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 many types of invariances. Combining kernels in a few simple ways gives us a rich language with which to build appropriate models.

1.1 Definition

Since we'll be discussing covariance functions at length, we now give a precise definition. A kernel $k(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 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, it can also be the space of images, documents, categories or points on a sphere, for example.

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

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

Colloquially, kernels are often said to specify the similarity between two objects. This is slightly misleading in the context of GPs, 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   Basic Kernels

Commonly used kernels families include the squared exponential (SE), periodic (Per), linear (Lin), and rational quadratic (RQ). These kernels are defined in Table 1.1. There is nothing special about these kernels in particular, except that they represent a
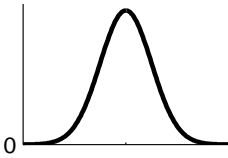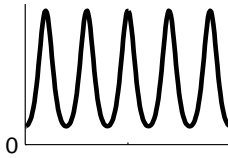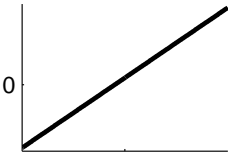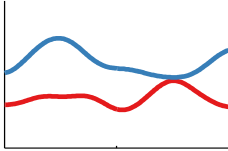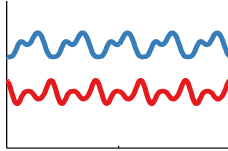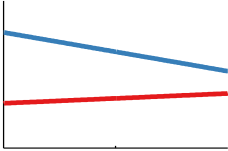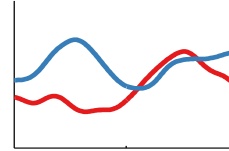
Squared-exp (SE)	Periodic (Per)	Linear (Lin)	Rational quadratic (RQ)
$\exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$	$\exp\left(-\frac{2}{\ell^2}\sin^2\left(\pi\frac{x-x'}{p}\right)\right)$	$(x-c)(x'-c)$	$\left(1+\frac{(x-x')^2}{2\alpha\ell^2}\right)^{-\alpha}$

| local variation | repeating structure | linear functions | multi-scale variation |

Table 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.

diverse set.

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.

1.3   Combining Kernels

The rest of this chapter will explore ways in which kernels can be combined to create new ones with different properties. Using a few simple base kernels and operations for combining them will give us a rich language with which to build models with diverse sorts of properties.
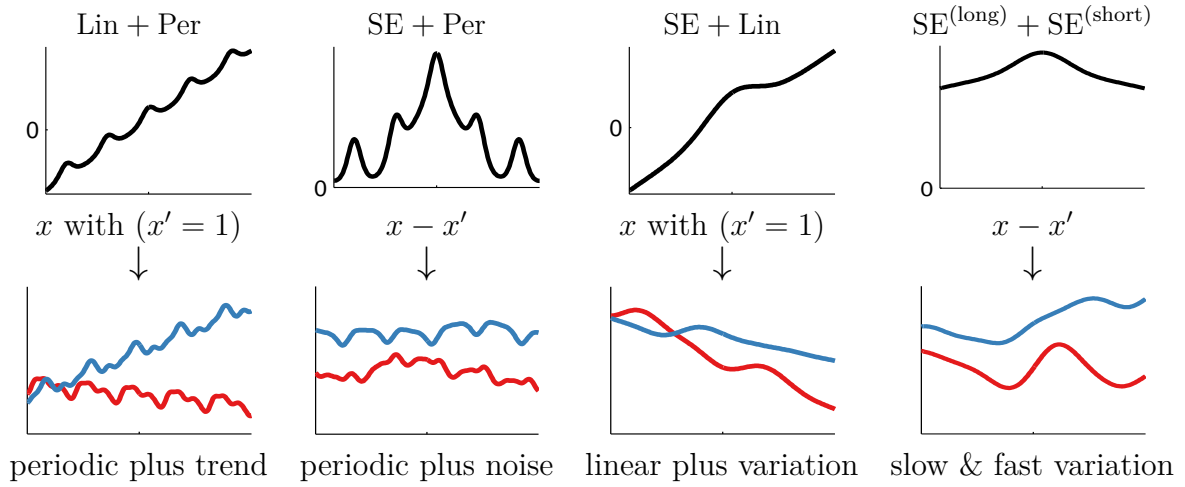
Fig. 1.1 Examples of one-dimensional structures expressible by adding kernels. Plots have same meaning as in figure 1.1.

1.3.1 Notation

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

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

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

All of the base kernels we'll consider in this chapter 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.

Each kernel has a number of parameters (sometimes called *hyperparameters*) which specify the precise shape of the covariance function. To remove clutter, we'll usually refer to a kernel without specifying these parameters.

1.4 Modeling Additive Functions

Additivity is a very useful modeling assumption in a wide variety of contexts. It often aids in building interpretable models, as well as enabling extrapolation in high-dimensional settings. Fortunately, this assumption is easy to encode in GP models.

