C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

Symbols and Notation

Matrices are capitalized and vectors are in bold type. We do not generally distinguish between probabilities and probability densities. A subscript asterisk, such as in X_*, indicates reference to a *test set* quantity. A superscript asterisk denotes complex conjugate.

Symbol	Meaning
$\backslash$	left matrix divide: $A\backslash\mathbf{b}$ is the vector $\mathbf{x}$ which solves $A\mathbf{x}=\mathbf{b}$
$\triangleq$	an equality which acts as a definition
$\overset{c}{=}$	equality up to an additive constant
$\lvert K\rvert$	determinant of K matrix
$\lvert\mathbf{y}\rvert$	Euclidean length of vector $\mathbf{y}$, i.e. $\left(\sum_i y_i^2\right)^{1/2}$
$\langle f, g\rangle_{\mathcal{H}}$	RKHS inner product
$\lVert f\rVert_{\mathcal{H}}$	RKHS norm
$\mathbf{y}^\top$	the transpose of vector $\mathbf{y}$
$\propto$	proportional to; e.g. $p(x\lvert y)\propto f(x,y)$ means that $p(x\lvert y)$ is equal to $f(x,y)$ times a factor which is independent of x
$\sim$	distributed according to; example: $x\sim\mathcal{N}(\mu,\sigma^2)$
∇ or $\nabla_{\mathbf{f}}$	partial derivatives (w.r.t. $\mathbf{f}$)
$\nabla\nabla$	the (Hessian) matrix of second derivatives
$\mathbf{0}$ or $\mathbf{0}_n$	vector of all 0's (of length n)
$\mathbf{1}$ or $\mathbf{1}_n$	vector of all 1's (of length n)
C	number of classes in a classification problem
cholesky(A)	Cholesky decomposition: L is a lower triangular matrix such that $LL^\top=A$
cov$(\mathbf{f}_*)$	Gaussian process posterior covariance
D	dimension of input space $\mathcal{X}$
$\mathcal{D}$	data set: $\mathcal{D}=\{(\mathbf{x}_i,y_i)\lvert i=1,\ldots,n\}$
diag$(\mathbf{w})$	(vector argument) a diagonal matrix containing the elements of vector $\mathbf{w}$
diag(W)	(matrix argument) a vector containing the diagonal elements of matrix W
δ_{pq}	Kronecker delta, $\delta_{pq}=1$ iff $p=q$ and 0 otherwise
$\mathbb{E}$ or $\mathbb{E}_{q(x)}[z(x)]$	expectation; expectation of $z(x)$ when $x\sim q(x)$
$f(\mathbf{x})$ or $\mathbf{f}$	Gaussian process (or vector of) latent function values, $\mathbf{f}=(f(\mathbf{x}_1),\ldots,f(\mathbf{x}_n))^\top$
$\mathbf{f}_*$	Gaussian process (posterior) prediction (random variable)
$\bar{\mathbf{f}}_*$	Gaussian process posterior mean
$\mathcal{GP}$	Gaussian process: $f\sim\mathcal{GP}\big(m(\mathbf{x}),k(\mathbf{x},\mathbf{x}')\big)$, the function f is distributed as a Gaussian process with mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x},\mathbf{x}')$
$h(\mathbf{x})$ or $\mathbf{h}(\mathbf{x})$	*either* fixed basis function (or set of basis functions) *or* weight function
H or $H(X)$	set of basis functions evaluated at all training points
I or I_n	the identity matrix (of size n)
$J_\nu(z)$	Bessel function of the first kind
$k(\mathbf{x},\mathbf{x}')$	covariance (or kernel) function evaluated at $\mathbf{x}$ and $\mathbf{x}'$
K or $K(X,X)$	$n\times n$ covariance (or Gram) matrix
K_*	$n\times n_*$ matrix $K(X,X_*)$, the covariance between training and test cases
$\mathbf{k}(\mathbf{x}_*)$ or $\mathbf{k}_*$	vector, short for $K(X,\mathbf{x}_*)$, when there is only a single test case
K_f or K	covariance matrix for the (noise free) $\mathbf{f}$ values

Symbol	Meaning
K_y	covariance matrix for the (noisy) $\mathbf{y}$ values; for independent homoscedastic noise, $K_y = K_f + \sigma_n^2 I$
$K_\nu(z)$	modified Bessel function
$\mathcal{L}(a, b)$	loss function, the loss of predicting b, when a is true; note argument order
$\log(z)$	natural logarithm (base e)
$\log_2(z)$	logarithm to the base 2
ℓ or ℓ_d	characteristic length-scale (for input dimension d)
$\lambda(z)$	logistic function, $\lambda(z) = 1/\big(1 + \exp(-z)\big)$
$m(\mathbf{x})$	the mean function of a Gaussian process
μ	a measure (see section A.7)
$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ or $\mathcal{N}(\mathbf{x}\|\boldsymbol{\mu}, \Sigma)$	(the variable $\mathbf{x}$ has a) Gaussian (Normal) distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ
$\mathcal{N}(\mathbf{x})$	short for unit Gaussian $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, I)$
n and n_*	number of training (and test) cases
N	dimension of feature space
N_H	number of hidden units in a neural network
$\mathbb{N}$	the natural numbers, the positive integers
$\mathcal{O}(\cdot)$	big Oh; for functions f and g on $\mathbb{N}$, we write $f(n) = \mathcal{O}(g(n))$ if the ratio $f(n)/g(n)$ remains bounded as $n \to \infty$
O	*either* matrix of all zeros *or* differential operator
$y\|x$ and $p(y\|x)$	conditional random variable y given x and its probability (density)
$\mathbb{P}_N$	the regular n-polygon
$\boldsymbol{\phi}(\mathbf{x}_i)$ or $\Phi(X)$	feature map of input $\mathbf{x}_i$ (or input set X)
$\Phi(z)$	cumulative unit Gaussian: $\Phi(z) = (2\pi)^{-1/2} \int_{-\infty}^{z} \exp(-t^2/2) dt$
$\pi(\mathbf{x})$	the sigmoid of the latent value: $\pi(\mathbf{x}) = \sigma(f(\mathbf{x}))$ (stochastic if $f(\mathbf{x})$ is stochastic)
$\hat{\pi}(\mathbf{x}_*)$	MAP prediction: π evaluated at $\bar{f}(\mathbf{x}_*)$.
$\bar{\pi}(\mathbf{x}_*)$	mean prediction: expected value of $\pi(\mathbf{x}_*)$. Note, in general that $\hat{\pi}(\mathbf{x}_*) \neq \bar{\pi}(\mathbf{x}_*)$
$\mathbb{R}$	the real numbers
$R_{\mathcal{L}}(f)$ or $R_{\mathcal{L}}(c)$	the risk or expected loss for f, or classifier c (averaged w.r.t. inputs and outputs)
$\tilde{R}_{\mathcal{L}}(l\|\mathbf{x}_*)$	expected loss for predicting l, averaged w.r.t. the model's pred. distr. at $\mathbf{x}_*$
$\mathcal{R}_c$	decision region for class c
$S(\mathbf{s})$	power spectrum
$\sigma(z)$	any sigmoid function, e.g. logistic $\lambda(z)$, cumulative Gaussian $\Phi(z)$, etc.
σ_f^2	variance of the (noise free) signal
σ_n^2	noise variance
$\boldsymbol{\theta}$	vector of hyperparameters (parameters of the covariance function)
$\mathrm{tr}(A)$	trace of (square) matrix A
$\mathbb{T}_l$	the circle with circumference l
$\mathbb{V}$ or $\mathbb{V}_{q(x)}[z(x)]$	variance; variance of $z(x)$ when $x \sim q(x)$
$\mathcal{X}$	input space and also the index set for the stochastic process
X	$D \times n$ matrix of the training inputs $\{\mathbf{x}_i\}_{i=1}^n$: the design matrix
X_*	matrix of test inputs
$\mathbf{x}_i$	the ith training input
x_{di}	the dth coordinate of the ith training input $\mathbf{x}_i$
$\mathbb{Z}$	the integers $\ldots, -2, -1, 0, 1, 2, \ldots$

