Chapter 1

Expressing Structure with Kernels

In this chapter, we'll show how to use kernels, also called *covariance functions*, to build many different kinds of models of functions. By combining a few simple kernels through addition and multiplication, we'll be able to express many different kinds of structure: additivity, symmetry, periodicity, interactions between variables, and change-points. We'll also show several ways to encode group invariances into kernels. Combining kernels in these simple ways gives us a rich, open-ended language of models.

1.1 Definition

Since we'll be discussing covariance functions at length, we now give a precise definition. A kernel $k(\mathbf{x}, \mathbf{x}') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is any positive-definite function between two points x, x' in some space $\mathcal{X}$. In this chapter, $\mathcal{X}$ is usually taken to be Euclidian space, but it can be any type of space, corresponding to spaces of images, documents, categories or points on a sphere.

Gaussian process models use a kernel to define the prior covariance between any two function values:

$$\mathrm{Cov}\left(f(\mathbf{x}), f(\mathbf{x}')\right) = k(\mathbf{x}, \mathbf{x}') \tag{1.1}$$

Colloquially, kernels are often said to specify the similarity between two objects. This is slightly misleading in this context, since what is actually being specified is the similarity between two values of a *function* over objects. The kernel specifies which functions are likely under the GP prior, which in turn determines the generalization properties of the model.

1.2 A Few Basic Kernels

To begin understanding the types of structures expressible by GPs, we'll start by briefly examining the types of structure encoded by a diverse set of commonly used kernels: the squared-exponential (SE), periodic (Per), and linear (Lin) kernels. These kernels are defined in Table 1.1.

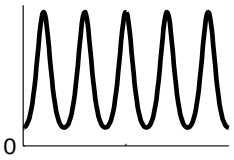
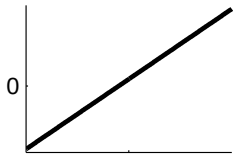
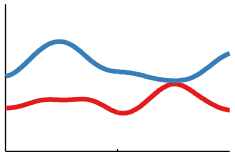
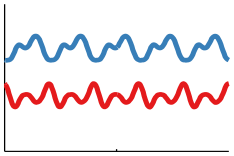
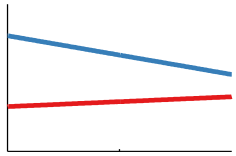
Kernel name:	Squared-exp (SE)	Periodic (Per)	Linear (Lin)
$k(x, x') =$	$\exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	$\exp\left(-\frac{2}{\ell^2}\sin^2\left(\pi\frac{x-x'}{p}\right)\right)$	$(x-c)(x'-c)$
Plot of kernel:			
	$x - x'$	$x - x'$	x (with $x' = 1$)
Samples from prior:			
Type of structure:	local variation	repeating structure	linear functions

Table 1.1 Examples of structures expressible by some basic kernels.

Each covariance function corresponds to a different set of assumptions made about the function we wish to model. For example, using a squared-exp (SE) kernel implies that the function we are modeling has infinitely many derivatives. There exist many variants of the SE kernel, each encoding slightly different assumptions of the smoothness of the underlying function.

Kernel parameters Each kernel has a number of parameters which specify the precise shape of the covariance function. These are sometimes referred to as *hyperparameters*, since they can be viewed as specifying a distribution over function parameters, instead of specifying the function directly.

Stationary and Non-stationary The SE and Per kernels are *stationary*, meaning that their value only depends on the difference $x - x'$. This implies that the probability of observing a particular dataset doesn't depend on the particular location of the dataset.

In contrast, the linear kernel Lin is non-stationary, meaning that the model will produce different answers if the data is moved around but the kernel parameters are kept fixed.

1.3 Combining Kernels

What if the kind of structure we need isn't expressed by any existing kernel? For many types of structure, it's possible to build a "made to order" kernel with the desired properties. The next few sections of this chapter will explore ways in which kernels can be combined to create new ones with different properties. This will allow us to include as much high-level structure as necessary into our models.

1.3.1 Notation

In this chapter, we'll focus on two ways of combining kernels: addition and multiplication. We'll often write these operations in shorthand, without arguments:

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

$$k_1 \times k_2 = k_1(\mathbf{x}, \mathbf{x}') \times k_2(\mathbf{x}, \mathbf{x}') \tag{1.3}$$

All of the basic kernels we considered in section 1.2 are one-dimensional, but kernels over multidimensional inputs can be constructed by adding and multiplying between kernels on different dimensions. The dimension on which a kernel operates is denoted by a subscript. For example, SE_2 represents an SE kernel over the second dimension of $\mathbf{x}$.

To remove clutter, we'll usually refer to a kernels without specifying their parameters.

1.4 Modeling Sums of Functions

An additive function is one which can be expressed as $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$. Additivity is a very useful modeling assumption in a wide variety of contexts, especially if it allows us to make stronger assumptions about the component functions in the sum. Restricting the flexibility of the component functions often aids in building interpretable models, and often enables extrapolation in high dimensions. Fortunately, additivity is easy to

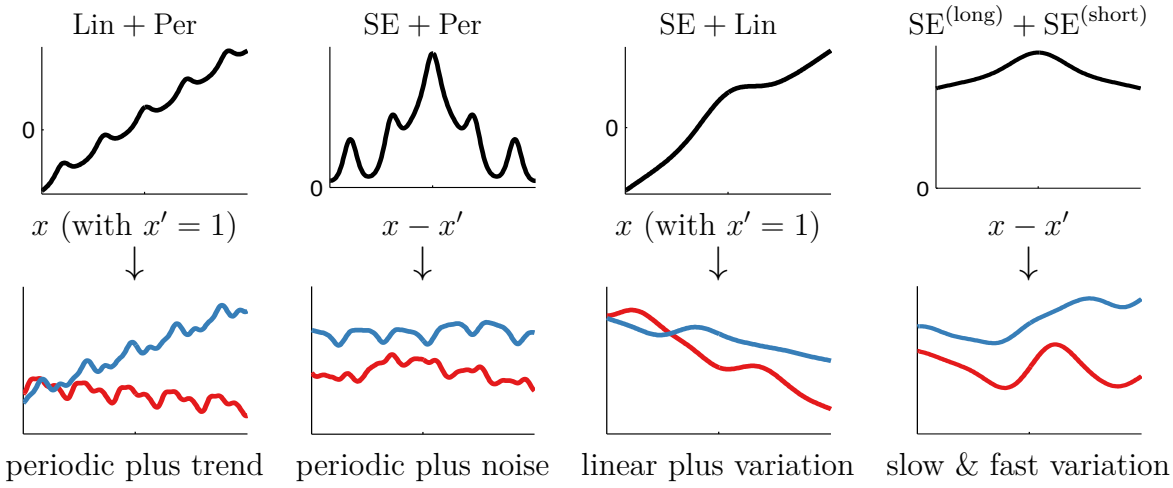


Table 1.2 Examples of one-dimensional structures expressible by adding kernels. Rows have the same meaning as in Table 1.1. $\mathrm{SE}^{(\mathrm{long})}$ denotes a SE kernel whose lengthscale is long relative to that of $\mathrm{SE}^{(\mathrm{short})}$

encode in GP models. Suppose functions f_1, f_2 are drawn independently from GP priors:

$$f_1 \sim \mathcal{GP}(\mu_1, k_1) \tag{1.4}$$
$$f_2 \sim \mathcal{GP}(\mu_2, k_2) \tag{1.5}$$

Then the sum of those functions is simply another GP:

$$f_1 + f_2 \sim \mathcal{GP}(\mu_1 + \mu_2, k_1 + k_2). \tag{1.6}$$

Kernels k_1 and k_2 can be kernels of different types, allowing us to model the data as a sum of independent functions, each possibly representing a different type of structure. We can also sum any number of components this way.

