\ \mathbf{f}_1(\mathbf{x}) + \mathbf{f}_2(\mathbf{x}) \\ \mathbf{f}_1(\mathbf{x}^\star) + \mathbf{f}_2(\mathbf{x}^\star) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_1^\star \\ \mu_2 \\ \mu_2^\star \\ \mu_1 + \mu_2 \\ \mu_1^\star + \mu_2^\star \end{bmatrix}, \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_1^\star & 0 & 0 & \mathbf{K}_1 & \mathbf{K}_1^\star \\ \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} & 0 & 0 & \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} \\ 0 & 0 & \mathbf{K}_2 & \mathbf{K}_2^\star & \mathbf{K}_2 & \mathbf{K}_2^\star \\ 0 & 0 & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} \\ \mathbf{K}_1 & \mathbf{K}_1^\star & \mathbf{K}_2 & \mathbf{K}_2^\star & \mathbf{K}_1 + \mathbf{K}_2 & \mathbf{K}_1^\star + \mathbf{K}_2^\star \\ \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} & \mathbf{K}_1^\star + \mathbf{K}_2^\star & \mathbf{K}_1^{\star\star} + \mathbf{K}_2^{\star\star} \end{bmatrix} \right) \tag{1.5}$$

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:

$$\mathbf{x}_A|\mathbf{x}_B \sim \mathcal{N}(\mu_A + \mathbf{\Sigma}_{AB}\mathbf{\Sigma}_{BB}^{-1}\left(\mathbf{x}_B - \mu_B\right), \mathbf{\Sigma}_{AA} - \mathbf{\Sigma}_{AB}\mathbf{\Sigma}_{BB}^{-1}\mathbf{\Sigma}_{BA}), \tag{1.6}$$

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

$$\mathbf{f}_1^\star|\mathbf{f} \sim \mathcal{N}\Big(\mu_1^\star + \mathbf{k}_1^{\star\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\left(\mathbf{f} - \mu_1 - \mu_2\right), \mathbf{k}_1^{\star\star} - \mathbf{k}_1^{\star\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\mathbf{k}_1^\star\Big) \tag{1.7}$$

We can even compute the posterior covariance between two components, conditioned on their sum:

$$\operatorname{cov}\left[\mathbf{f}_1^\star, \mathbf{f}_2^\star|\mathbf{f}\right] = -\mathbf{k}_1^{\star\mathsf{T}}(\mathbf{K}_1 + \mathbf{K}_2)^{-1}\mathbf{k}_2^\star \tag{1.8}$$

We conjecture that 1.8 is always negative.

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

1.1.2 Interpretability in High Dimensions

One of the chief advantages of additive models such as GAMs is their interpretability. Plate (1999) demonstrated 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 informative to plot the corresponding low-order functions, as in Figure 1.3.

R-squared = 0.9094

Additivity in multiple dimensions Figure 1.5 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 1.6 shows a dataset which 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