A sum of two orthogonal one-dimensional kernels   Mutliplicative kernels

$k_1(x_1, x_1')$ + $k_2(x_2, x_2')$ = $k_1(x_1, x_1') + k_2(x_2, x_2')$ $k_1(x_1, x_1')k_2(x_2, x_2')$

$f_1 \sim \mathcal{GP}(0, k_1)$ + $f_2 \sim \mathcal{GP}(0, k_2)$ = $f_1(x_1) + f_2(x_2)$ $f(x_1, x_2)$
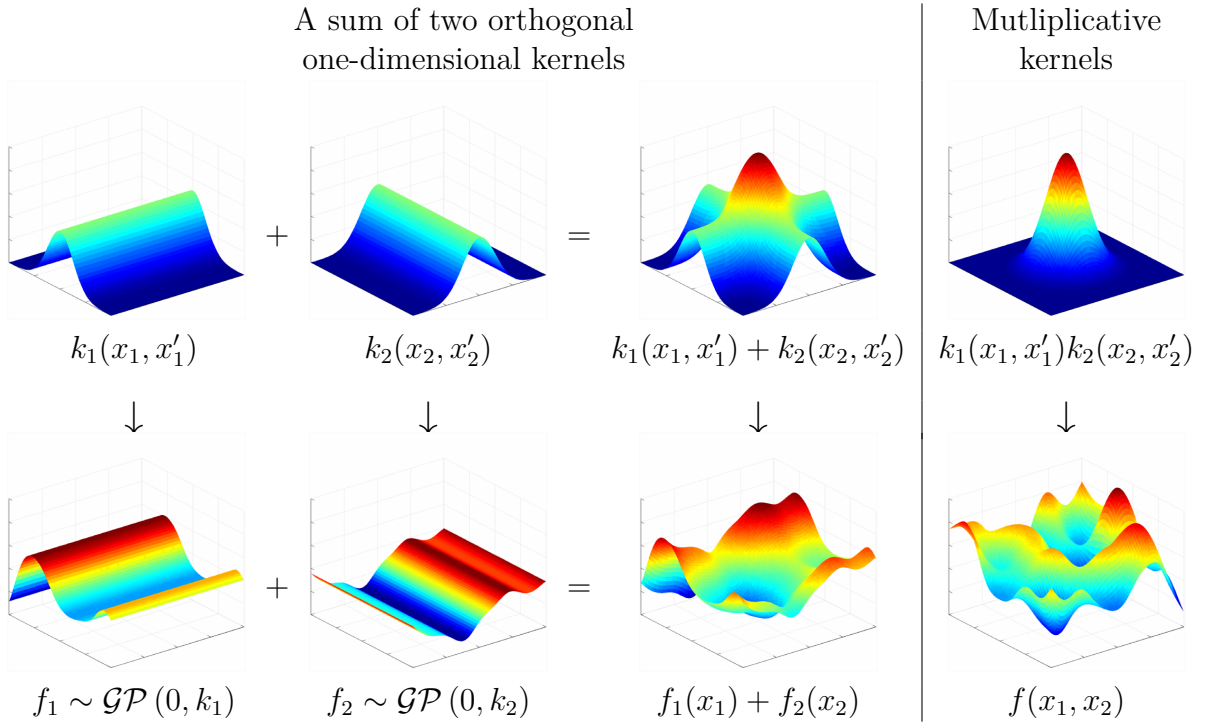
Fig. 1.2 Top left: An additive kernel is a sum of kernels. Bottom left: A draw from an additive kernel corresponds to a sum of draws from independent GP priors with the corresponding kernels. Top right: A product kernel. Bottom right: A GP prior with a product of kernels does not correspond to a product of draws from GPs.

By summing kernels, we can model the data as a sum of independent functions, each possibly representing a different type of structure. 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 we can model the sum of those functions through 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 any type, and we can sum up any number of kernels this way.

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

$$f(x_1, x_2) \sim \mathcal{GP}(\mathbf{0}, k_1(x_1, x_1') + k_2(x_2, x_2')) \tag{1.7}$$

Then this is equivalent to the model

$$f_1(x_1) \sim \mathcal{GP}(\mathbf{0}, k_1(x_1, x_1')) \tag{1.8}$$

$$f_2(x_2) \sim \mathcal{GP}(\mathbf{0}, k_2(x_2, x_2')) \tag{1.9}$$

$$f(x_1, x_2) = f_1(x_1) + f_2(x_2) \tag{1.10}$$

Figure 1.2 illustrates this decomposition. 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

We now give an example of how additive kernels give rise to interpetable models. We model measurements of the compressive strength of concrete, as a function of the concentration of 7 ingredients, plus the age of the concrete (Yeh, 1998).