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

Chapter 1

Introduction

In this book we will be concerned with supervised learning, which is the problem of learning input-output mappings from empirical data (the training dataset). Depending on the characteristics of the output, this problem is known as either regression, for continuous outputs, or classification, when outputs are discrete.

digit classification

A well known example is the classification of images of handwritten digits. The training set consists of small digitized images, together with a classification from $0, \ldots, 9$, normally provided by a human. The goal is to learn a mapping from image to classification label, which can then be used on new, unseen images. Supervised learning is an attractive way to attempt to tackle this problem, since it is not easy to specify accurately the characteristics of, say, the handwritten digit 4.

robotic control

An example of a regression problem can be found in robotics, where we wish to learn the inverse dynamics of a robot arm. Here the task is to map from the state of the arm (given by the positions, velocities and accelerations of the joints) to the corresponding torques on the joints. Such a model can then be used to compute the torques needed to move the arm along a given trajectory. Another example would be in a chemical plant, where we might wish to predict the yield as a function of process parameters such as temperature, pressure, amount of catalyst etc.

the dataset

In general we denote the input as $\mathbf{x}$, and the output (or target) as y. The input is usually represented as a vector $\mathbf{x}$ as there are in general many input variables—in the handwritten digit recognition example one may have a 256-dimensional input obtained from a raster scan of a 16×16 image, and in the robot arm example there are three input measurements for each joint in the arm. The target y may either be continuous (as in the regression case) or discrete (as in the classification case). We have a dataset $\mathcal{D}$ of n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$.

training is inductive

Given this training data we wish to make predictions for new inputs $\mathbf{x}_*$ that we have not seen in the training set. Thus it is clear that the problem at hand is *inductive*; we need to move from the finite training data $\mathcal{D}$ to a

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

function f that makes predictions for all possible input values. To do this we must make assumptions about the characteristics of the underlying function, as otherwise any function which is consistent with the training data would be equally valid. A wide variety of methods have been proposed to deal with the supervised learning problem; here we describe two common approaches. The first is to restrict the class of functions that we consider, for example by only considering linear functions of the input. The second approach is (speaking rather loosely) to give a prior probability to every possible function, where higher probabilities are given to functions that we consider to be more likely, for example because they are smoother than other functions.[1] The first approach has an obvious problem in that we have to decide upon the richness of the class of functions considered; if we are using a model based on a certain class of functions (e.g. linear functions) and the target function is not well modelled by this class, then the predictions will be poor. One may be tempted to increase the flexibility of the class of functions, but this runs into the danger of overfitting, where we can obtain a good fit to the training data, but perform badly when making test predictions.

two approaches

The second approach appears to have a serious problem, in that surely there are an uncountably infinite set of possible functions, and how are we going to compute with this set in finite time? This is where the Gaussian process comes to our rescue. A Gaussian *process* is a generalization of the Gaussian probability *distribution*. Whereas a probability distribution describes random variables which are scalars or vectors (for multivariate distributions), a stochastic *process* governs the properties of functions. Leaving mathematical sophistication aside, one can loosely think of a function as a very long vector, each entry in the vector specifying the function value $f(x)$ at a particular input x. It turns out, that although this idea is a little naïve, it is surprisingly close what we need. Indeed, the question of how we deal computationally with these infinite dimensional objects has the most pleasant resolution imaginable: if you ask only for the properties of the function at a finite number of points, then inference in the Gaussian process will give you the same answer if you ignore the infinitely many other points, as if you would have taken them all into account! And these answers are consistent with answers to any other finite queries you may have. One of the main attractions of the Gaussian process framework is precisely that it unites a sophisticated and consistent view with computational tractability.

Gaussian process

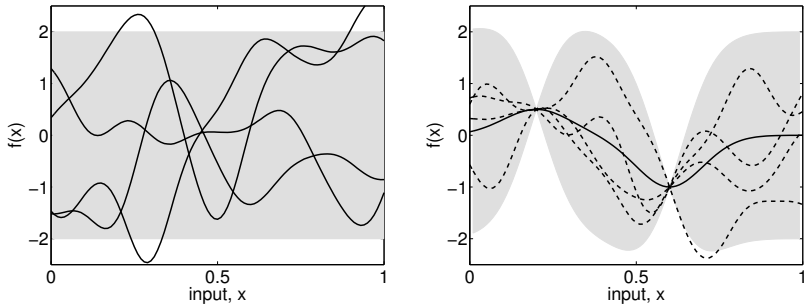
consistency

tractability

It should come as no surprise that these ideas have been around for some time, although they are perhaps not as well known as they might be. Indeed, many models that are commonly employed in both machine learning and statistics are in fact special cases of, or restricted kinds of Gaussian processes. In this volume, we aim to give a systematic and unified treatment of the area, showing connections to related models.

[1] These two approaches may be regarded as imposing a *restriction* bias and a *preference* bias respectively; see e.g. Mitchell [1997].

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml



(a), prior (b), posterior

Figure 1.1: Panel (a) shows four samples drawn from the prior distribution. Panel (b) shows the situation after two datapoints have been observed. The mean prediction is shown as the solid line and four samples from the posterior are shown as dashed lines. In both plots the shaded region denotes twice the standard deviation at each input value x.

1.1 A Pictorial Introduction to Bayesian Modelling

In this section we give graphical illustrations of how the second (Bayesian) method works on some simple regression and classification examples.

regression

We first consider a simple 1-d *regression* problem, mapping from an input x to an output $f(x)$. In Figure 1.1(a) we show a number of sample functions drawn at random from the *prior* distribution over functions specified by a particular Gaussian process which favours smooth functions. This prior is taken to represent our prior beliefs over the kinds of functions we expect to observe, before seeing any data. In the absence of knowledge to the contrary we have assumed that the average value over the sample functions at each x is zero. Although the specific random functions drawn in Figure 1.1(a) do not have a mean of zero, the mean of $f(x)$ values for any fixed x would become zero, independent of x as we kept on drawing more functions. At any value of x we can also characterize the variability of the sample functions by computing the variance at that point. The shaded region denotes twice the pointwise standard deviation; in this case we used a Gaussian process which specifies that the prior variance does not depend on x.

random functions

mean function

pointwise variance

Suppose that we are then given a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)\}$ consisting of two observations, and we wish now to only consider functions that pass though these two data points exactly. (It is also possible to give higher preference to functions that merely pass "close" to the datapoints.) This situation is illustrated in Figure 1.1(b). The dashed lines show sample functions which are consistent with $\mathcal{D}$, and the solid line depicts the mean value of such functions. Notice how the uncertainty is reduced close to the observations. The combination of the prior and the data leads to the *posterior* distribution over functions.

functions that agree with observations

posterior over functions

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

If more datapoints were added one would see the mean function adjust itself to pass through these points, and that the posterior uncertainty would reduce close to the observations. Notice, that since the Gaussian process is not a parametric model, we do not have to worry about whether it is possible for the model to fit the data (as would be the case if e.g. you tried a linear model on strongly non-linear data). Even when a lot of observations have been added, there may still be some flexibility left in the functions. One way to imagine the reduction of flexibility in the distribution of functions as the data arrives is to draw many random functions from the prior, and reject the ones which do not agree with the observations. While this is a perfectly valid way to do inference, it is impractical for most purposes—the exact analytical computations required to quantify these properties will be detailed in the next chapter.