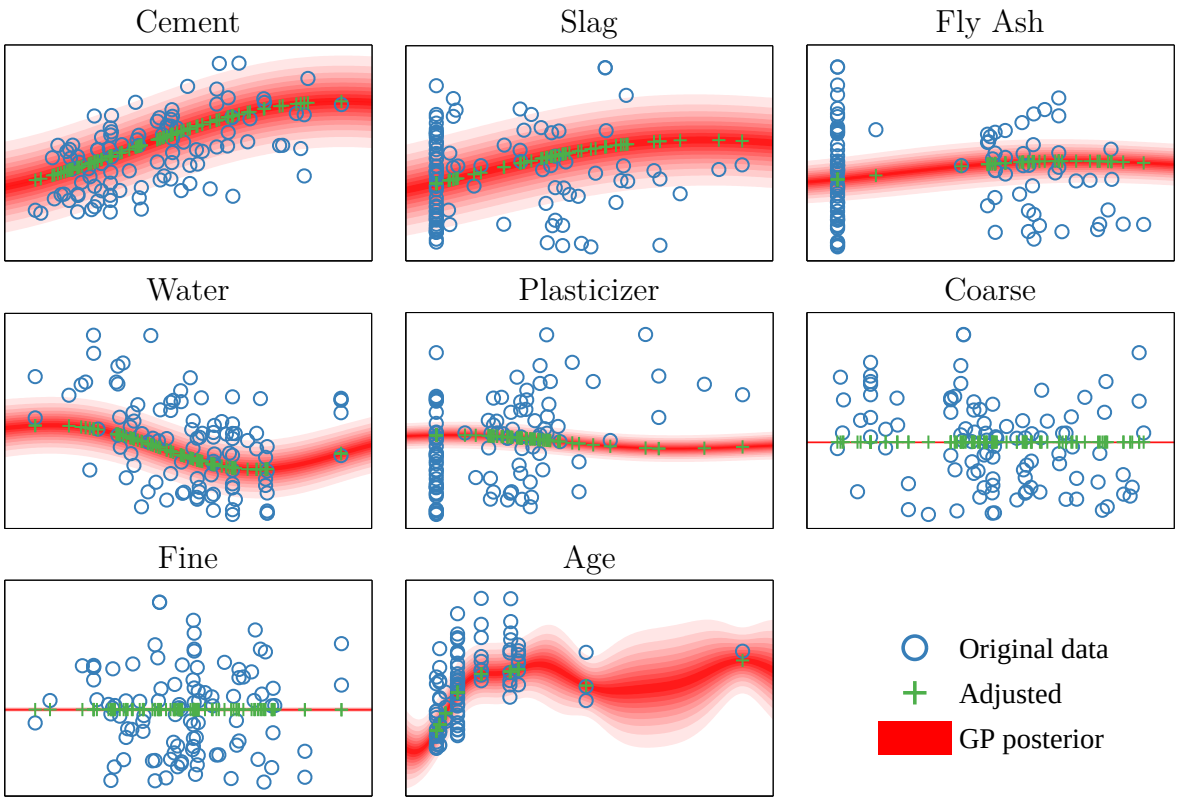


Fig. 1.3 By considering only one-dimensional terms of the additive kernel, we recover a form of Generalized Additive Model, and can plot the corresponding 1-dimensional functions. Blue points indicate the original data, green points are data after the contribution from all other terms has been subtracted. The vertical axis is the same for all plots.

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.