We build an additive model

$$\begin{aligned} f(\mathbf{x}) = {} & f_1(\text{cement}) + f_2(\text{slag}) + f_3(\text{fly ash}) + f_4(\text{water}) \\ & + f_5(\text{plasticizer}) + f_6(\text{coarse}) + f_7(\text{fine}) + f_8(\text{age}) + \epsilon \end{aligned} \tag{1.11}$$

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

Figure 1.3 shows the marginal posterior distribution of each of the 8 components in Equation (1.11). 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).
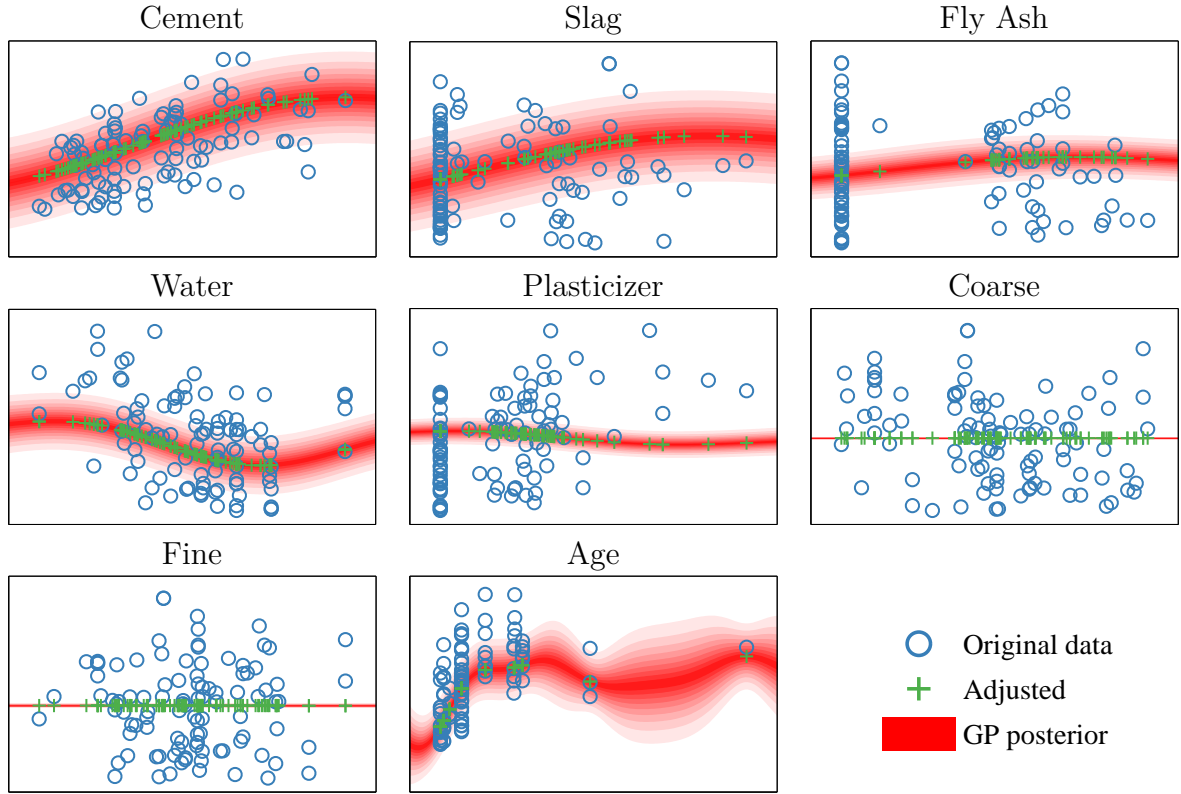
Fig. 1.3 By considering only one-dimensional terms of the additive kernel, we recover a form of Generalized Additive Model, and can plot the corresponding 1-dimensional functions. Blue points indicate the original data, green points are data after the contribution from all other terms has been subtracted. The vertical axis is the same for all plots.

1.4.3   Marginal Covariance of Components

How to compute the posterior distributions shown in Figure 1.3? Here we derive the posterior marginal variance and covariance of each of the additive components of a GP.

First, we'll write down the joint prior over the sum of two functions. 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

$$
\begin{bmatrix}
\boldsymbol{f}_1(\mathbf{x}) \\
\boldsymbol{f}_1(\mathbf{x}^\star) \\
\boldsymbol{f}_2(\mathbf{x}) \\
\boldsymbol{f}_2(\mathbf{x}^\star) \\
\boldsymbol{f}_1(\mathbf{x}) + \boldsymbol{f}_2(\mathbf{x}) \\
\boldsymbol{f}_1(\mathbf{x}^\star) + \boldsymbol{f}_2(\mathbf{x}^\star)
\end{bmatrix}
\sim \mathcal{N}\left(
\begin{bmatrix}
\mu_1 \\
\mu_1^\star \\
\mu_2 \\
\mu_2^\star \\
\mu_1 + \mu_2 \\
\mu_1^\star + \mu_2^\star
\end{bmatrix},
\begin{bmatrix}
\mathbf{K}_1 & \mathbf{K}_1^\star & 0 & 0 & \mathbf{K}_1 & \mathbf{K}_1^\star \\
\mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} & 0 & 0 & \mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} \\
0 & 0 & \mathbf{K}_2 & \mathbf{K}_2^\star & \mathbf{K}_2 & \mathbf{K}_2^\star \\
0 & 0 & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} \\
\mathbf{K}_1 & \mathbf{K}_1^\star & \mathbf{K}_2 & \mathbf{K}_2^\star & \mathbf{K}_1 + \mathbf{K}_2 & \mathbf{K}_1^\star + \mathbf{K}_2^\star \\
\mathbf{K}_1^\star & \mathbf{K}_1^{\star\star} & \mathbf{K}_2^\star & \mathbf{K}_2^{\star\star} & \mathbf{K}_1^\star + \mathbf{K}_2^\star & \mathbf{K}_1^{\star\star} + \mathbf{K}_2^{\star\star}
\end{bmatrix}
\right)
\tag{1.12}
$$