non-parametric

inference

prior specification

The specification of the prior is important, because it fixes the properties of the functions considered for inference. Above we briefly touched on the mean and pointwise variance of the functions. However, other characteristics can also be specified and manipulated. Note that the functions in Figure 1.1(a) are smooth and stationary (informally, stationarity means that the functions look similar at all x locations). These are properties which are induced by the *covariance function* of the Gaussian process; many other covariance functions are possible. Suppose, that for a particular application, we think that the functions in Figure 1.1(a) vary too rapidly (i.e. that their characteristic length-scale is too short). Slower variation is achieved by simply adjusting parameters of the covariance function. The problem of *learning* in Gaussian processes is exactly the problem of finding suitable properties for the covariance function. Note, that this gives us a model of the data, and characteristics (such a smoothness, characteristic length-scale, etc.) which we can *interpret*.

covariance function

modelling and interpreting

classification

We now turn to the *classification* case, and consider the binary (or two-class) classification problem. An example of this is classifying objects detected in astronomical sky surveys into stars or galaxies. Our data has the label $+1$ for stars and -1 for galaxies, and our task will be to predict $\pi(\mathbf{x})$, the probability that an example with input vector $\mathbf{x}$ is a star, using as inputs some features that describe each object. Obviously $\pi(\mathbf{x})$ should lie in the interval $[0, 1]$. A Gaussian process prior over functions does not restrict the output to lie in this interval, as can be seen from Figure 1.1(a). The approach that we shall adopt is to squash the prior function f pointwise through a response function which restricts the output to lie in $[0, 1]$. A common choice for this function is the logistic function $\lambda(z) = (1 + \exp(-z))^{-1}$, illustrated in Figure 1.2(b). Thus the prior over f induces a prior over probabilistic classifications π.

squashing function

This set up is illustrated in Figure 1.2 for a 2-d input space. In panel (a) we see a sample drawn from the prior over functions f which is squashed through the logistic function (panel (b)). A dataset is shown in panel (c), where the white and black circles denote classes $+1$ and -1 respectively. As in the regression case the effect of the data is to downweight in the posterior those functions that are incompatible with the data. A contour plot of the posterior mean for $\pi(\mathbf{x})$ is shown in panel (d). In this example we have chosen a short characteristic length-scale for the process so that it can vary fairly rapidly; in

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

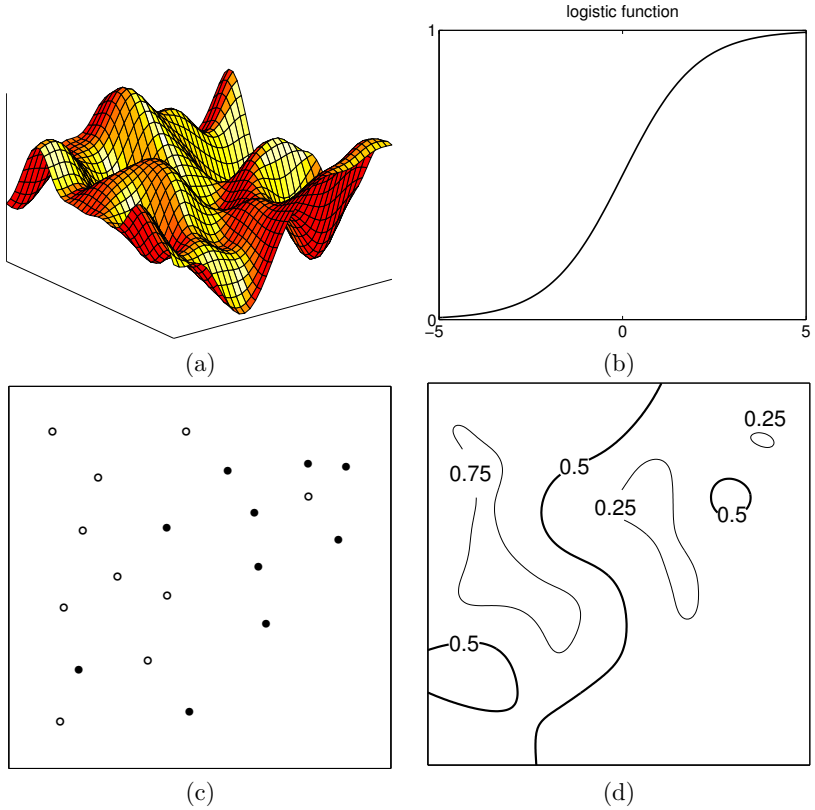


Figure 1.2: Panel (a) shows a sample from prior distribution on f in a 2-d input space. Panel (b) is a plot of the logistic function $\lambda(z)$. Panel (c) shows the location of the data points, where the open circles denote the class label $+1$, and closed circles denote the class label -1. Panel (d) shows a contour plot of the mean predictive probability as a function of $\mathbf{x}$; the decision boundaries between the two classes are shown by the thicker lines.

this case notice that all of the training points are correctly classified, including the two "outliers" in the NE and SW corners. By choosing a different length-scale we can change this behaviour, as illustrated in section 3.7.1.

1.2 Roadmap

The book has a natural split into two parts, with the chapters up to and including chapter 5 covering core material, and the remaining chapters covering the connections to other methods, fast approximations, and more specialized properties. Some sections are marked by an asterisk. These sections may be omitted on a first reading, and are not pre-requisites for later (un-starred) material.

regression

Chapter 2 contains the definition of Gaussian processes, in particular for the use in regression. It also discusses the computations needed to make predictions for regression. Under the assumption of Gaussian observation noise the computations needed to make predictions are tractable and are dominated by the inversion of a $n \times n$ matrix. In a short experimental section, the Gaussian process model is applied to a robotics task.

classification

Chapter 3 considers the classification problem for both binary and multi-class cases. The use of a non-linear response function means that exact computation of the predictions is no longer possible analytically. We discuss a number of approximation schemes, include detailed algorithms for their implementation and discuss some experimental comparisons.

covariance functions

As discussed above, the key factor that controls the properties of a Gaussian process is the covariance function. Much of the work on machine learning so far, has used a very limited set of covariance functions, possibly limiting the power of the resulting models. In chapter 4 we discuss a number of valid covariance functions and their properties and provide some guidelines on how to combine covariance functions into new ones, tailored to specific needs.

learning

Many covariance functions have adjustable parameters, such as the characteristic length-scale and variance illustrated in Figure 1.1. Chapter 5 describes how such parameters can be inferred or learned from the data, based on either Bayesian methods (using the marginal likelihood) or methods of cross-validation. Explicit algorithms are provided for some schemes, and some simple practical examples are demonstrated.

connections

Gaussian process predictors are an example of a class of methods known as kernel machines; they are distinguished by the probabilistic viewpoint taken. In chapter 6 we discuss other kernel machines such as support vector machines (SVMs), splines, least-squares classifiers and relevance vector machines (RVMs), and their relationships to Gaussian process prediction.

theory

In chapter 7 we discuss a number of more theoretical issues relating to Gaussian process methods including asymptotic analysis, average-case learning curves and the PAC-Bayesian framework.

fast approximations

One issue with Gaussian process prediction methods is that their basic complexity is $\mathcal{O}(n^3)$, due to the inversion of a $n \times n$ matrix. For large datasets this is prohibitive (in both time and space) and so a number of approximation methods have been developed, as described in chapter 8.

The main focus of the book is on the core supervised learning problems of regression and classification. In chapter 9 we discuss some rather less standard settings that GPs have been used in, and complete the main part of the book with some conclusions.