A known mean function can be moved into the covariance function

Specifically, if we wish to model an unknown function $f(\mathbf{x})$ with known mean $m(\mathbf{x})$, (with unknown magnitude $c \sim \mathcal{N}(0, \sigma_c^2)$), we can equivalently express this model using another GP with zero mean:

$$f \sim \mathcal{GP}(cm(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')), \quad c \sim \mathcal{N}\left(0, \sigma_c^2\right) \iff f \sim \mathcal{GP}\left(\mathbf{0}, c^2 m(\mathbf{x}) m(\mathbf{x}') + k(\mathbf{x}, \mathbf{x}')\right) \tag{1.7}$$

By moving the mean function into the covariance function, we get the same model, but we can integrate over the magnitude of the mean function at no additional cost. This is one advantage of moving as much structure as possible into the covariance function.

An unknown mean function can be moved into the covariance function If we wish to express our ignorance about the mean function, one way would be by putting a GP prior on it.

$$m \sim \mathcal{GP}(\mathbf{0}, k_1(\mathbf{x}, \mathbf{x}'))\,, \quad f \sim \mathcal{GP}(m(\mathbf{x}), k_2(\mathbf{x}, \mathbf{x}')) \iff f \sim \mathcal{GP}(\mathbf{0}, k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')) \tag{1.8}$$

1.4.1 Additivity Across Multiple Dimensions

When modeling functions of multiple dimensions, summing kernels can give rise to additive structure across different dimensions. To be more precise, if the kernels being added together are functions only of a subset of input dimensions, then the implied prior over functions decomposes in the same way. For example, if

$$f(x_1, x_2) \;\sim\; \mathcal{GP}(\mathbf{0}, k_1(x_1, x_1') + k_2(x_2, x_2')) \tag{1.9}$$

Then this is equivalent to the model

$$f_1(x_1) \;\sim\; \mathcal{GP}(\mathbf{0}, k_1(x_1, x_1')) \tag{1.10}$$

$$f_2(x_2) \;\sim\; \mathcal{GP}(\mathbf{0}, k_2(x_2, x_2')) \tag{1.11}$$

$$f(x_1, x_2) = f_1(x_1) + f_2(x_2) \tag{1.12}$$

Figure 1.1 illustrates a decomposition of this form. Note that the product of two kernels does not have an analogous interpretation as the product of two functions.

1.4.2 Example: Additive Model of Concrete Compressive Strength

To illustrate how additive kernels give rise to interpetable models, we'll build an additive model of the strength of concrete as a function of the amount of 7 different ingredients, plus the age of the concrete (Yeh, 1998).

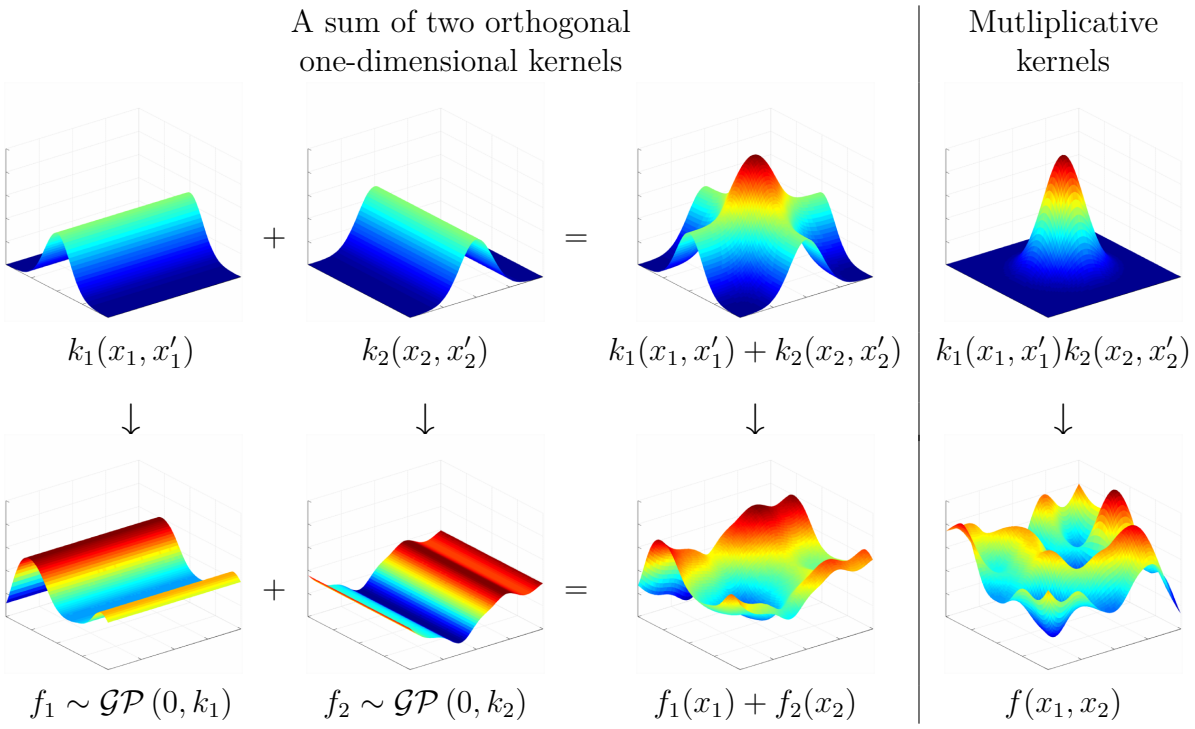


Fig. 1.1 *Top left:* An additive kernel is a sum of kernels. *Bottom left:* A draw from an additive kernel corresponds to a sum of draws from independent GP priors with the corresponding kernels. *Top right:* A product kernel. *Bottom right:* A GP prior with a product of kernels does not correspond to a product of draws from GPs.

Our simple additive model looks like

$$\begin{aligned} f(\mathbf{x}) = {} & f_1(\text{cement}) + f_2(\text{slag}) + f_3(\text{fly ash}) + f_4(\text{water}) \\ & + f_5(\text{plasticizer}) + f_6(\text{coarse}) + f_7(\text{fine}) + f_8(\text{age}) + \text{noise} \end{aligned} \tag{1.13}$$

where noise $\overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_n)$. After learning the kernel parameters by maximizing the marginal likelihood of the data, we can visualize the predictive distribution of each component of the model.