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

$$\mathbf{K}_1 = k_1(\mathbf{X}, \mathbf{X}) \tag{1.13}$$

$$\mathbf{K}_1^\star = k_1(\mathbf{X}^\star, \mathbf{X}) \tag{1.14}$$

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

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

$$
\begin{aligned}
\boldsymbol{f}_1^\star | \boldsymbol{f} \sim \mathcal{N}\Big(& \mu_1^\star + {\mathbf{K}_1^\star}^{\mathsf{T}} (\mathbf{K}_1 + \mathbf{K}_2)^{-1} \left(\boldsymbol{f} - \mu_1 - \mu_2\right) \\
& \mathbf{K}_1^{\star\star} - {\mathbf{K}_1^\star}^{\mathsf{T}} (\mathbf{K}_1 + \mathbf{K}_2)^{-1} \mathbf{K}_1^\star \Big)
\end{aligned}
\tag{1.16}
$$

These formulae express the posterior model uncertainty about 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 Components

One of the advantages of using a generative, model-based approach is that we can examine any aspect of the model we wish to. For instance, we can compute the posterior covariance between any two components, conditioned on their sum:

$$\mathrm{cov}\left[\boldsymbol{f}_1^\star, \boldsymbol{f}_2^\star | \boldsymbol{f}\right] = -{\mathbf{K}_1^\star}^{\mathsf{T}} (\mathbf{K}_1 + \mathbf{K}_2)^{-1} \mathbf{K}_2^\star \tag{1.17}$$
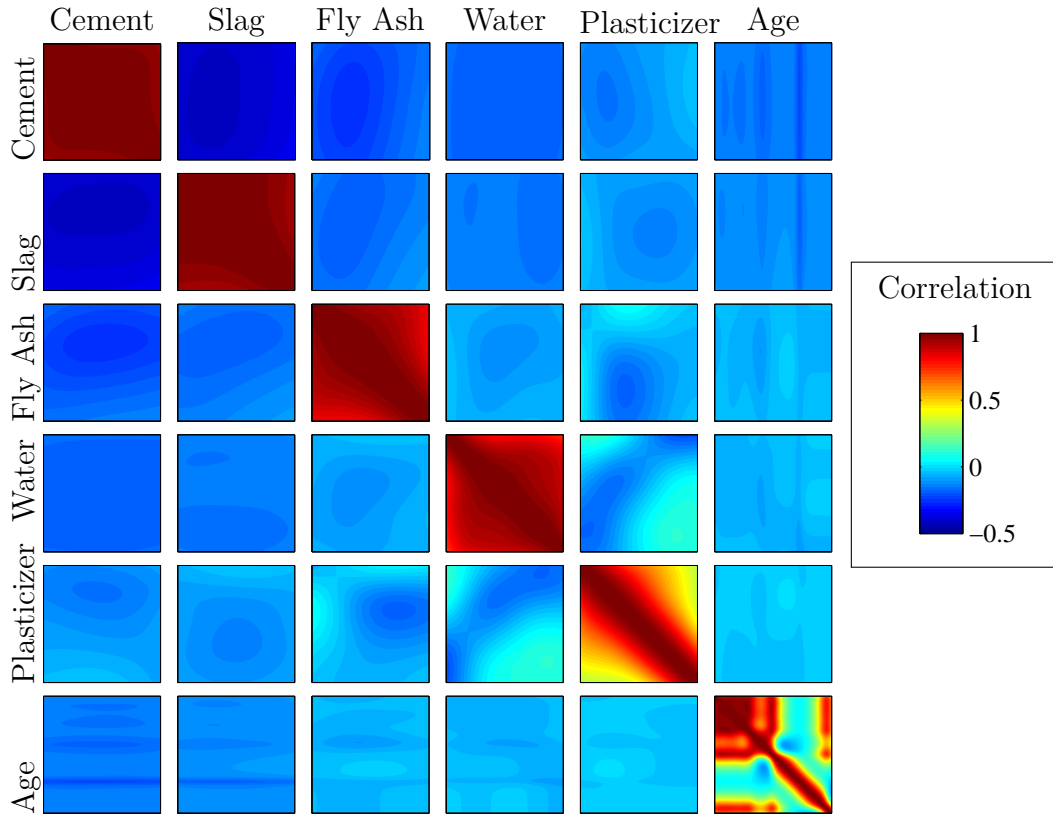
Fig. 1.4 Visualizing posterior correlations between the components explaining the concrete dataset. Each plot shows the additive model's posterior correlation between two components, plotted over the domain of the data ±5%. Red indicates high correlation, teal indicates no correlation, and blue indicates 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.