Appendix A gives some mathematical background, while Appendix B deals specifically with Gaussian Markov processes. Appendix C gives details of how to access the data and programs that were used to make the some of the figures and run the experiments described in the book.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

Chapter 2

Regression

Supervised learning can be divided into regression and classification problems. Whereas the outputs for classification are discrete class labels, regression is concerned with the prediction of continuous quantities. For example, in a financial application, one may attempt to predict the price of a commodity as a function of interest rates, currency exchange rates, availability and demand. In this chapter we describe Gaussian process methods for regression problems; classification problems are discussed in chapter 3.

two equivalent views

There are several ways to interpret Gaussian process (GP) regression models. One can think of a Gaussian process as defining a distribution over functions, and inference taking place directly in the space of functions, the *function-space view*. Although this view is appealing it may initially be difficult to grasp, so we start our exposition in section 2.1 with the equivalent *weight-space view* which may be more familiar and accessible to many, and continue in section 2.2 with the function-space view. Gaussian processes often have characteristics that can be changed by setting certain parameters and in section 2.3 we discuss how the properties change as these parameters are varied. The predictions from a GP model take the form of a full predictive distribution; in section 2.4 we discuss how to combine a loss function with the predictive distributions using decision theory to make point predictions in an optimal way. A practical comparative example involving the learning of the inverse dynamics of a robot arm is presented in section 2.5. We give some theoretical analysis of Gaussian process regression in section 2.6, and discuss how to incorporate explicit basis functions into the models in section 2.7. As much of the material in this chapter can be considered fairly standard, we postpone most references to the historical overview in section 2.8.

2.1 Weight-space View

The simple linear regression model where the output is a linear combination of the inputs has been studied and used extensively. Its main virtues are simplic-

ity of implementation and interpretability. Its main drawback is that it only allows a limited flexibility; if the relationship between input and output cannot reasonably be approximated by a linear function, the model will give poor predictions.

In this section we first discuss the Bayesian treatment of the linear model. We then make a simple enhancement to this class of models by projecting the inputs into a high-dimensional *feature space* and applying the linear model there. We show that in some feature spaces one can apply the "kernel trick" to carry out computations implicitly in the high dimensional space; this last step leads to computational savings when the dimensionality of the feature space is large compared to the number of data points.

training set

We have a training set $\mathcal{D}$ of n observations, $\mathcal{D} = \{(\mathbf{x}_i, y_i) \mid i = 1, \ldots, n\}$, where $\mathbf{x}$ denotes an input vector (covariates) of dimension D and y denotes a scalar output or target (dependent variable); the column vector inputs for all n cases are aggregated in the $D \times n$ *design matrix*[1] X, and the targets are collected in the vector $\mathbf{y}$, so we can write $\mathcal{D} = (X, \mathbf{y})$. In the regression setting the targets are real values. We are interested in making inferences about the relationship between inputs and targets, i.e. the conditional distribution of the targets given the inputs (but we are not interested in modelling the input distribution itself).

design matrix

2.1.1 The Standard Linear Model

We will review the Bayesian analysis of the standard linear regression model with Gaussian noise

$$f(\mathbf{x}) = \mathbf{x}^\top \mathbf{w}, \qquad y = f(\mathbf{x}) + \varepsilon, \tag{2.1}$$

where $\mathbf{x}$ is the input vector, $\mathbf{w}$ is a vector of weights (parameters) of the linear model, f is the function value and y is the observed target value. Often a bias weight or offset is included, but as this can be implemented by augmenting the input vector $\mathbf{x}$ with an additional element whose value is always one, we do not explicitly include it in our notation. We have assumed that the observed values y differ from the function values $f(\mathbf{x})$ by additive noise, and we will further assume that this noise follows an independent, identically distributed Gaussian distribution with zero mean and variance σ_n^2

bias, offset

$$\varepsilon \sim \mathcal{N}(0, \sigma_n^2). \tag{2.2}$$

likelihood

This noise assumption together with the model directly gives rise to the *likelihood*, the probability density of the observations given the parameters, which is

[1] In statistics texts the design matrix is usually taken to be the transpose of our definition, but our choice is deliberate and has the advantage that a data point is a standard (column) vector.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

factored over cases in the training set (because of the independence assumption) to give

$$\begin{aligned} p(\mathbf{y}|X,\mathbf{w}) &= \prod_{i=1}^{n} p(y_i|\mathbf{x}_i,\mathbf{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp\big(-\frac{(y_i-\mathbf{x}_i^\top\mathbf{w})^2}{2\sigma_n^2}\big) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp\big(-\frac{1}{2\sigma_n^2}|\mathbf{y}-X^\top\mathbf{w}|^2\big) = \mathcal{N}(X^\top\mathbf{w},\sigma_n^2 I), \end{aligned} \tag{2.3}$$

where $|\mathbf{z}|$ denotes the Euclidean length of vector $\mathbf{z}$. In the Bayesian formalism we need to specify a *prior* over the parameters, expressing our beliefs about the parameters before we look at the observations. We put a zero mean Gaussian prior with covariance matrix Σ_p on the weights

prior

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p). \tag{2.4}$$

The rôle and properties of this prior will be discussed in section 2.2; for now we will continue the derivation with the prior as specified.

Inference in the Bayesian linear model is based on the posterior distribution over the weights, computed by Bayes' rule, (see eq. (A.3))[2]

posterior

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \qquad p(\mathbf{w}|\mathbf{y},X) = \frac{p(\mathbf{y}|X,\mathbf{w})p(\mathbf{w})}{p(\mathbf{y}|X)}, \tag{2.5}$$

where the normalizing constant, also known as the marginal likelihood (see page 19), is independent of the weights and given by

marginal likelihood

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|X,\mathbf{w})p(\mathbf{w})\, d\mathbf{w}. \tag{2.6}$$

The posterior in eq. (2.5) combines the likelihood and the prior, and captures everything we know about the parameters. Writing only the terms from the likelihood and prior which depend on the weights, and "completing the square" we obtain

$$\begin{aligned} p(\mathbf{w}|X,\mathbf{y}) &\propto \exp\big(-\frac{1}{2\sigma_n^2}(\mathbf{y}-X^\top\mathbf{w})^\top(\mathbf{y}-X^\top\mathbf{w})\big)\exp\big(-\frac{1}{2}\mathbf{w}^\top\Sigma_p^{-1}\mathbf{w}\big) \\ &\propto \exp\big(-\frac{1}{2}(\mathbf{w}-\bar{\mathbf{w}})^\top\big(\frac{1}{\sigma_n^2}XX^\top+\Sigma_p^{-1}\big)(\mathbf{w}-\bar{\mathbf{w}})\big), \end{aligned} \tag{2.7}$$

where $\bar{\mathbf{w}} = \sigma_n^{-2}(\sigma_n^{-2}XX^\top+\Sigma_p^{-1})^{-1}X\mathbf{y}$, and we recognize the form of the posterior distribution as Gaussian with mean $\bar{\mathbf{w}}$ and covariance matrix A^{-1}

$$p(\mathbf{w}|X,\mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{w}}=\frac{1}{\sigma_n^2}A^{-1}X\mathbf{y},\ A^{-1}), \tag{2.8}$$

where $A = \sigma_n^{-2}XX^\top+\Sigma_p^{-1}$. Notice that for this model (and indeed for any Gaussian posterior) the *mean* of the posterior distribution $p(\mathbf{w}|\mathbf{y},X)$ is also its mode, which is also called the *maximum a posteriori* (MAP) estimate of

MAP estimate

[2] Often Bayes' rule is stated as $p(a|b) = p(b|a)p(a)/p(b)$; here we use it in a form where we additionally condition everywhere on the inputs X (but neglect this extra conditioning for the prior which is independent of the inputs).