1.2 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 ker-

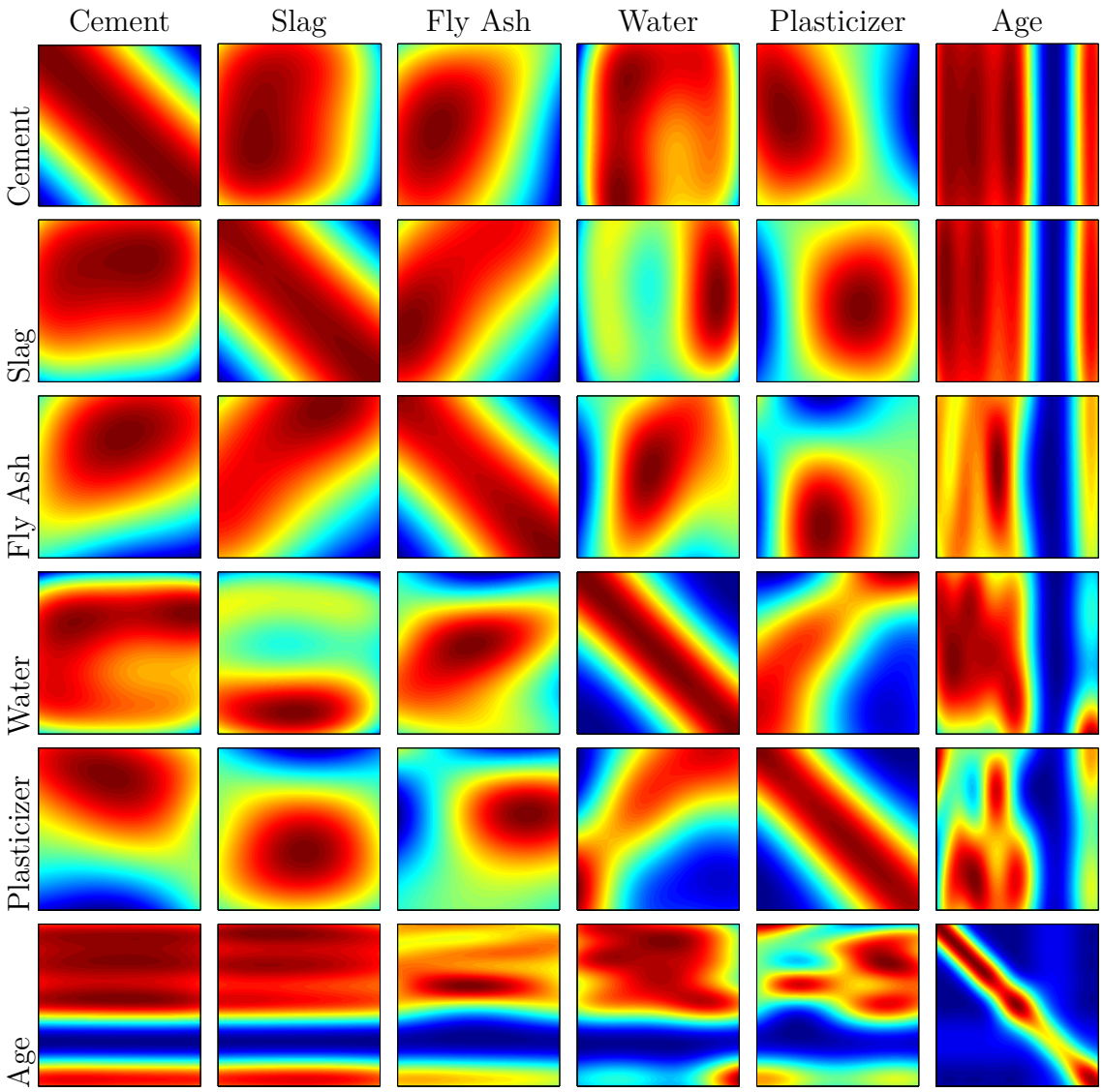


Fig. 1.4 two-way interactions on the concrete dataset. Plots on the diagonal show the posterior covariance of each function. Red indicates high covariance, blue indicates low. Off-diagonal plots show posterior covariance between each pair of functions, as a function of both inputs. Because each The 'Coarse' and 'Fine' dimensions are not shown here because their variance was zero.

nel 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 1.5.

Many architectures for learning complex functions, such as convolutional networks LeCun et al. (1989) and sum-product networks Poon and Domingos (2011), include units