For example, Figure 1.4 plots 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

Finding additive structure is useful, because it allows us to make predictions far from the training data. Figure 1.5 compares the extrapolations made by additive versus

True function:
$f(x_1, x_2) = \sin(x_1) + \sin(x_2)$

GP mean using
product of SE kernels:
$k(x_1, x_1') \times k(x_2, x_2')$

GP mean using
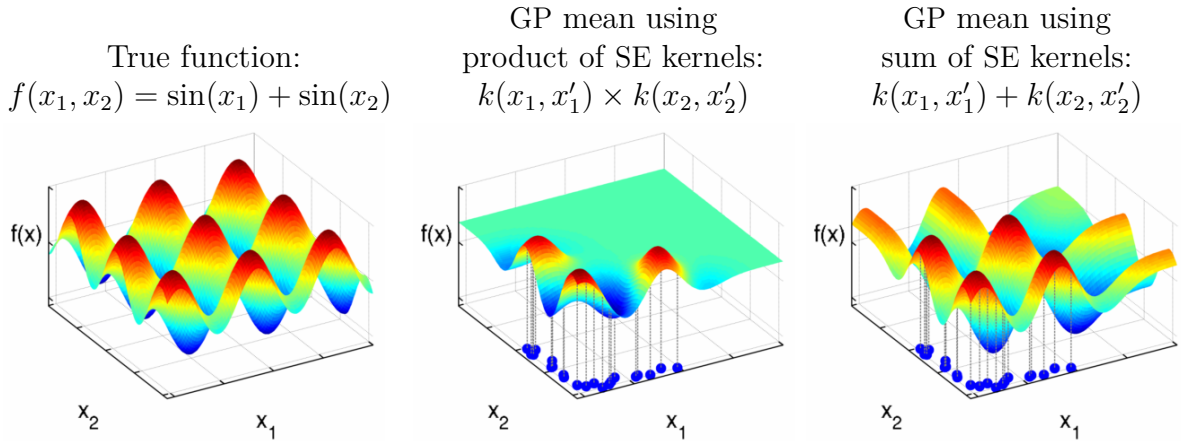sum of SE kernels:
$k(x_1, x_1') + k(x_2, x_2')$

Fig. 1.5 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.

non-additive GP models, conditioned on data from a sum of two axis-aligned sine functions, evaluated in a small, L-shaped area. In this example, the additive model is able to correctly predict the value of the function at unseen combinations of inputs. The product-kernel model is more flexible, and so remains uncertain about the function value at unseen combinations of inputs.

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 × SE corresponds to locally periodic structure, as shown in row 1 of figure 1.2.

Products of SE kernels, having a different lengthscale parameter for each dimension,

Lin × Lin	SE × Per	Lin × SE	Lin × Per

x with $(x' = 1)$	$x - x'$	x with $(x' = 1)$	x with $(x' = 1)$
↓	↓	↓	↓

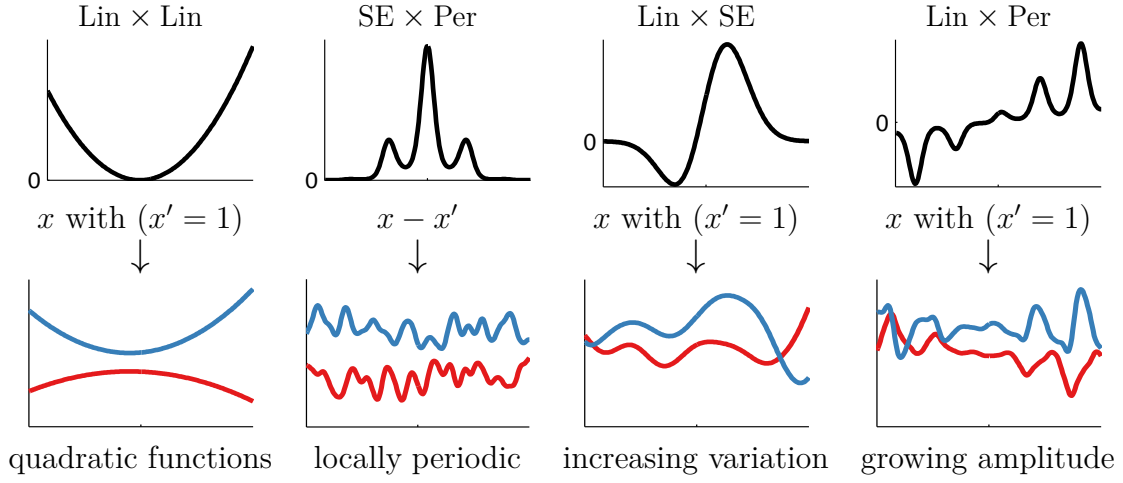| quadratic functions	locally periodic	increasing variation	growing amplitude

Fig. 1.6 Examples of one-dimensional structures expressible by multiplying kernels. Plots have same meaning as in figure 1.1.

are equivalent to the SE-ARD kernel:

$$\text{SE-ARD}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{\ell_d^2}\right) \tag{1.18}$$

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. [TODO: flesh this section out]

1.6 Changepoints

An example of aChangepoints can be defined through addition and multiplication with sigmoidal functions:

$$\text{CP}(k_1, k_2) = k_1 \times \boldsymbol{\sigma} + k_2 \times \bar{\boldsymbol{\sigma}} \tag{1.19}$$

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.