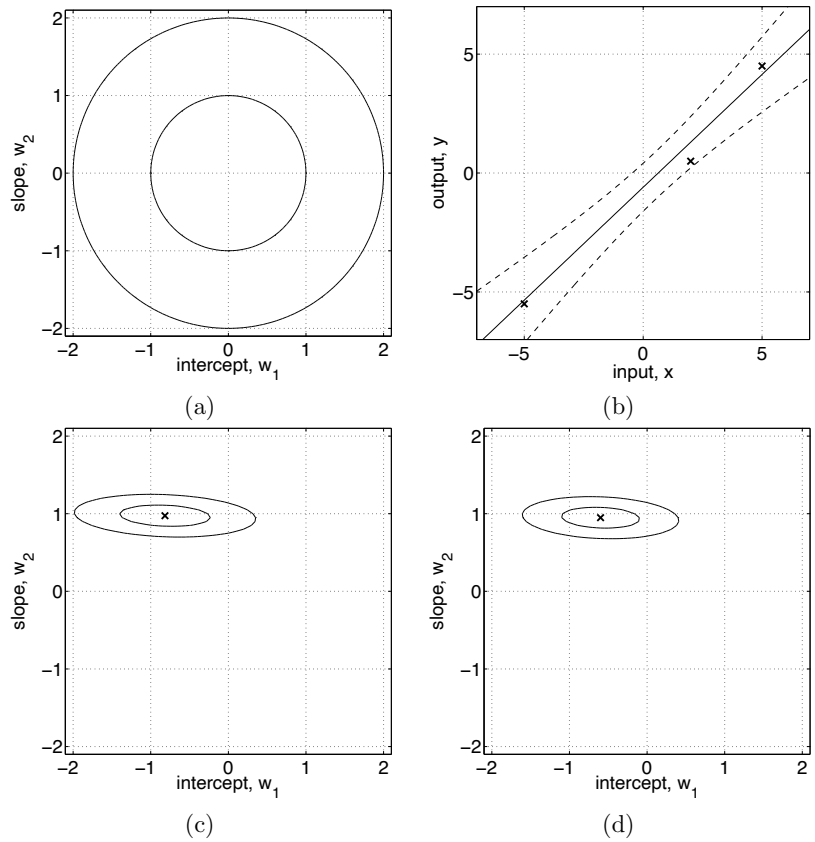


Figure 2.1: Example of Bayesian linear model $f(x) = w_1 + w_2 x$ with intercept w_1 and slope parameter w_2. Panel (a) shows the contours of the prior distribution $p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, I)$, eq. (2.4). Panel (b) shows three training points marked by crosses. Panel (c) shows contours of the likelihood $p(\mathbf{y}|X, \mathbf{w})$ eq. (2.3), assuming a noise level of $\sigma_n = 1$; note that the slope is much more "well determined" than the intercept. Panel (d) shows the posterior, $p(\mathbf{w}|X, \mathbf{y})$ eq. (2.7); comparing the maximum of the posterior to the likelihood, we see that the intercept has been shrunk towards zero whereas the more 'well determined' slope is almost unchanged. All contour plots give the 1 and 2 standard deviation equi-probability contours. Superimposed on the data in panel (b) are the predictive mean plus/minus two standard deviations of the (noise-free) predictive distribution $p(f_*|\mathbf{x}_*, X, \mathbf{y})$, eq. (2.9).

$\mathbf{w}$. In a non-Bayesian setting the negative log prior is sometimes thought of as a *penalty* term, and the MAP point is known as the penalized maximum likelihood estimate of the weights, and this may cause some confusion between the two approaches. Note, however, that in the Bayesian setting the MAP estimate plays no special rôle.[3] The penalized maximum likelihood procedure

[3] In this case, due to symmetries in the model and posterior, it happens that the mean of the predictive distribution is the same as the prediction at the mean of the posterior. However, this is not the case in general.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

is known in this case as *ridge* regression [Hoerl and Kennard, 1970] because of the effect of the quadratic penalty term $\frac{1}{2}\mathbf{w}^\top \Sigma_p^{-1}\mathbf{w}$ from the log prior. ridge regression

To make predictions for a test case we average over all possible parameter values, weighted by their posterior probability. This is in contrast to non-Bayesian schemes, where a single parameter is typically chosen by some criterion. Thus the predictive distribution for $f_* \triangleq f(\mathbf{x}_*)$ at $\mathbf{x}_*$ is given by averaging the output of all possible linear models w.r.t. the Gaussian posterior predictive distribution

$$\begin{aligned} p(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int p(f_*|\mathbf{x}_*, \mathbf{w}) p(\mathbf{w}|X, \mathbf{y})\, d\mathbf{w} \\ &= \mathcal{N}\big(\frac{1}{\sigma_n^2}\mathbf{x}_*^\top A^{-1} X\mathbf{y},\ \mathbf{x}_*^\top A^{-1}\mathbf{x}_*\big). \end{aligned} \tag{2.9}$$

The predictive distribution is again Gaussian, with a mean given by the posterior mean of the weights from eq. (2.8) multiplied by the test input, as one would expect from symmetry considerations. The predictive variance is a quadratic form of the test input with the posterior covariance matrix, showing that the predictive uncertainties grow with the magnitude of the test input, as one would expect for a linear model.

An example of Bayesian linear regression is given in Figure 2.1. Here we have chosen a 1-d input space so that the weight-space is two-dimensional and can be easily visualized. Contours of the Gaussian prior are shown in panel (a). The data are depicted as crosses in panel (b). This gives rise to the likelihood shown in panel (c) and the posterior distribution in panel (d). The predictive distribution and its error bars are also marked in panel (b).

2.1.2 Projections of Inputs into Feature Space

In the previous section we reviewed the Bayesian linear model which suffers from limited expressiveness. A very simple idea to overcome this problem is to first project the inputs into some high dimensional space using a set of basis functions and then apply the linear model in this space instead of directly on the inputs themselves. For example, a scalar input x could be projected into the space of powers of x: $\boldsymbol{\phi}(x) = (1, x, x^2, x^3, \ldots)^\top$ to implement polynomial regression. As long as the projections are fixed functions (i.e. independent of the parameters $\mathbf{w}$) the model is still linear in the parameters, and therefore analytically tractable.[4] This idea is also used in classification, where a dataset which is not linearly separable in the original data space may become linearly separable in a high dimensional feature space, see section 3.3. Application of this idea begs the question of how to choose the basis functions? As we shall demonstrate (in chapter 5), the Gaussian process formalism allows us to answer this question. For now, we assume that the basis functions are given. feature space; polynomial regression; linear in the parameters

Specifically, we introduce the function $\boldsymbol{\phi}(\mathbf{x})$ which maps a D-dimensional input vector $\mathbf{x}$ into an N dimensional feature space. Further let the matrix

[4] Models with adaptive basis functions, such as e.g. multilayer perceptrons, may at first seem like a useful extension, but they are much harder to treat, except in the limit of an infinite number of hidden units, see section 4.2.3.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

$\Phi(X)$ be the aggregation of columns $\boldsymbol{\phi}(\mathbf{x})$ for all cases in the training set. Now the model is

$$f(\mathbf{x}) \;=\; \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}, \tag{2.10}$$

where the vector of parameters now has length N. The analysis for this model is analogous to the standard linear model, except that everywhere $\Phi(X)$ is substituted for X. Thus the predictive distribution becomes

explicit feature space formulation

$$f_*|\mathbf{x}_*, X, \mathbf{y} \;\sim\; \mathcal{N}\big(\frac{1}{\sigma_n^2}\boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1}\Phi\mathbf{y},\; \boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1}\boldsymbol{\phi}(\mathbf{x}_*)\big) \tag{2.11}$$

with $\Phi = \Phi(X)$ and $A = \sigma_n^{-2}\Phi\Phi^\top + \Sigma_p^{-1}$. To make predictions using this equation we need to invert the A matrix of size $N \times N$ which may not be convenient if N, the dimension of the feature space, is large. However, we can rewrite the equation in the following way

alternative formulation

$$\begin{aligned} f_*|\mathbf{x}_*, X, \mathbf{y} \;\sim\; \mathcal{N}\big(&\boldsymbol{\phi}_*^\top\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\mathbf{y}, \\ &\boldsymbol{\phi}_*^\top\Sigma_p\boldsymbol{\phi}_* - \boldsymbol{\phi}_*^\top\Sigma_p\Phi(K+\sigma_n^2 I)^{-1}\Phi^\top\Sigma_p\boldsymbol{\phi}_*\big), \end{aligned} \tag{2.12}$$