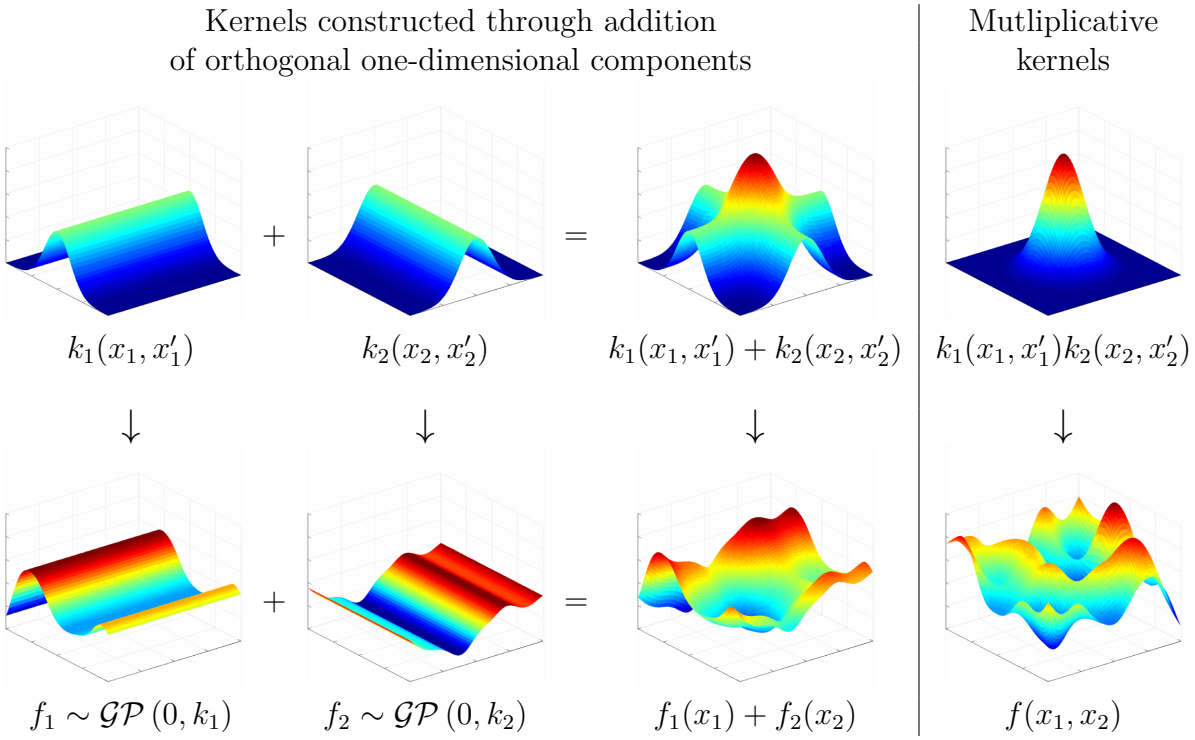
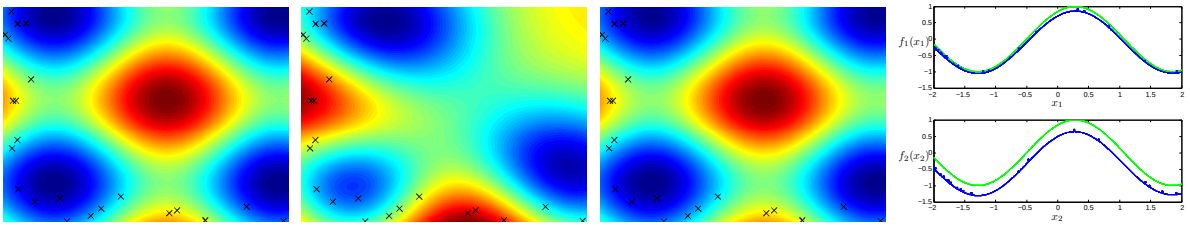


Fig. 1.5 Top left: An additive kernel is simply a sum of kernels. Bottom left: A draw from an additive kernel corresponds to a sum of draws from GPs with the corresponding kernels. Top right: a product kernel is a product of kernels. Bottom right: A draw from a product kernel does not correspond to a product of draws from the corresponding kernels.



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

Fig. 1.6 Long-range inference in functions with additive structure.

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

$$k_1(\mathbf{x}, \mathbf{x}') \times k_2(\mathbf{x}, \mathbf{x}') = [\phi_1(\mathbf{x}) + \phi_2(\mathbf{x})]^\mathsf{T} [\phi_1(\mathbf{x}') + \phi_2(\mathbf{x}')] \tag{1.9}$$

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

1.2.1 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.

1.2.2 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

1.3 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.

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:

$$f(x) = f(x_i + \tau_i) \tag{1.10}$$

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

$$k_{\text{periodic}}(x, x') = k(\sin(x), \cos(x), \sin(x'), \cos(x')) \tag{1.11}$$

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

Reflective Symmetry along an axis we can enforce the symmetry

$$f(x) = f(-x) \tag{1.12}$$

by the kernel transform

$$k_{\text{symm arg1}}(x, x') = k(x, x') + k(x, -x') + k(-x, x') + k(-x, -x') \tag{1.13}$$

Note that, because $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x}')$ for any PSD kernel, so we don't need to include the corresponding terms $k(-x, x')$ and $k(-x, -x')$.