Figure 1.2 shows the marginal posterior distribution of each of the 8 components in Equation (1.13). We can see that the parameters controlling the variance of two of the components, Coarse and Fine, were set to zero, meaning that the marginal likelihood preferred a parsimonious model which did not depend on these dimensions. This is an example of the automatic sparsity that arises by maximizing marginal likelihood in GP models, also known as automatic relevance determination (ARD) (Neal, 1995). The ability to learn kernel parameters in this way is much more difficult when using non-

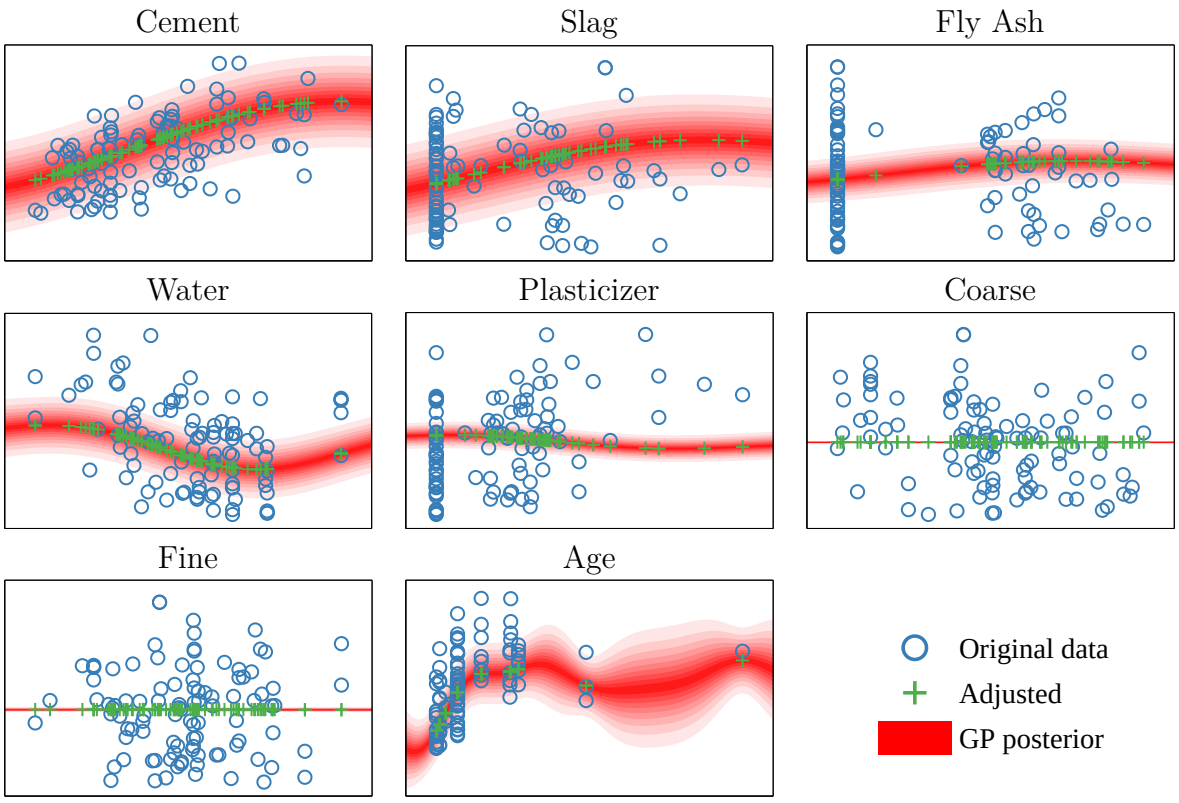


Fig. 1.2 By considering only one-dimensional terms of the additive kernel, we can plot the predictive distribution of each 1-dimensional function. Blue points indicate the original data, green points are data after the contribution from all other terms has been subtracted. The vertical axis is the same for all plots.

probabilistic methods such as Support Vector Machines [cite], for which cross-validation is often the best method to select kernel parameters.

1.4.3 Posterior Variance of Additive Components

How did we produce the plots shown in Figure 1.2? Here we derive the posterior variance and covariance of all of the additive components of a GP.

First, we'll write down the joint prior over the sum of two functions drawn from GP priors. We'll distinguish between $\boldsymbol{f}(\mathbf{X})$ (the function values at the training locations) and $\boldsymbol{f}(\mathbf{X}^\star)$ (the function values at some set of query locations).

If f_1 and f_2 are *a priori* independent, and $f_1 \sim \mathrm{GP}(\mu_1, k_1)$ and $f_2 \sim \mathrm{GP}(\mu_2, k_2)$, then

$$\begin{bmatrix} \boldsymbol{f}_1(\mathbf{X}) \\ \boldsymbol{f}_1(\mathbf{X}^\star) \\ \boldsymbol{f}_2(\mathbf{X}) \\ \boldsymbol{f}_2(\mathbf{X}^\star) \\ \boldsymbol{f}_1(\mathbf{X}) + \boldsymbol{f}_2(\mathbf{X}) \\ \boldsymbol{f}_1(\mathbf{X}^\star) + \boldsymbol{f}_2(\mathbf{X}^\star) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_1^\star \\ \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_2^\star \\ \boldsymbol{\mu}_1 + \boldsymbol{\mu}_2 \\ \boldsymbol{\mu}_1^\star + \boldsymbol{\mu}_2^\star \end{bmatrix}, \begin{bmatrix} \mathbf{K}_1 & \mathbf{K}_1^\star & 0 & 0 & \mathbf{K}_1 & \mathbf{K}_1^\star \\ \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} & 0 & 0 & \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} \\ 0 & 0 & \mathbf{K}_2 & \mathbf{K}_2^\star & \mathbf{K}_2 & \mathbf{K}_2^\star \\ 0 & 0 & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} \\ \mathbf{K}_1 & \mathbf{K}_1^\star & \mathbf{K}_2 & \mathbf{K}_2^\star & \mathbf{K}_1 + \mathbf{K}_2 & \mathbf{K}_1^\star + \mathbf{K}_2^\star \\ \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} & \mathbf{K}_1^\star + \mathbf{K}_2^\star & \mathbf{K}_1^{\star\star} + \mathbf{K}_2^{\star\star} \end{bmatrix} \right) \tag{1.14}$$

where we represent the Gram matrices, evaluated at all pairs of vectors in bold capitals as $\mathbf{K}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. So

$$\mathbf{K}_1 = k_1(\mathbf{X}, \mathbf{X}) \tag{1.15}$$

$$\mathbf{K}_1^\star = k_1(\mathbf{X}^\star, \mathbf{X}) \tag{1.16}$$

$$\mathbf{K}_1^{\star\star} = k_1(\mathbf{X}^\star, \mathbf{X}^\star) \tag{1.17}$$

By the formula for Gaussian conditionals, (given by **??**), we get that the conditional distribution of a GP-distributed function conditioned on its sum with another GP-distributed function is given by

$$\begin{aligned} \boldsymbol{f}_1(\mathbf{X}^\star) | \boldsymbol{f}_1(\mathbf{X}) + \boldsymbol{f}_2(\mathbf{X}) \sim \mathcal{N}\Big(& \boldsymbol{\mu}_1^\star + \mathbf{K}_1^{\star\mathsf{T}} (\mathbf{K}_1 + \mathbf{K}_2)^{-1} \left[\boldsymbol{f}_1(\mathbf{X}) + \boldsymbol{f}_2(\mathbf{X}) - \boldsymbol{\mu}_1 - \boldsymbol{\mu}_2 \right] \\ & \mathbf{K}_1^{\star\star} - \mathbf{K}_1^{\star\mathsf{T}} (\mathbf{K}_1 + \mathbf{K}_2)^{-1} \mathbf{K}_1^\star \Big) \end{aligned} \tag{1.18}$$

These formulae express the model's posterior uncertainty about the different components of the signal, integrating over the possible configurations of the other components. If we wish to condition on the sum of more than two functions, the term $\mathbf{K}_1 + \mathbf{K}_2$ can simply be replaced by $\sum_i \mathbf{K}_i$ everywhere.