where we have used the shorthand $\boldsymbol{\phi}(\mathbf{x}_*) = \boldsymbol{\phi}_*$ and defined $K = \Phi^\top\Sigma_p\Phi$. To show this for the mean, first note that using the definitions of A and K we have $\sigma_n^{-2}\Phi(K+\sigma_n^2 I) = \sigma_n^{-2}\Phi(\Phi^\top\Sigma_p\Phi + \sigma_n^2 I) = A\Sigma_p\Phi$. Now multiplying through by A^{-1} from left and $(K+\sigma_n^2 I)^{-1}$ from the right gives $\sigma_n^{-2}A^{-1}\Phi = \Sigma_p\Phi(K+\sigma_n^2 I)^{-1}$, showing the equivalence of the mean expressions in eq. (2.11) and eq. (2.12). For the variance we use the matrix inversion lemma, eq. (A.9), setting $Z^{-1} = \Sigma_p$, $W^{-1} = \sigma_n^2 I$ and $V = U = \Phi$ therein. In eq. (2.12) we need to invert matrices of size $n \times n$ which is more convenient when $n < N$. Geometrically, note that n datapoints can span at most n dimensions in the feature space.

computational load

Notice that in eq. (2.12) the feature space always enters in the form of $\Phi^\top\Sigma_p\Phi$, $\boldsymbol{\phi}_*^\top\Sigma_p\Phi$, or $\boldsymbol{\phi}_*^\top\Sigma_p\boldsymbol{\phi}_*$; thus the entries of these matrices are invariably of the form $\boldsymbol{\phi}(\mathbf{x})^\top\Sigma_p\boldsymbol{\phi}(\mathbf{x}')$ where $\mathbf{x}$ and $\mathbf{x}'$ are in either the training or the test sets. Let us define $k(\mathbf{x},\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top\Sigma_p\boldsymbol{\phi}(\mathbf{x}')$. For reasons that will become clear later we call $k(\cdot,\cdot)$ a *covariance function* or *kernel*. Notice that $\boldsymbol{\phi}(\mathbf{x})^\top\Sigma_p\boldsymbol{\phi}(\mathbf{x}')$ is an inner product (with respect to Σ_p). As Σ_p is positive definite we can define $\Sigma_p^{1/2}$ so that $(\Sigma_p^{1/2})^2 = \Sigma_p$; for example if the SVD (singular value decomposition) of $\Sigma_p = UDU^\top$, where D is diagonal, then one form for $\Sigma_p^{1/2}$ is $UD^{1/2}U^\top$. Then defining $\boldsymbol{\psi}(\mathbf{x}) = \Sigma_p^{1/2}\boldsymbol{\phi}(\mathbf{x})$ we obtain a simple dot product representation $k(\mathbf{x},\mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x}')$.

kernel

If an algorithm is defined solely in terms of inner products in input space then it can be lifted into feature space by replacing occurrences of those inner products by $k(\mathbf{x},\mathbf{x}')$; this is sometimes called the *kernel trick*. This technique is particularly valuable in situations where it is more convenient to compute the kernel than the feature vectors themselves. As we will see in the coming sections, this often leads to considering the kernel as the object of primary interest, and its corresponding feature space as having secondary practical importance.

kernel trick

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

2.2 Function-space View

An alternative and equivalent way of reaching identical results to the previous section is possible by considering inference directly in function space. We use a Gaussian process (GP) to describe a distribution over functions. Formally:

Gaussian process

Definition 2.1 *A* Gaussian process *is a collection of random variables, any finite number of which have a joint Gaussian distribution.* □

covariance and mean function

A Gaussian process is completely specified by its mean function and covariance function. We define mean function $m(\mathbf{x})$ and the covariance function $k(\mathbf{x}, \mathbf{x}')$ of a real process $f(\mathbf{x})$ as

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{2.13}$$

and will write the Gaussian process as

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big). \tag{2.14}$$

Usually, for notational simplicity we will take the mean function to be zero, although this need not be done, see section 2.7.

index set ≡ input domain

In our case the random variables represent the value of the function $f(\mathbf{x})$ at location $\mathbf{x}$. Often, Gaussian processes are defined over time, i.e. where the index set of the random variables is time. This is not (normally) the case in our use of GPs; here the index set $\mathcal{X}$ is the set of possible inputs, which could be more general, e.g. $\mathbb{R}^D$. For notational convenience we use the (arbitrary) enumeration of the cases in the training set to identify the random variables such that $f_i \triangleq f(\mathbf{x}_i)$ is the random variable corresponding to the case $(\mathbf{x}_i, y_i)$ as would be expected.

marginalization property

A Gaussian process is defined as a collection of random variables. Thus, the definition automatically implies a *consistency* requirement, which is also sometimes known as the marginalization property. This property simply means that if the GP e.g. specifies $(y_1, y_2) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then it must also specify $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ where Σ_{11} is the relevant submatrix of Σ, see eq. (A.6). In other words, examination of a larger set of variables does not change the distribution of the smaller set. Notice that the consistency requirement is automatically fulfilled if the covariance function specifies entries of the covariance matrix.[5] The definition does not exclude Gaussian processes with finite index sets (which would be simply Gaussian *distributions*), but these are not particularly interesting for our purposes.

finite index set

[5] Note, however, that if you instead specified e.g. a function for the entries of the *inverse* covariance matrix, then the marginalization property would no longer be fulfilled, and one could not think of this as a consistent collection of random variables—this would not qualify as a Gaussian process.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

Bayesian linear model is a Gaussian process

A simple example of a Gaussian process can be obtained from our Bayesian linear regression model $f(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{w}$ with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p)$. We have for the mean and covariance

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}[\mathbf{w}] = 0, \\ \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}[\mathbf{w}\mathbf{w}^\top]\boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \Sigma_p \boldsymbol{\phi}(\mathbf{x}'). \end{aligned} \tag{2.15}$$

Thus $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian with zero mean and covariance given by $\boldsymbol{\phi}(\mathbf{x})^\top \Sigma_p \boldsymbol{\phi}(\mathbf{x}')$. Indeed, the function values $f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)$ corresponding to any number of input points n are jointly Gaussian, although if $N < n$ then this Gaussian is singular (as the joint covariance matrix will be of rank N).

In this chapter our running example of a covariance function will be the *squared exponential*[6] (SE) covariance function; other covariance functions are discussed in chapter 4. The covariance function specifies the covariance between pairs of random variables

$$\operatorname{cov}\big(f(\mathbf{x}_p), f(\mathbf{x}_q)\big) = k(\mathbf{x}_p, \mathbf{x}_q) = \exp\big(-\tfrac{1}{2}|\mathbf{x}_p - \mathbf{x}_q|^2\big). \tag{2.16}$$

Note, that the covariance between the *outputs* is written as a function of the *inputs*. For this particular covariance function, we see that the covariance is almost unity between variables whose corresponding inputs are very close, and decreases as their distance in the input space increases.

basis functions

It can be shown (see section 4.3.1) that the squared exponential covariance function corresponds to a Bayesian linear regression model with an infinite number of basis functions. Indeed for every positive definite covariance function $k(\cdot, \cdot)$, there exists a (possibly infinite) expansion in terms of basis functions (see Mercer's theorem in section 4.3). We can also obtain the SE covariance function from the linear combination of an infinite number of Gaussian-shaped basis functions, see eq. (4.13) and eq. (4.30).