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,

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

(1.20)

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':

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

(1.21)

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 **??**.

1.7.1   Relation to Linear Regression

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

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

(1.22)

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

1.7.2 Feature-space view of Combining Kernels

Many architectures for learning complex functions, such as convolutional networks Le-Cun 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

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

$$k_b(\mathbf{x}, \mathbf{x}') = \mathbf{b}(\mathbf{x})^\mathsf{T}\mathbf{b}(\mathbf{x}') \tag{1.24}$$

and we consider their addition, then

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

$$= \begin{bmatrix} \mathbf{a}(\mathbf{x}) \\ \mathbf{b}(\mathbf{x}) \end{bmatrix}^\mathsf{T} \begin{bmatrix} \mathbf{a}(\mathbf{x}') \\ \mathbf{b}(\mathbf{x}') \end{bmatrix} \tag{1.26}$$

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:

$$k_a(\mathbf{x}, \mathbf{x}') \times k_b(\mathbf{x}, \mathbf{x}') = \left[\mathbf{a}(\mathbf{x})^\mathsf{T}\mathbf{a}(\mathbf{x}')\right] \times \left[\mathbf{b}(\mathbf{x})^\mathsf{T}\mathbf{b}(\mathbf{x}')\right] \tag{1.27}$$

$$= \sum_i a_i(\mathbf{x})a_i(\mathbf{x}') \times \sum_j b_j(\mathbf{x})b_j(\mathbf{x}') \tag{1.28}$$

$$= \sum_i \sum_j a_i(\mathbf{x})a_i(\mathbf{x}')b_j(\mathbf{x})b_j(\mathbf{x}') \tag{1.29}$$

$$= \sum_{i,j} \left[a_i(\mathbf{x})b_j(\mathbf{x})\right]\left[a_i(\mathbf{x}')b_j(\mathbf{x}')\right] \tag{1.30}$$

$$= \text{vec}\left(\mathbf{a}(\mathbf{x}) \otimes \mathbf{b}(\mathbf{x}')\right)^\mathsf{T} \text{vec}\left(\mathbf{a}(\mathbf{x}) \otimes \mathbf{b}(\mathbf{x}')\right) \tag{1.31}$$

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 w.r.t. 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 argumentwise invariant:

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}, \forall g, g' \in G, \quad k(g(\mathbf{x}), g(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}') \tag{1.32}$$

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

$$g_1(f(x, y)) = f(x, y) \qquad \text{(identity)} \tag{1.33}$$
$$g_2(f(x, y)) = f(y, x) \qquad \text{(swap)} \tag{1.34}$$

It might not be clear how to come up with a kernel obeying these symmetries. Fortunately, for finite groups, there are a few simple ways to transform any kernel into one which is argumentwise invariant to actions under the group:

1. **Sum over the Orbit** Ginsbourger et al. (2012) and Kondor (2008) suggest a

Additive method	Projection method	Product method

$\mathrm{SE}(x, y, x', y')$
$+\mathrm{SE}(x, y, y', x')$ | $\mathrm{SE}(\min(x, y), \max(x, y),$
$\min(x', y'), \max(x', y'))$ | $\mathrm{SE}(x, y, x', y')$
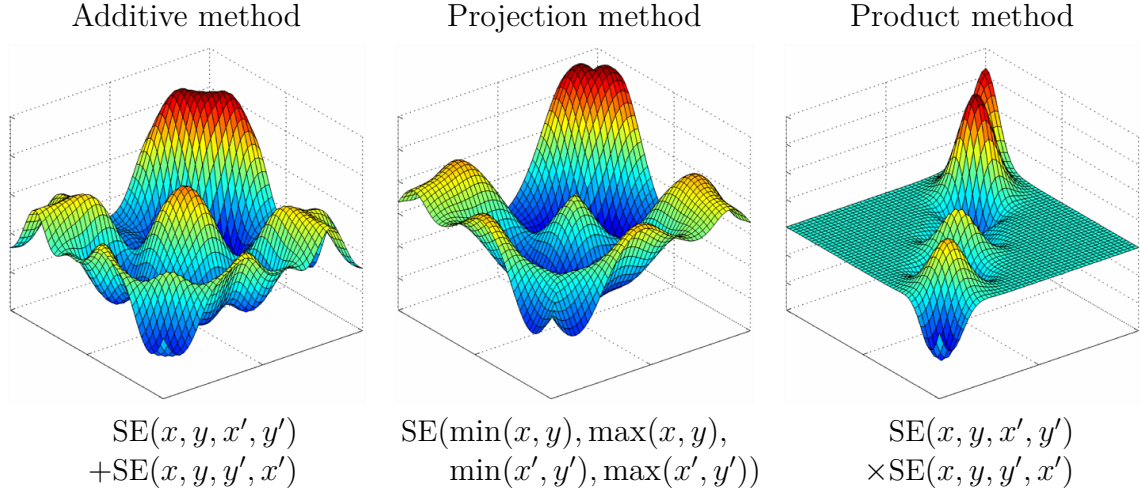$\times\mathrm{SE}(x, y, y', x')$