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

$$f(x, y) = f(y, x) \tag{1.14}$$

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

$$k_{\text{reflect add}}(x, y, x', y') = k(x, y, x', y') + k(x, y, y', x') + k(y, x, x', y') + k(y, x, y', x') \tag{1.15}$$

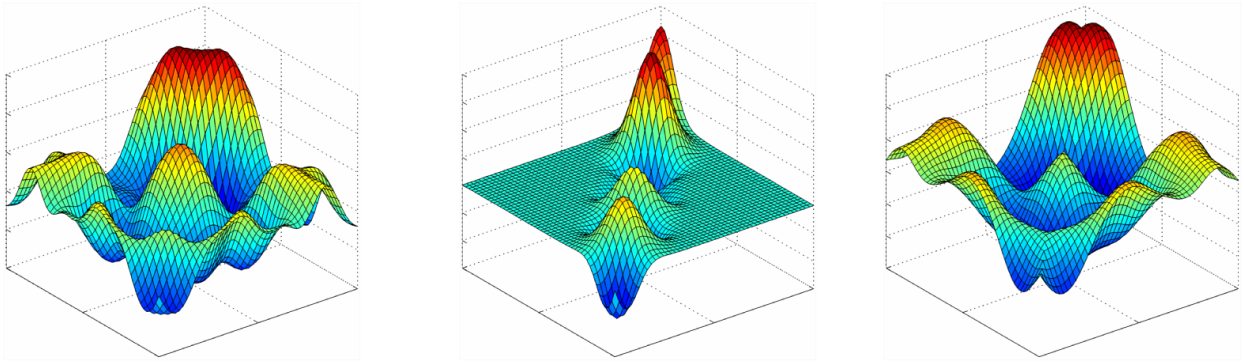
or by

$$k_{\text{reflect min}}(x, y, x', y') = k(\min(x, y), \max(x, y), \min(x', y'), \max(x', y')) \tag{1.16}$$

however, the second method will in general lead to non-differentiability along $x = y$. Figure 1.7 shows the difference.

1.3.1 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



Additive method	Product method	Representer method
$k(x,y,x',y') + k(x,y,y',x')$ $+ k(y,x,x',y') + k(y,x,y',x')$	$k(x,y,x',y') \times k(x,y,y',x')$ $\times k(y,x,x',y') \times k(y,x,y',x')$	$k(\min(x,y), \max(x,y),$ $\min(x',y'), \max(x',y'))$

Fig. 1.7 An illustration of three methods of introducing symmetry. Left: The additive method. Center: The product method. Right: The representer method. All three methods introduce a different type of nonstationarity.

1.3.2 Checking for Symmetries

In some situations, we might not know *a priori* whether a particular symmetry is present in a function. Because GPs let us build models both with and without certain symmetries, we can compute the amount of evidence that the data provide for each of these possibilities. To do so, we simple need to compare the marginal likelihood of the data We demonstrate that marginal likeihood an be used to automatically search over such structures.

1.4 Generating shapes with a given topology

In this section, we give a geometric illustration of the symmetries encoded by GPs with different sorts of kernels. The language of models of functions exhibiting symmetries, can be used to create a prior on surfaces, by warping a latent surface $\mathbf{x}$ to an observed surface $\mathbf{y} = \mathbf{f}(\mathbf{x})$. The distribution on $\mathbf{f}$ allows us to put mass on

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.