Posterior Covariance of Additive Components

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

$$\mathrm{cov}\left[\boldsymbol{f}_1(\mathbf{X}^\star), \boldsymbol{f}_2(\mathbf{X}^\star) | \boldsymbol{f}(\mathbf{X}) \right] = -\mathbf{K}_1^{\star\mathsf{T}} (\mathbf{K}_1 + \mathbf{K}_2)^{-1} \mathbf{K}_2^\star \tag{1.19}$$

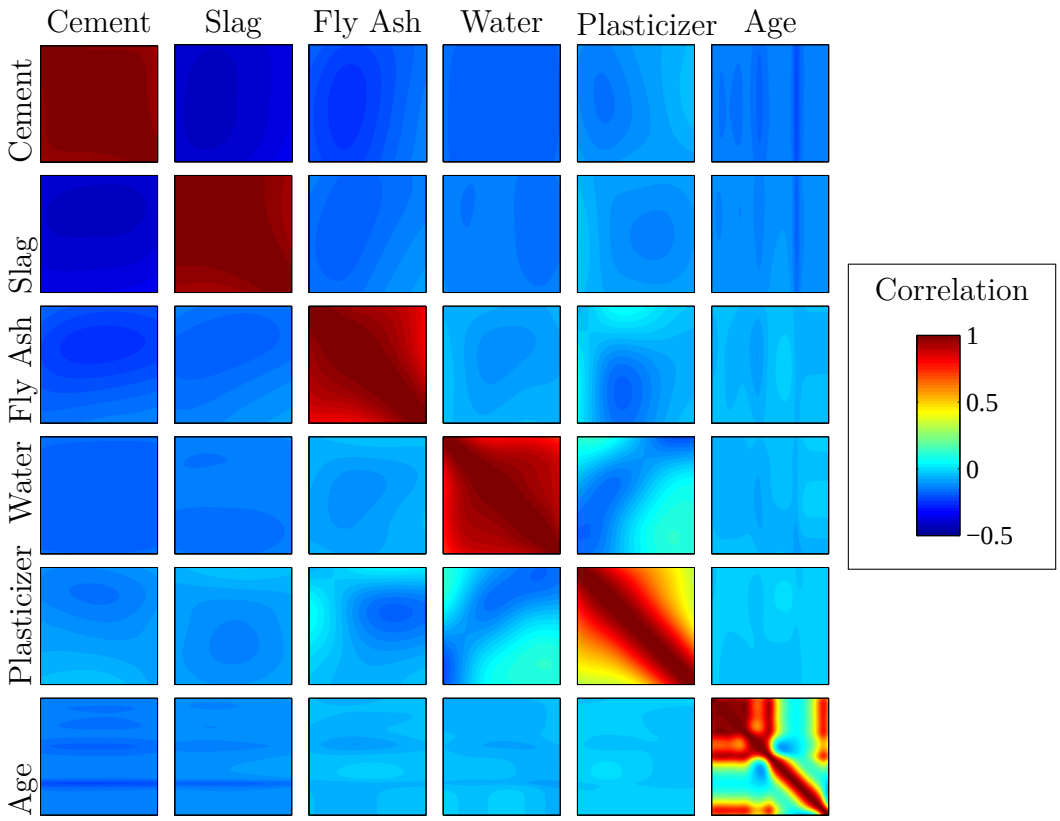


Fig. 1.3 Visualizing posterior correlations between the components explaining the concrete dataset. Each plot shows the additive model's posterior correlation between two components, plotted over the domain of the data ±5%. Red indicates high correlation, teal indicates no correlation, and blue indicates negative correlation. There is negative correlation between the "Cement" and "Slag" variables, meaning that - one of these functions is high and the other low, but which one is uncertain under the model. Dimensions 'Coarse' and 'Fine' are not shown, because their variance was zero.

If this quantity is negative, it means that there is ambiguity about which of the two components explains the data at that location. Figure 1.3 shows the posterior correlation between all non-zero components of the concrete model. We can see that there is negative correlation between the "Cement" and "Slag" variables. This reflects an ambiguity in the model about which one of these functions is high and the other low.

1.4.4 Long-range Extrapolation through Additivity

Additive structure often allows us to make predictions far from the training data. Figure 1.4 compares the extrapolations made by additive versus non-additive GP models, conditioned on data from a sum of two axis-aligned sine functions, evaluated in a small,

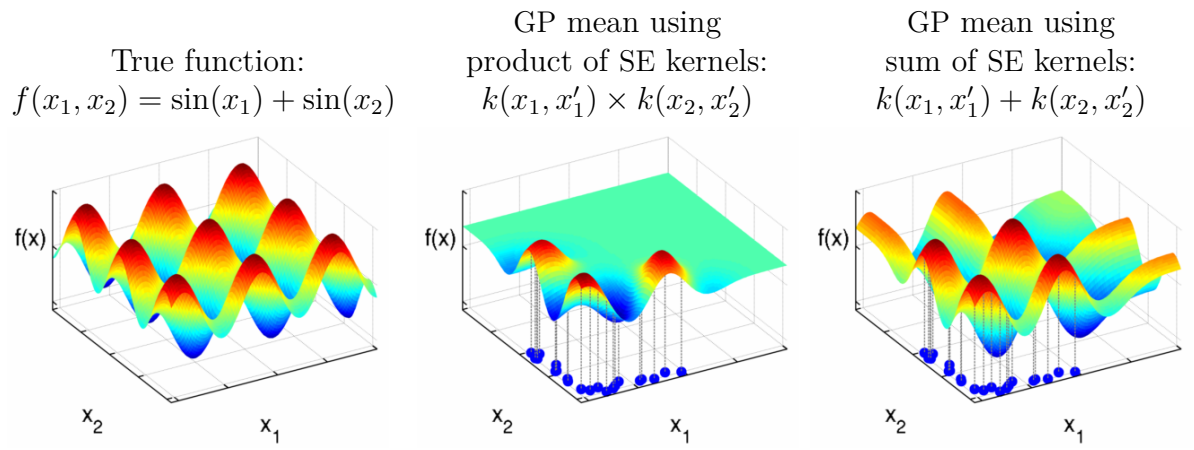


Fig. 1.4 Inference in functions with additive structure. When the function being modeled has additive structure, we can exploit this fact to extrapolate far from the training data. The product kernel allows a different function value for every combination of inputs, and so is uncertain about function values away from the training data.