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

double sum over the orbits of $\mathbf{x}$ and $\mathbf{x}'$ w.r.t. G:

$$k_{\mathrm{sum}}(\mathbf{x}, \mathbf{x}') = \sum_{g, \in G} \sum_{g' \in G} k(g(\mathbf{x}), g'(\mathbf{x}')) \tag{1.35}$$

For our example group G_{switch}, this operation results in:

$$k_{\mathrm{switch}}(\mathbf{x}, \mathbf{x}') = \sum_{g, \in G_{\mathrm{switch}}} \sum_{g' \in G_{\mathrm{switch}}} k(g(\mathbf{x}), g'(\mathbf{x}')) \tag{1.36}$$

$$= k(x, y, x', y') + k(x, y, y', x') + k(y, x, x', y') + k(y, x, y', x') \tag{1.37}$$

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

2. **Projection to a Fundamental Domain** Ginsbourger et al. (2013) also explore the possibility of projecting each datapoint into a fundamental domain of the group:

$$k_{\mathrm{rep}}(\mathbf{x}, \mathbf{x}') = k(A_G(\mathbf{x}), A_G(\mathbf{x}')) \tag{1.38}$$

For our example group G_{switch}, a fundamental domain is $\{x, y : x < y\}$, which can

be mapped to using $A_{G_{\text{switch}}}(x, y) = \left[\min(x, y), \max(x, y)\right]$. Constructing a kernel using this method introduces a non-differentiable "seam" along $x = y$.

The projection method also works for infinite groups, as we shall see below.

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

$$k_{\text{sum}}(\mathbf{x}, \mathbf{x}') = \prod_{g, \in G} \prod_{g' \in G} k(g(\mathbf{x}), g'(\mathbf{x}')) \tag{1.39}$$

This method is not recommended, since it will often produce GP priors with zero variance in some regions of the space, as in Figure 1.7c.

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.

1.8.1   Example: Periodicity

Periodicity in a one-dimensional function corresponds to the invariance

$$f(x) = f(x + \tau) \tag{1.40}$$

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

$$\text{Per}(x, x') = \text{SE}(\left[\sin(x), \cos(x)\right], \left[\sin(x'), \cos(x')\right]) \tag{1.41}$$

achieves the invariance in Equation (1.40). Some simple algebra reduces this kernel to the form shown in Table 1.1.

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

1.8.2   Example: Reflective Symmetry Along an Axis

We can enforce symmetry about zero

$$f(x) = f(-x) \tag{1.42}$$

using the sum over orbits method, by the transform

$$k_{\text{symm axis}}(x, x') = k(x, x') + k(x, -x') + k(-x, x') + k(-x, -x') \tag{1.43}$$

1.8.3   Example: Translation Invariance in Images

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

   Note that this sort of translational 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 every pixel. We are interested in creating priors on functions that are invariant to shifting a signal along its pixels:

$$f\left(\begin{array}{c}\blacksquare\end{array} \right) = f\left(\begin{array}{c}\blacksquare\end{array} \right) \tag{1.44}$$

Translational invariance in this setting is equivalent to invariance two swapping sets of dimensions in the input space.

   This invariance can be achieved in one dimension by using the transformation

$$k_{\text{invariant}}\left(\mathbf{x}, \mathbf{x}'\right) = \sum_{i=1}^{D} \sum_{j=1}^{D} k(\text{shift}(\mathbf{x}, i), \text{shift}(\mathbf{x}, j)) \tag{1.45}$$

where

$$\text{shift}(\mathbf{x}, i) = \left[x_{\text{mod}(i+1,D)}, x_{\text{mod}(i+2,D)}, \ldots, x_{\text{mod}(D+i,D)} \right] \tag{1.46}$$

The extension to two dimensions is straightforward, but notationally cumbersome. We are simply defining the covariance between two images to be the sum of all covariances
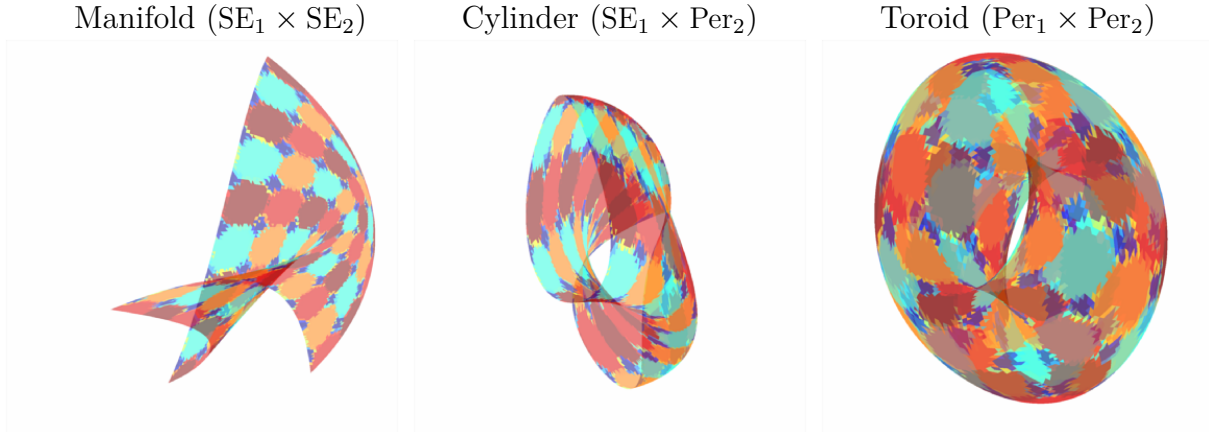
| Manifold ($SE_1 \times SE_2$) | Cylinder ($SE_1 \times Per_2$) | Toroid ($Per_1 \times Per_2$) |

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).

between all translations of the two images.