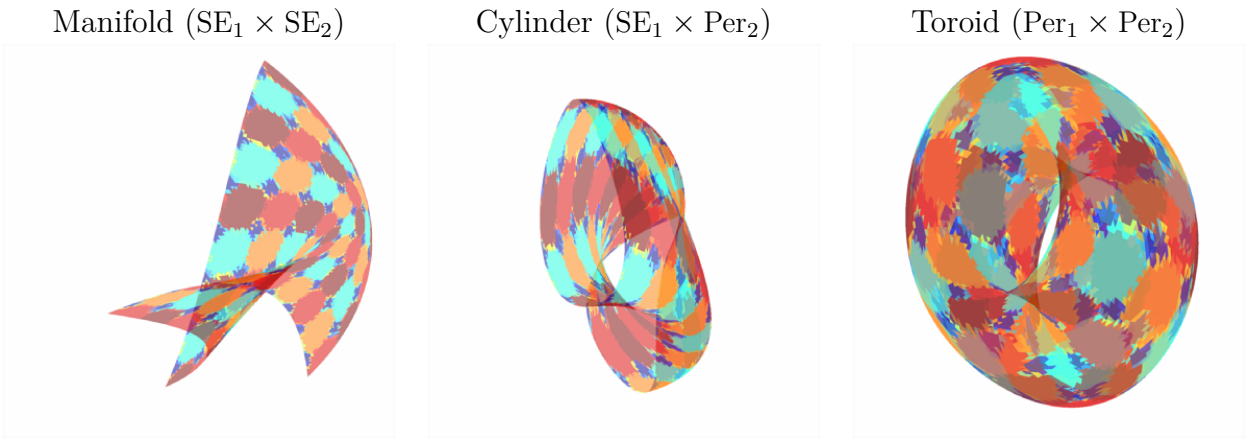


Fig. 1.8 Generating 2D manifolds with different topological structures. By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, the surfaces created have the corresponding topologies, ignoring self-intersections.

1.4.1 Möbius strips

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

$$f(x, y) = f(x, y + \tau_y) \tag{1.17}$$

$$f(x, y) = f(x + \tau_x, y) \tag{1.18}$$

$$f(x, y) = f(y, x) \tag{1.19}$$

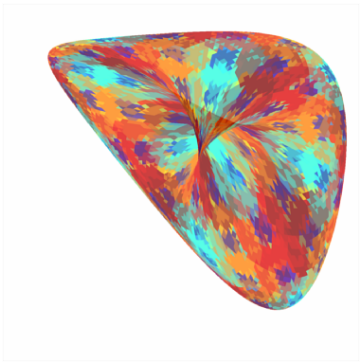
If we imagine moving along the edge of a Möbius strip, that is equivalent to moving along a diagonal in the function generated. Figure 1.9 shows this. The second example is doesn't resemble a typical Möbius strip because the edge of the mobius strip is in a geometric circle. This kind of embedding is 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

1.5 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.

Möbius manifold
($\text{Per}_1 \times \text{Per}_2$) and $f(x, y) = f(y, x)$



Sudanese Möbius strip
generated parametrically

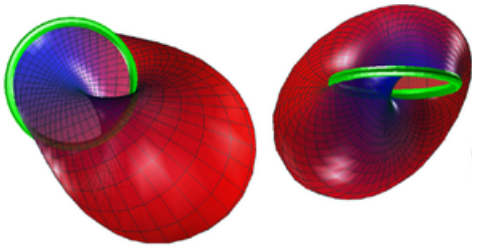


Fig. 1.9 Generating Möbius strips. Left: By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, surfaces sampled from the prior have topology corresponding to a Möbius strip. Möbius strips generated this way do not have the familiar shape of a circular flat surface with a half-twist; rather they tend to look like *Sudanese* Möbius strips (Lerner and Asimov, 1984), whose edge has a circular shape. Right: A Sudanese projection of a Möbius strip. Image adapted from (Commons, 2005).

1.5.1 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.

1.6 Examples

1.6.1 Computing molecular energies

Figure 1.10 gives one example of a function which obeys the same symmetries as a Möbius strip, in some subsets of its arguments.