The specification of the covariance function implies a distribution over functions. To see this, we can draw samples from the distribution of functions evaluated at any number of points; in detail, we choose a number of input points,[7] X_* and write out the corresponding covariance matrix using eq. (2.16) elementwise. Then we generate a random Gaussian vector with this covariance matrix

$$\mathbf{f}_* \sim \mathcal{N}\big(\mathbf{0}, K(X_*, X_*)\big), \tag{2.17}$$

and plot the generated values as a function of the inputs. Figure 2.2(a) shows three such samples. The generation of multivariate Gaussian samples is described in section A.2.

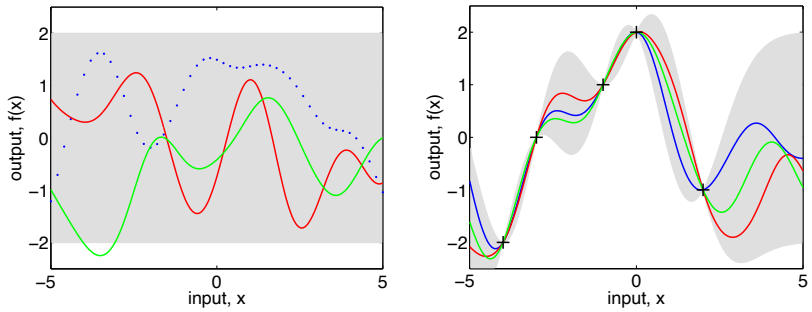
smoothness

characteristic length-scale

In the example in Figure 2.2 the input values were equidistant, but this need not be the case. Notice that "informally" the functions look smooth. In fact the squared exponential covariance function is infinitely differentiable, leading to the process being infinitely mean-square differentiable (see section 4.1). We also see that the functions seem to have a characteristic length-scale,

[6] Sometimes this covariance function is called the Radial Basis Function (RBF) or Gaussian; here we prefer squared exponential.

[7] Technically, these input points play the rôle of *test inputs* and therefore carry a subscript asterisk; this will become clearer later when both training and test points are involved.



(a), prior (b), posterior

Figure 2.2: Panel (a) shows three functions drawn at random from a GP prior; the dots indicate values of y actually generated; the two other functions have (less correctly) been drawn as lines by joining a large number of evaluated points. Panel (b) shows three random functions drawn from the posterior, i.e. the prior conditioned on the five noise free observations indicated. In both plots the shaded area represents the pointwise mean plus and minus two times the standard deviation for each input value (corresponding to the 95% confidence region), for the prior and posterior respectively.

which informally can be thought of as roughly the distance you have to move in input space before the function value can change significantly, see section 4.2.1. For eq. (2.16) the characteristic length-scale is around one unit. By replacing $|\mathbf{x}_p - \mathbf{x}_q|$ by $|\mathbf{x}_p - \mathbf{x}_q|/\ell$ in eq. (2.16) for some positive constant ℓ we could change the characteristic length-scale of the process. Also, the overall variance of the random function can be controlled by a positive pre-factor before the exp in eq. (2.16). We will discuss more about how such factors affect the predictions in section 2.3, and say more about how to set such scale parameters in chapter 5.

magnitude

Prediction with Noise-free Observations

We are usually not primarily interested in drawing random functions from the prior, but want to incorporate the knowledge that the training data provides about the function. Initially, we will consider the simple special case where the observations are noise free, that is we know $\{(\mathbf{x}_i, f_i)|i = 1, \ldots, n\}$. The joint distribution of the training outputs, $\mathbf{f}$, and the test outputs $\mathbf{f}_*$ according to the prior is

joint prior

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{2.18}$$

If there are n training points and n_* test points then $K(X, X_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated at all pairs of training and test points, and similarly for the other entries $K(X,X)$, $K(X_*,X_*)$ and $K(X_*,X)$. To get the posterior distribution over functions we need to restrict this joint prior distribution to contain only those functions which agree with the observed data points. Graphically in Figure 2.2 you may think of generating functions from the prior, and rejecting the ones that disagree with the observations, al-

graphical rejection

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

though this strategy would not be computationally very efficient. Fortunately, in probabilistic terms this operation is extremely simple, corresponding to *conditioning* the joint Gaussian prior distribution on the observations (see section A.2 for further details) to give

noise-free predictive distribution

$$\begin{aligned}\mathbf{f}_*|X_*, X, \mathbf{f} \;\sim\; \mathcal{N}\big(&K(X_*, X)K(X, X)^{-1}\mathbf{f}, \\ &K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\big).\end{aligned} \tag{2.19}$$

Function values $\mathbf{f}_*$ (corresponding to test inputs X_*) can be sampled from the joint posterior distribution by evaluating the mean and covariance matrix from eq. (2.19) and generating samples according to the method described in section A.2.

Figure 2.2(b) shows the results of these computations given the five datapoints marked with + symbols. Notice that it is trivial to extend these computations to multidimensional inputs – one simply needs to change the evaluation of the covariance function in accordance with eq. (2.16), although the resulting functions may be harder to display graphically.

Prediction using Noisy Observations

It is typical for more realistic modelling situations that we do not have access to function values themselves, but only noisy versions thereof $y = f(\mathbf{x}) + \varepsilon$.[8] Assuming additive independent identically distributed Gaussian noise ε with variance σ_n^2, the prior on the noisy observations becomes

$$\operatorname{cov}(y_p, y_q) \;=\; k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2\delta_{pq} \quad \text{or} \quad \operatorname{cov}(\mathbf{y}) \;=\; K(X, X) + \sigma_n^2 I, \tag{2.20}$$

where δ_{pq} is a Kronecker delta which is one iff $p = q$ and zero otherwise. It follows from the independence[9] assumption about the noise, that a diagonal matrix[10] is added, in comparison to the noise free case, eq. (2.16). Introducing the noise term in eq. (2.18) we can write the joint distribution of the observed target values and the function values at the test locations under the prior as

$$\begin{bmatrix}\mathbf{y} \\ \mathbf{f}_*\end{bmatrix} \;\sim\; \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*)\end{bmatrix}\right). \tag{2.21}$$

predictive distribution

Deriving the conditional distribution corresponding to eq. (2.19) we arrive at the key predictive equations for Gaussian process regression

$$\mathbf{f}_*|X, \mathbf{y}, X_* \;\sim\; \mathcal{N}\big(\bar{\mathbf{f}}_*,\; \operatorname{cov}(\mathbf{f}_*)\big), \quad \text{where} \tag{2.22}$$

$$\bar{\mathbf{f}}_* \;\triangleq\; \mathbb{E}[\mathbf{f}_*|X, \mathbf{y}, X_*] \;=\; K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y}, \tag{2.23}$$

$$\operatorname{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*). \tag{2.24}$$

[8] There are some situations where it is reasonable to assume that the observations are noise-free, for example for computer simulations, see e.g. Sacks et al. [1989].

[9] More complicated noise models with non-trivial covariance structure can also be handled, see section 9.2.

[10] Notice that the Kronecker delta is on the index of the cases, not the value of the input; for the signal part of the covariance function the input *value* is the index set to the random variables describing the function, for the noise part it is the *identity* of the point.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

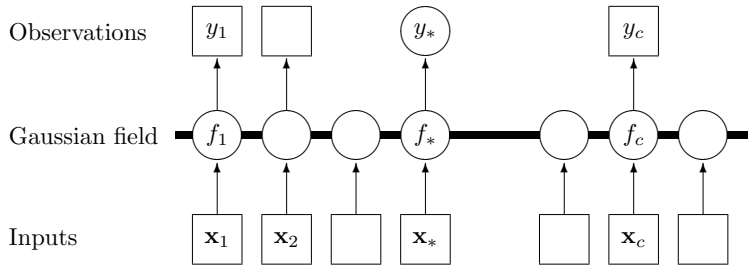


Figure 2.3: Graphical model (chain graph) for a GP for regression. Squares represent observed variables and circles represent unknowns. The thick horizontal bar represents a set of fully connected nodes. Note that an observation y_i is conditionally independent of all other nodes given the corresponding latent variable, f_i. Because of the marginalization property of GPs addition of further inputs, $\mathbf{x}$, latent variables, f, and *unobserved* targets, y_*, does not change the distribution of any other variables.