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

1.9 Generating Topological Manifolds

We now give a geometric illustration of the symmetries encoded by GPs with different sorts 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 do so, first create a mesh in 2D. Next, draw 3 independent functions from a GP prior. Then, map the 2D mesh through those functions to get 3D coordinates of each point on the mesh.

This construction is similar in spirit to the GP latent variable model (GP-LVM) (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.

1.9.1 Surfaces, Cylinders and Torii

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.
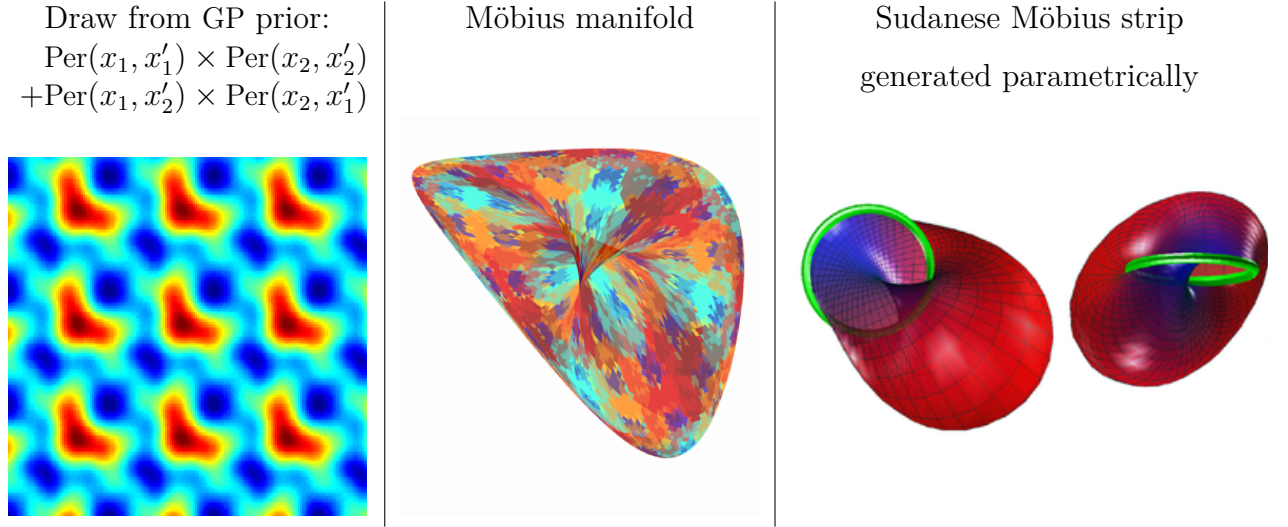
Draw from GP prior: $\text{Per}(x_1, x_1') \times \text{Per}(x_2, x_2')$ $+\text{Per}(x_1, x_2') \times \text{Per}(x_2, x_1')$	Möbius manifold	Sudanese Möbius strip generated parametrically

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).

1.9.2   Möbius Strips

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

$$f(x, y) = f(x, y + \tau_y) \tag{1.47}$$

$$f(x, y) = f(x + \tau_x, y) \tag{1.48}$$

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

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 $x \in \{1, 2, 3, 4, 5\}$, one-of-k(3) = $[0, 0, 1, 0, 0]$.

$$k_{\text{categorical}}(x, x') = \text{SE-ARD}(\text{one-of-k}(x), \text{one-of-k}(x')) \tag{1.50}$$

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:

$$\text{SE-full}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\mathbf{x}^{\mathsf{T}}\mathbf{L}\mathbf{x}'\right) \tag{1.51}$$

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 11)

Wikimedia Commons. Stereographic projection of a Sudanese Möbius band, 2005. URL http://commons.wikimedia.org/wiki/File:MobiusSnail2B.png. (page 18)

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 13 and 14)

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. (page 13)

T.J. Hastie and R.J. Tibshirani. *Generalized additive models*. Chapman & Hall/CRC, 1990. (page 9)

Imre Risi Kondor. *Group theoretical methods in machine learning*. PhD thesis, Columbia University, 2008. (pages 13 and 16)

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 17)

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 12)

Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361, 1995. (page 16)

D. Lerner and D. Asimov. The Sudanese Möbius band. In *SIGGRAPH Electronic Theatre*, 1984. (page 18)

David JC MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998. (page 15)

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 11)

Ha Quang Minh, Partha Niyogi, and Yuan Yao. Mercer's theorem, feature maps, and smoothing. In *Learning theory*, pages 154–168. Springer, 2006. (page 11)

Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, University of Toronto, 1995. (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 12)

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)