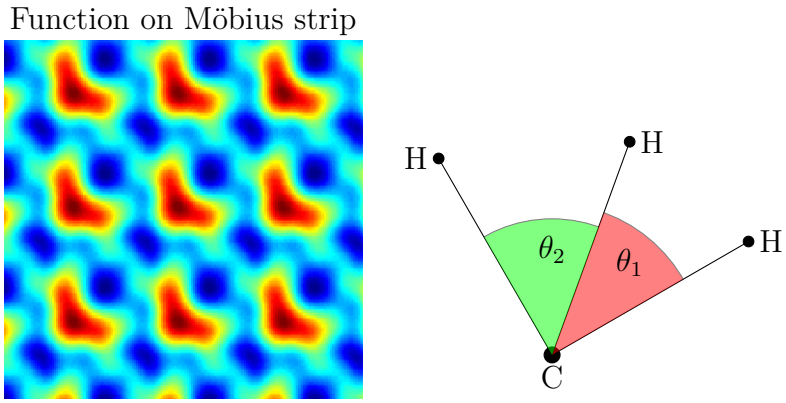


Fig. 1.10 An example of a function expressing the same symmetries as a Möbius strip in two of its arguments. The energy of a molecular configuration $f(\theta_1, \theta_2)$ depends only on the relative angles between atoms, and because each atom is indistiguishable, is invariant to permuting the atoms.

1.6.2 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:

$$f\left(\quad\right) = f\left(\quad\right) \tag{1.20}$$

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:

$$k\left((x_1, x_2, \ldots, x_D), (x'_1, x'_2, \ldots, x'_D)\right) = \sum_{i=1}^{D} \prod_{j=1}^{D} k(x_j, x'_{i+j \bmod D}) \tag{1.21}$$

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

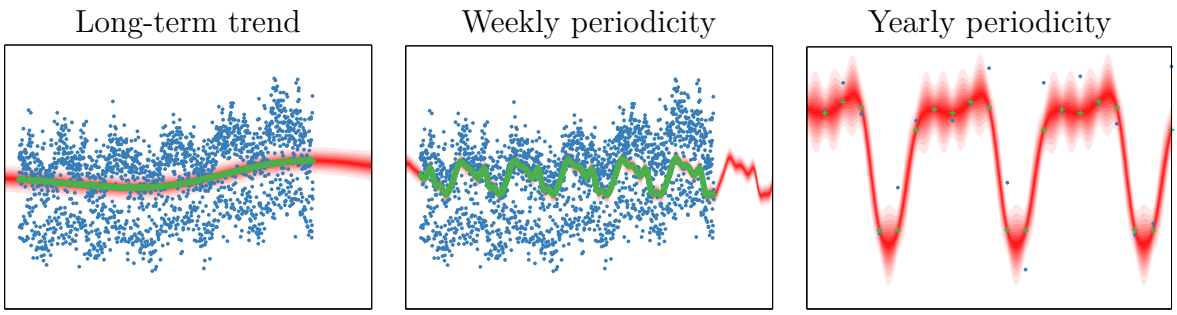


Fig. 1.11 A composite GP model of births data. (blue)

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:

$$k_{\text{conv}}\left((x_{11}, x_{12}, \ldots, x_{RC}), (x'_{11}, x'_{12}, \ldots, x'_{RC})\right) = \sum_{i=-L}^{L} \sum_{j=-L}^{L} k\left(\mathbf{x}, T_{ij}(\mathbf{x}')\right) \tag{1.22}$$

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 $\sqrt{k_{\text{conv}}(\mathbf{x}, \mathbf{x}) k_{\text{conv}}(\mathbf{x}', \mathbf{x}')}$.

1.7 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 $k(x, x') = k(x, T(x'))$.