L-shaped area. In this example, the additive model is able to correctly predict the height of the function at unseen combinations of inputs. The product-kernel model is more flexible, and so remains uncertain about the function away from the data.

These types of additive models have been well-explored in the statistics literature. For example, generalized additive models (Hastie and Tibshirani, 1990) have seen wide adoption. In high dimensions, we can also consider sums of functions of more than one dimension. Chapter 1.13 considers this model class in more detail.

1.5 Modifying Kernels through Multiplication

Multiplying kernels allows us to account for interactions between different input dimensions, or different notions of similarity. 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 1.1.

Products of SE kernels, having a different lengthscale parameter for each dimension, are equivalent to the SE-ARD kernel:

$$\text{SE-ARD}(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{D} \exp\left(-\frac{1}{2}\frac{(x_d - x_d')^2}{\ell_d^2}\right) = \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x_d')^2}{\ell_d^2}\right) \tag{1.20}$$

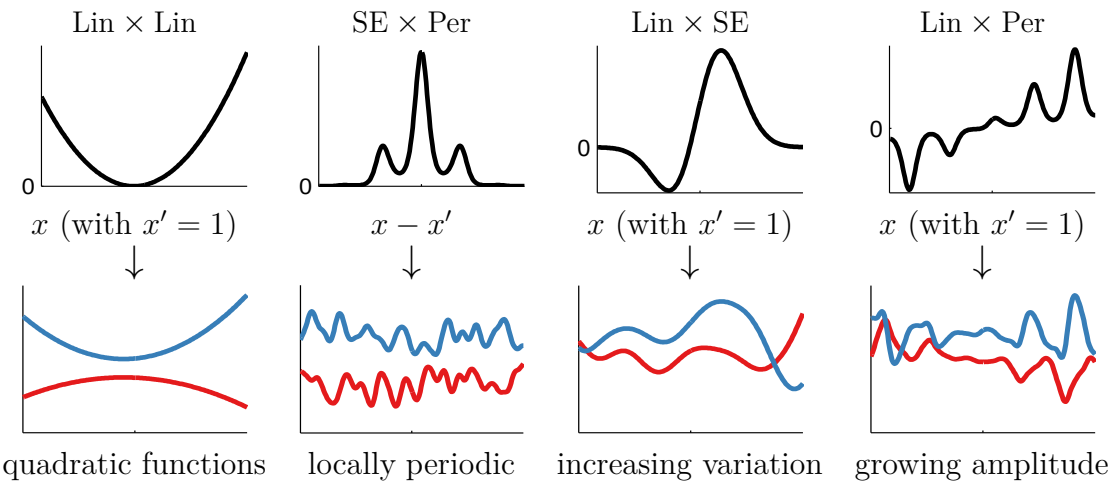


Fig. 1.5 Examples of one-dimensional structures expressible by multiplying kernels. Plots have same meaning as in figure 1.1.

Where ARD stands for Automatic Relevance Determination .

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.

1.6 Changepoints

A simple example of how multiplication of kernels can give rise to more structure priors is given by chapngepoint kernels. Changepoints can be defined through addition and multiplication with sigmoidal functions:

$$\mathrm{CP}(k_1, k_2) = k_1 \times \boldsymbol{\sigma} \; + \; k_2 \times \bar{\boldsymbol{\sigma}} \tag{1.21}$$

where $\boldsymbol{\sigma} = \sigma(x)\sigma(x')$ and $\bar{\boldsymbol{\sigma}} = (1 - \sigma(x))(1 - \sigma(x'))$. This compound kernel expresses a change from one kernel to another. The parameters of the sigmoid determine where, and how rapidly, this change occurs.

We can also express a function which changes structure within some interval - a change window - by replacing $\sigma(x)$ with a product of two sigmoids, one increasing and one decreasing.

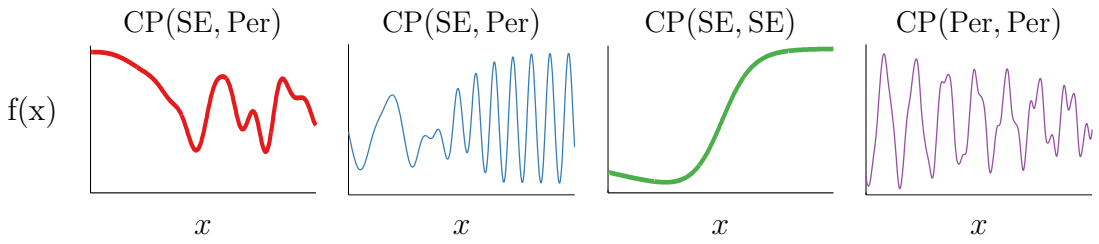


Fig. 1.6 Draws from different priors on changepoints, constructed by adding and multiplying together base kernels with sigmoidal functions.

1.6.1 Multiplication by a Known Function

More generally, we can model an unkown function that's been multiplied by some fixed, known function $a(x)$, by multiplying the kernel by $a(\mathbf{x})a(\mathbf{x}')$. Formally,

$$f(\mathbf{x}) = a(\mathbf{x})g(\mathbf{x}), \quad g \sim \mathcal{GP}(g \,|\, \mathbf{0}, k(\mathbf{x}, \mathbf{x}')) \quad \iff \quad f \sim \mathcal{GP}(f \,|\, \mathbf{0}, a(\mathbf{x})k(\mathbf{x}, \mathbf{x}')a(\mathbf{x}')) \tag{1.22}$$

1.7 Feature Representation

By Mercer's theorem (Mercer, 1909), any positive-definite kernel can be represented as the inner product between a fixed set of features, evaluated at x and at x':

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{h}(\mathbf{x})^{\mathsf{T}}\mathbf{h}(\mathbf{x}') \tag{1.23}$$

As a simple example, the squared-exponential kernel (SE) on the real line has a representation in terms of infinitely many radial-basis functions. Any stationary kernel (one which only depends on the distance between its inputs) on the real line can be represented by a set of sines and cosines - a Fourier representation (Bochner, 1959). In general, the feature representation of a kernel is not unique, and depends on which space $\mathcal{X}$ is being considered (Minh et al., 2006).

In some cases, $\mathcal{X}$ can even be the infinite-dimensional feature mapping of another kernel. Composing feature maps in this way leads to *deep kernels*, a topic explored in chapter 1.13.

1.7.1 Relation to Linear Regression

Suprisingly, GP regression is equivalent to Bayesian linear regression on $\mathbf{h}(\mathbf{x})$:

$$f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{h}(\mathbf{x}), \quad \mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \Sigma) \quad \iff \quad f \sim \mathcal{GP}\Big(f \,\Big|\, \mathbf{0}, \mathbf{h}(\mathbf{x})^\mathsf{T}\Sigma\mathbf{h}(\mathbf{x})\Big) \tag{1.24}$$

The link between Gaussian processes, linear regression, and neural networks is explored further in chapter 1.13.

1.7.2 Feature-space view of Combining Kernels

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.