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

1.8.1 Modeling multiple periodicities

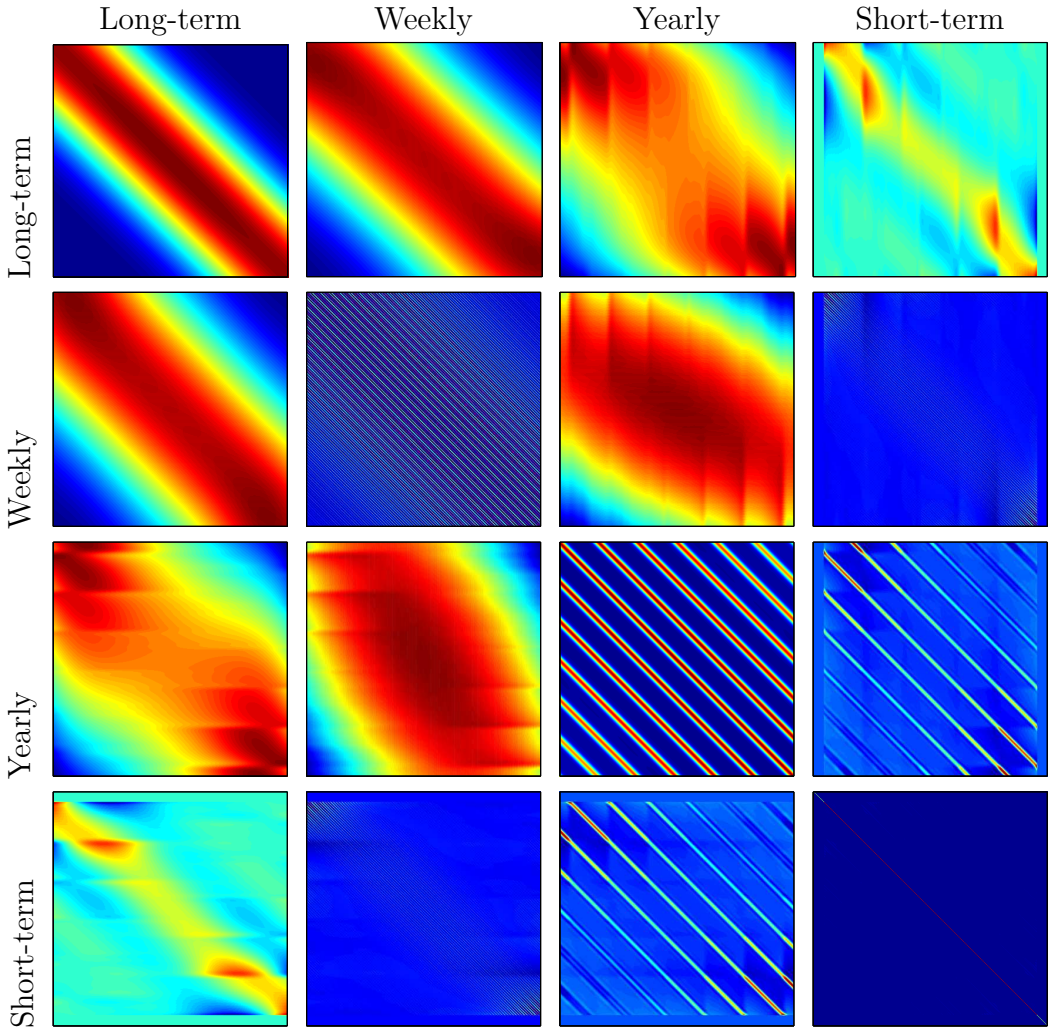


Fig. 1.12 Two-way interactions in births data

References

Wikimedia Commons. stereographic projection of a Sudanese Mobius band, 2005. URL http://commons.wikimedia.org/wiki/File:MobiusSnail2B.png. (page 14)

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 16)

T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990. (page 4)

N. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *The Journal of Machine Learning Research*, 6:1783–1816, 2005. (page 12)

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 7)

D. Lerner and D. Asimov. The Sudanese Mobius band. In *SIGGRAPH Electronic Theatre*, 1984. (page 14)

T.A. Plate. Accuracy versus interpretability in flexible modeling: Implementing a tradeoff using Gaussian process models. *Behaviormetrika*, 26:29–50, 1999. ISSN 0385-7417. (page 5)

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Conference on Uncertainty in AI*, pages 689–690. IEEE, 2011. (page 7)

C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*, volume 38. The MIT Press, Cambridge, MA, USA, 2006. (pages 1 and 10)