We can also view kernel addition and multiplication as a combination of the features of the original kernels. For example, if we have two kernels

$$k_a(\mathbf{x}, \mathbf{x}') = \mathbf{a}(\mathbf{x})^\mathsf{T}\mathbf{a}(\mathbf{x}') \tag{1.25}$$

$$k_b(\mathbf{x}, \mathbf{x}') = \mathbf{b}(\mathbf{x})^\mathsf{T}\mathbf{b}(\mathbf{x}') \tag{1.26}$$

and we consider their addition, then

$$k_a(\mathbf{x}, \mathbf{x}') + k_b(\mathbf{x}, \mathbf{x}) = \mathbf{a}(\mathbf{x})^\mathsf{T}\mathbf{b}(\mathbf{x}') + \mathbf{a}(\mathbf{x})^\mathsf{T}\mathbf{b}(\mathbf{x}') \tag{1.27}$$

$$= \begin{bmatrix} \mathbf{a}(\mathbf{x}) \\ \mathbf{b}(\mathbf{x}) \end{bmatrix}^\mathsf{T} \begin{bmatrix} \mathbf{a}(\mathbf{x}') \\ \mathbf{b}(\mathbf{x}') \end{bmatrix} \tag{1.28}$$

meaning that the features of $k_a + k_b$ are the concatenation of the features of each kernel.

We can examine kernel multiplication in a similar way:

$$k_a(\mathbf{x}, \mathbf{x}') \times k_b(\mathbf{x}, \mathbf{x}') = \left[\mathbf{a}(\mathbf{x})^\mathsf{T}\mathbf{a}(\mathbf{x}')\right] \times \left[\mathbf{b}(\mathbf{x})^\mathsf{T}\mathbf{b}(\mathbf{x}')\right] \tag{1.29}$$

$$= \sum_i a_i(\mathbf{x})a_i(\mathbf{x}') \times \sum_j b_j(\mathbf{x})b_j(\mathbf{x}') \tag{1.30}$$

$$= \sum_i \sum_j a_i(\mathbf{x})a_i(\mathbf{x}')b_j(\mathbf{x})b_j(\mathbf{x}') \tag{1.31}$$

$$= \sum_{i,j} \left[a_i(\mathbf{x})b_j(\mathbf{x})\right]\left[a_i(\mathbf{x}')b_j(\mathbf{x}')\right] \tag{1.32}$$

$$= \operatorname{vec}\left(\mathbf{a}(\mathbf{x}) \otimes \mathbf{b}(\mathbf{x}')\right)^\mathsf{T} \operatorname{vec}\left(\mathbf{a}(\mathbf{x}) \otimes \mathbf{b}(\mathbf{x}')\right) \tag{1.33}$$

In other words, the features of $k_a \times k_b$ are just the cartesian product (all possible combinations) of the original two sets of features. For example, a set of infinitely many Gaussian bumps spread evenly along the real line gives rise to one-dimensional SE kernel. The cartesian product of these features with another set spread along a different dimension gives a tiling of the plane with two-dimensional Gaussian bumps. This tiling corresponds to a two-dimensional SE kernel.

1.8 Expressing Symmetries and Invariances

When modeling functions, encoding known symmetries improves both learning and prediction. Many types of symmetry can be enforced through operations on the covariance function. In this section, we'll look at different ways in which we can encode a given symmetry into a prior on functions.

We'll demonstrate the properties of the resulting models by sampling functions from their priors. By using these priors to define warpings from $\mathbb{R}^2 \to \mathbb{R}^3$, we'll also show how to build a nonparametric prior on an open-ended family of topological manifolds, such as cylinders, torii, and Möbius strips.

Ginsbourger et al. (2012) and Ginsbourger et al. (2013) characterized the set of GP priors on functions invariant to given transformations. They showed that the only way to construct a prior on functions which respect a given invariance is to construct a kernel which respects the same invariance with respect to each of its two inputs.

Formally, given a finite group of operations G to which we wish our function to remain invariant, and $f \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$, f is invariant under G if and only if $k(\cdot, \cdot)$ is

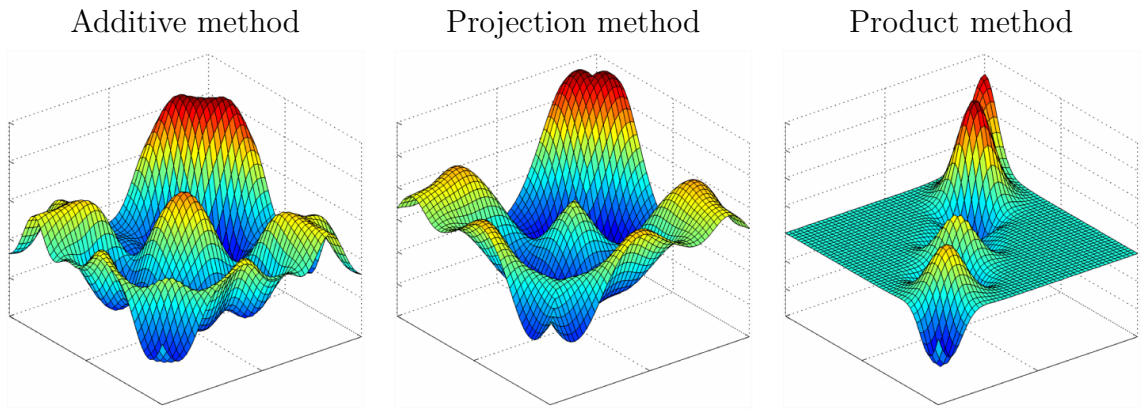


Fig. 1.7 Three methods of introducing symmetry, illustrated through draws from the corresponding priors. All three methods introduce a different type of nonstationarity.

argumentwise invariant:

$$k(g(\mathbf{x}), g(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}'), \quad \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \quad \forall g, g' \in G \tag{1.34}$$

As a simple example, consider the symmetry $f(x, y) = f(y, x)$, or the set of functions invariant to swapping their two arguments. The elements of the group G_{swap} describing this symmetry are

$$g_1(f(x, y)) = f(x, y) \qquad \text{(identity)} \tag{1.35}$$

$$g_2(f(x, y)) = f(y, x) \qquad \text{(swap)} \tag{1.36}$$

However, it might not be clear how to find a kernel obeying these symmetries.

1.8.1 Three Recipes for Group Invariance

Fortunately, for finite groups, there are a few simple ways to transform any kernel into one which is argumentwise invariant to actions under any finite group:

1. **Sum over the Orbit.** Ginsbourger et al. (2012) and Kondor (2008) suggest a double sum over the orbits of $\mathbf{x}$ and $\mathbf{x}'$ w.r.t. G:

$$k_{\text{sum}}(\mathbf{x}, \mathbf{x}') = \sum_{g, \in G} \sum_{g' \in G} k(g(\mathbf{x}), g'(\mathbf{x}')) \tag{1.37}$$

For our example group G_{swap}, this operation results in the kernel:

$$k_{\text{switch}}(\mathbf{x}, \mathbf{x}') = \sum_{g, \in G_{\text{swap}}} \sum_{g' \in G_{\text{swap}}} k(g(\mathbf{x}), g'(\mathbf{x}')) \tag{1.38}$$

$$= k(x, y, x', y') + k(x, y, y', x') + k(y, x, x', y') + k(y, x, y', x') \tag{1.39}$$

For stationary kernels, some pairs of elements in this sum will be identical, and can be ignored. Figure 1.7(a) shows a draw from a GP prior with an SE kernel symmetrized in this way. This construction has the property that the marginal variance is doubled near $x = y$, which may or may not be desirable.

2. **Project onto a Fundamental Domain.** Ginsbourger et al. (2013) also explore the possibility of projecting each datapoint into a fundamental domain of the group, using a mapping A_G:

$$k_{\text{rep}}(\mathbf{x}, \mathbf{x}') = k(A_G(\mathbf{x}), A_G(\mathbf{x}')) \tag{1.40}$$

For our example group G_{swap}, a fundamental domain is $\{x, y : x < y\}$, which can be mapped to using $A_{G_{\text{swap}}}(x, y) = \Big[\min(x, y), \max(x, y)\Big]$. Constructing a kernel using this method introduces a non-differentiable "seam" along $x = y$, as shown in Figure 1.7(b). The projection method also works for infinite groups, as we shall see below.

3. **Multiply over the Orbit.** Ryan P. Adams (personal communication) suggests a construction using products over the orbits:

$$k_{\text{sum}}(\mathbf{x}, \mathbf{x}') = \prod_{g, \in G} \prod_{g' \in G} k(g(\mathbf{x}), g'(\mathbf{x}')) \tag{1.41}$$

This method will often produce GP priors with zero variance in some regions of the space, as in Figure 1.7(c).

There are many possible ways to achive a given symmetry, but we must be careful to do so without compromising other qualities of the model we are constructing. For example, simply setting $k(\mathbf{x}, \mathbf{x}') = 0$ gives rise to a GP prior which obeys *all possible* symmetries, but this is presumably not a model we wish to use.

1.8.2 Periodicity

Periodicity in a one-dimensional function corresponds to the invariance

$$f(x) = f(x + \tau) \tag{1.42}$$

where τ is the period.

The most popular method for building a periodic kernel is due to MacKay (1998), who used the projection method in combination with an SE kernel. A fundamental domain of the symmetry group is a circle, so the kernel

$$\mathrm{Per}(x, x') = \mathrm{SE}\left(\left[\sin(x), \cos(x)\right], \left[\sin(x'), \cos(x')\right]\right) \tag{1.43}$$

achieves the invariance in Equation (1.42). Simple algebra reduces this kernel to the form shown in Table 1.1.

We could also build a periodic kernel with period τ by the mapping $A(x) = \mathrm{mod}(x, \tau)$. However, samples from this prior would be discontinuous at every integer multiple of τ.

1.8.3 Reflective Symmetry Along an Axis

We can enforce symmetry about zero

$$f(x) = f(-x) \tag{1.44}$$

using the sum over orbits method, by the transform

$$k_{\text{symm axis}}(x, x') = k(x, x') + k(x, -x') + k(-x, x') + k(-x, -x') \tag{1.45}$$

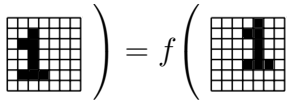
1.8.4 Translation Invariance in Images

Most models of images are invariant to spatial translations (LeCun and Bengio, 1995). Similarily, most models of sounds are also invariant to translation through time.

This sort of translation-invariance is completely distinct from the stationarity of kernels such as SE or Per. A stationary kernel implies that the prior is invariant to translations of the entire training and test set. In contrast, we are discussing here a discretized input space (into pixels or the audio equivalent), where the input vectors have one dimension for each pixel. We are interested in creating priors on functions that

are invariant to shifting a signal along its pixels:

$$f\left(\quad\right) = f\left(\quad\right) \tag{1.46}$$



In this setting, translation is equivalent to swapping dimensions of the input vector $\mathbf{x}$. For example, in a one-dimensional image, translation could be define as

$$\text{shift}(\mathbf{x}, i) = \left[x_{\text{mod}(i+1,D)}, x_{\text{mod}(i+2,D)}, \ldots, x_{\text{mod}(D+i,D)}\right] \tag{1.47}$$

The extension to two dimensions is straightforward, but notationally cumbersome.

Translation invariance in one dimension can be achieved by the transformation

$$k_{\text{invariant}}\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{i=1}^{D}\sum_{j=1}^{D} k(\text{shift}(\mathbf{x}, i), \text{shift}(\mathbf{x}, j)) \tag{1.48}$$

We are simply defining the covariance between two images to be the sum of all covariances between all translations of the two images.

Kondor (2008) built a more elaborate kernel between images, approximately invariant to both translation and rotation by using the projection method.

1.9 Generating Topological Manifolds

We now give a geometric illustration of the symmetries encoded by different combinations of kernels.

Priors on functions exhibiting symmetries can be used to create a prior on topological manifolds, by warping a latent surface $\mathbf{x}$ to an observed surface $\mathbf{y} = f(\mathbf{x})$. To build a prior on 2-dimensional manifolds embedded in 3-dimensional space, we simply need a prior on mappings from $\mathbb{R}^2$ to $\mathbb{R}^3$, which we can construct using 3 independent functions $[f_1(\mathbf{x}), f_2(\mathbf{x}), f_3(\mathbf{x})]$ mapping from $\mathbb{R}^2$ to $\mathbb{R}$, specified with GP priors. Symmetries in $[f_1, f_2, f_3]$ will connect different parts of the manifolds, giving rise to non-trivial topologies on the sampled surfaces.

This construction is similar in spirit to the GP latent variable model (GP-LVM) of Lawrence (2005), which learns a latent embedding of the data into a low-dimensional space, given a GP prior on the mapping from the latent space to the observed space.

Figure 1.8 shows 2D meshes warped into 3D by functions drawn from GP priors with different kernels. The different kernels give rise to different topologies.

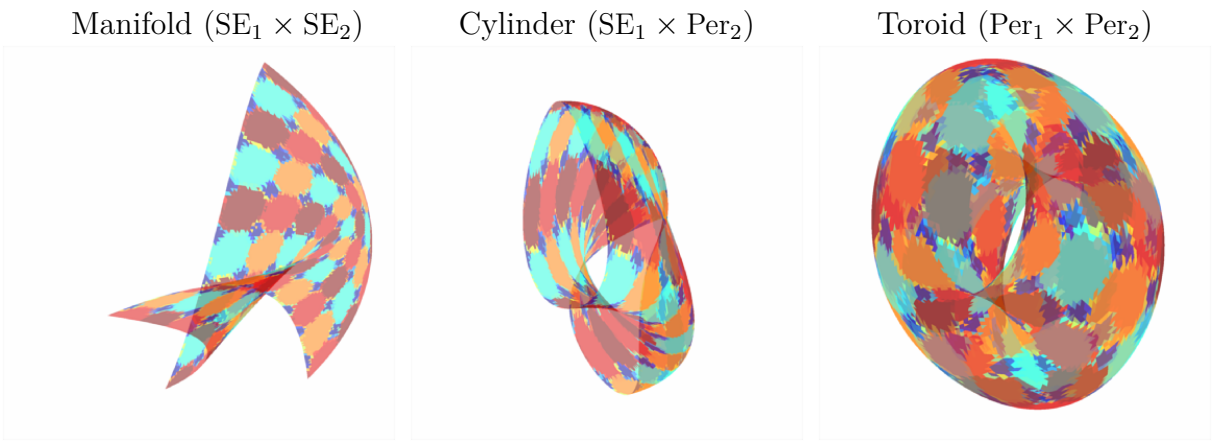


Fig. 1.8 Generating 2D manifolds with different topological structures. By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, the surfaces created have the corresponding topologies (ignoring self-intersections).

1.9.1 Möbius Strips

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

$$f(x, y) = f(x, y + \tau_y) \tag{1.49}$$

$$f(x, y) = f(x + \tau_x, y) \tag{1.50}$$

$$f(x, y) = f(y, x) \tag{1.51}$$

If we imagine moving along the edge of a Möbius strip, that is equivalent to moving along a diagonal in the function generated. Figure 1.9a shows an example of a function drawn from such a prior. Figure 1.9b shows an example of a 2D mesh mapped to 3D by functions drawn from such a prior. This surface doesn't resemble a typical Möbius strip, because the edge of the mobius strip is in roughly circular shape, as opposed to the double-loop that one obtains by gluing a strip of paper with a single twist. The surface shown resembles the Sudanese Möbius strip (Lerner and Asimov, 1984), shown in Figure 1.9c.

1.10 Kernels on Categorical Variables

One flexible way to build a kernel over categorical variables is simply to represent your categorical variable by a one-of-k encoding. For example, if $x \in \{1, 2, 3, 4, 5\}$,

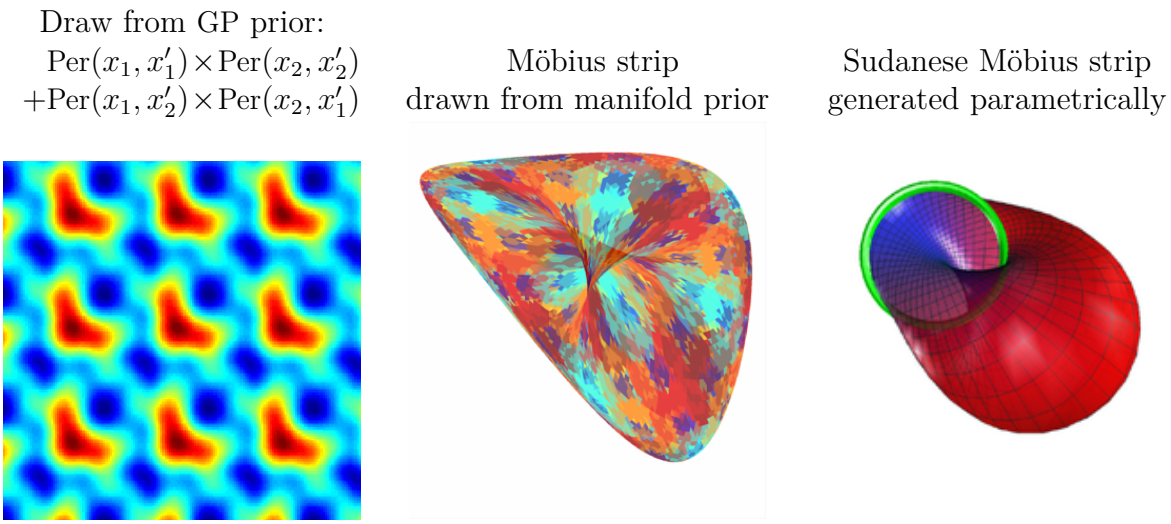


Fig. 1.9 Generating Möbius strips. *Left:* A function drawn from a GP prior obeying the same symmetries as a Möbius strip. *Center:* By enforcing that the functions mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$ obey the appropriate symmetries, surfaces sampled from the prior have topology corresponding to a Möbius strip. Möbius strips generated this way do not have the familiar shape of a circular flat surface with a half-twist; instead 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).

one-of-k$(3) = [0, 0, 1, 0, 0]$.

$$k_{\text{categorical}}(x, x') = \text{SE-ARD}(\text{one-of-k}(x), \text{one-of-k}(x')) \tag{1.52}$$

Short lengthscales for any particular dimension in the SE-ARD kernel indicate that that category is dissimilar to all others.

A more flexible parameterization suggested by Kevin Swersky (personal communication) allows complete flexibility about which pairs of categories are similar to one another, replacing the SE-ARD kernel with the fully-parameterized SE-full:

$$\text{SE-full}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{L}\mathbf{x}'\right) \tag{1.53}$$

where L is symmetric. This kernel individually parameterizes the covariance between each pair of categories.

1.11 Learning Kernel Parameters

One difficulty in building GP models is choosing, or integrating over, the kernel parameters. Fortunately, typical kernels only have $\mathcal{O}(D)$ parameters, meaning that if N is reasonably large, these parameters can be estimated by maximum marginal likelihood. For a fixed kernel form, these parameters can be optimized by gradient-based methods.

1.12 Automatically Choosing a Kernel

The marginal likelihood can also be used to select the form of the kernel.

For example, we might not know whether a particular structure or symmetry is present in the function we are trying to model. By building kernels with and without such structure, we can compute the marginal likelihoods of the corresponding GP models. The quantities represent the relative amount of evidence that the data provide for each of these possibilities, providing the assumptions of the model are correct.

1.13 Conclusion

We've seen that kernels are a flexible and powerful language for building models of different types of functions. However, for a given problem, it can difficult to specify an appropriate kernel, even after looking at the data. A better procedure would be to compare the predictive performance, or marginal likelihood, of a few different kernels. However, it might be difficult to enumerate all plausible kernels, and tedious to search over them.

Analogously, we usually don't expect to simply guess the best value of some parameter. Rather, we specify a search space and an objective, and ask the computer to the search this space for us. In the next chapter, we'll see how to perform such a search an automatic search over the open-ended, discrete space of kernel expressions.

References

Salomon Bochner. *Lectures on Fourier integrals*, volume 42. Princeton University Press, 1959. (page 12)

Wikimedia Commons. Stereographic projection of a Sudanese Möbius band, 2005. URL http://commons.wikimedia.org/wiki/File:MobiusSnail2B.png. (page 20)

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. (pages 14 and 16)

David Ginsbourger, Xavier Bay, Olivier Roustant, Laurent Carraro, et al. Argumentwise invariant kernels for the approximation of invariant functions. In *Annales de la Faculté de Sciences de Toulouse*, number 3, 2012. (pages 14 and 15)

T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990. (page 10)

Imre Risi Kondor. *Group theoretical methods in machine learning*. PhD thesis, Columbia University, 2008. (pages 15 and 18)

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 18)

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 13)

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995. (page 17)

D. Lerner and D. Asimov. The Sudanese Möbius band. In *SIGGRAPH Electronic Theatre*, 1984. (pages 19 and 20)

David JC MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998. (page 17)

James Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, pages 415–446, 1909. (page 12)

Ha Quang Minh, Partha Niyogi, and Yuan Yao. Mercer's theorem, feature maps, and smoothing. In *Learning theory*, pages 154–168. Springer, 2006. (page 12)

Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995. (page 6)

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Conference on Uncertainty in AI*, pages 689–690. IEEE, 2011. (page 13)

I-C Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808, 1998. (page 5)