Notice that we now have exact correspondence with the weight space view in eq. (2.12) when identifying $K(C,D) = \Phi(C)^\top \Sigma_p \Phi(D)$, where C, D stand for either X or X_*. For any set of basis functions, we can compute the corresponding covariance function as $k(\mathbf{x}_p, \mathbf{x}_q) = \boldsymbol{\phi}(\mathbf{x}_p)^\top \Sigma_p \boldsymbol{\phi}(\mathbf{x}_q)$; conversely, for every (positive definite) covariance function k, there exists a (possibly infinite) expansion in terms of basis functions, see section 4.3.

correspondence with weight-space view

The expressions involving $K(X,X)$, $K(X,X_*)$ and $K(X_*,X_*)$ etc. can look rather unwieldy, so we now introduce a compact form of the notation setting $K = K(X,X)$ and $K_* = K(X,X_*)$. In the case that there is only one test point $\mathbf{x}_*$ we write $\mathbf{k}(\mathbf{x}_*) = \mathbf{k}_*$ to denote the vector of covariances between the test point and the n training points. Using this compact notation and for a single test point $\mathbf{x}_*$, equations 2.23 and 2.24 reduce to

compact notation

$$\bar{f}_* = \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{y}, \tag{2.25}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*. \tag{2.26}$$

Let us examine the predictive distribution as given by equations 2.25 and 2.26. Note first that the mean prediction eq. (2.25) is a linear combination of observations $\mathbf{y}$; this is sometimes referred to as a *linear predictor*. Another way to look at this equation is to see it as a linear combination of n kernel functions, each one centered on a training point, by writing

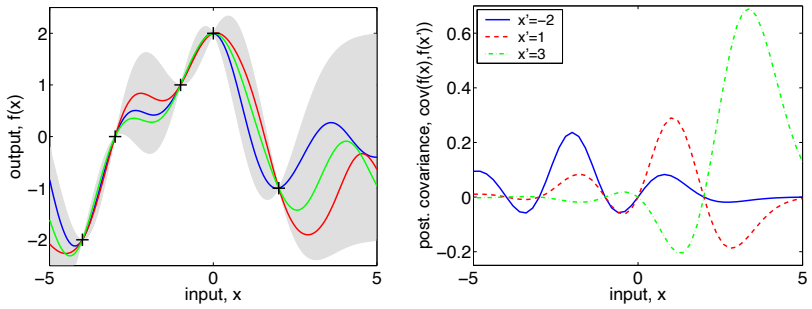
predictive distribution

linear predictor

$$\bar{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*) \tag{2.27}$$

where $\boldsymbol{\alpha} = (K + \sigma_n^2 I)^{-1} \mathbf{y}$. The fact that the mean prediction for $f(\mathbf{x}_*)$ can be written as eq. (2.27) despite the fact that the GP can be represented in terms of a (possibly infinite) number of basis functions is one manifestation of the *representer theorem*; see section 6.2 for more on this point. We can understand this result intuitively because although the GP defines a joint Gaussian distribution over all of the y variables, one for each point in the index set $\mathcal{X}$, for

representer theorem



(a), posterior (b), posterior covariance

Figure 2.4: Panel (a) is identical to Figure 2.2(b) showing three random functions drawn from the posterior. Panel (b) shows the posterior *co*-variance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ for the same data for three different values of $\mathbf{x}'$. Note, that the covariance at close points is high, falling to zero at the training points (where there is no variance, since it is a noise-free process), then becomes negative, etc. This happens because if the smooth function happens to be less than the mean on one side of the data point, it tends to exceed the mean on the other side, causing a reversal of the sign of the covariance at the data points. Note for contrast that the *prior* covariance is simply of Gaussian shape and never negative.

making predictions at $\mathbf{x}_*$ we only care about the $(n+1)$-dimensional distribution defined by the n training points and the test point. As a Gaussian distribution is marginalized by just taking the relevant block of the joint covariance matrix (see section A.2) it is clear that conditioning this $(n+1)$-dimensional distribution on the observations gives us the desired result. A graphical model representation of a GP is given in Figure 2.3.

Note also that the variance in eq. (2.24) does not depend on the observed targets, but only on the inputs; this is a property of the Gaussian distribution. The variance is the difference between two terms: the first term $K(X_*, X_*)$ is simply the prior covariance; from that is subtracted a (positive) term, representing the information the observations gives us about the function. We can

noisy predictions

very simply compute the predictive distribution of test targets $\mathbf{y}_*$ by adding $\sigma_n^2 I$ to the variance in the expression for $\mathrm{cov}(\mathbf{f}_*)$.

joint predictions

The predictive distribution for the GP model gives more than just pointwise errorbars of the simplified eq. (2.26). Although not stated explicitly, eq. (2.24) holds unchanged when X_* denotes multiple test inputs; in this case the *co*-variance of the test targets are computed (whose diagonal elements are the pointwise variances). In fact, eq. (2.23) is the mean function and eq. (2.24) the

posterior process

covariance function of the (Gaussian) posterior process; recall the definition of Gaussian process from page 13. The posterior covariance in illustrated in Figure 2.4(b).

It will be useful (particularly for chapter 5) to introduce the *marginal likelihood* (or evidence) $p(\mathbf{y}|X)$ at this point. The marginal likelihood is the integral

marginal likelihood

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_n^2 (noise level), $\mathbf{x}_*$ (test input)
2: $L := \text{cholesky}(K + \sigma_n^2 I)$
$\boldsymbol{\alpha} := L^\top \backslash (L \backslash \mathbf{y})$ } predictive mean eq. (2.25)
4: $\bar{f}_* := \mathbf{k}_*^\top \boldsymbol{\alpha}$
$\mathbf{v} := L \backslash \mathbf{k}_*$ } predictive variance eq. (2.26)
6: $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^\top \mathbf{v}$
$\log p(\mathbf{y}|X) := -\frac{1}{2}\mathbf{y}^\top \boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$ eq. (2.30)
8: **return**: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood)

Algorithm 2.1: Predictions and log marginal likelihood for Gaussian process regression. The implementation addresses the matrix inversion required by eq. (2.25) and (2.26) using Cholesky factorization, see section A.4. For multiple test cases lines 4-6 are repeated. The log determinant required in eq. (2.30) is computed from the Cholesky factor (for large n it may not be possible to represent the determinant itself). The computational complexity is $n^3/6$ for the Cholesky decomposition in line 2, and $n^2/2$ for solving triangular systems in line 3 and (for each test case) in line 5.

of the likelihood times the prior

$$p(\mathbf{y}|X) \;=\; \int p(\mathbf{y}|\mathbf{f}, X) p(\mathbf{f}|X)\, d\mathbf{f}. \tag{2.28}$$

The term *marginal* likelihood refers to the marginalization over the function values $\mathbf{f}$. Under the Gaussian process model the prior is Gaussian, $\mathbf{f}|X \sim \mathcal{N}(\mathbf{0}, K)$, or

$$\log p(\mathbf{f}|X) \;=\; -\tfrac{1}{2}\mathbf{f}^\top K^{-1}\mathbf{f} - \tfrac{1}{2}\log|K| - \tfrac{n}{2}\log 2\pi, \tag{2.29}$$

and the likelihood is a factorized Gaussian $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 I)$ so we can make use of equations A.7 and A.8 to perform the integration yielding the log marginal likelihood

$$\log p(\mathbf{y}|X) \;=\; -\tfrac{1}{2}\mathbf{y}^\top (K + \sigma_n^2 I)^{-1}\mathbf{y} - \tfrac{1}{2}\log|K + \sigma_n^2 I| - \tfrac{n}{2}\log 2\pi. \tag{2.30}$$

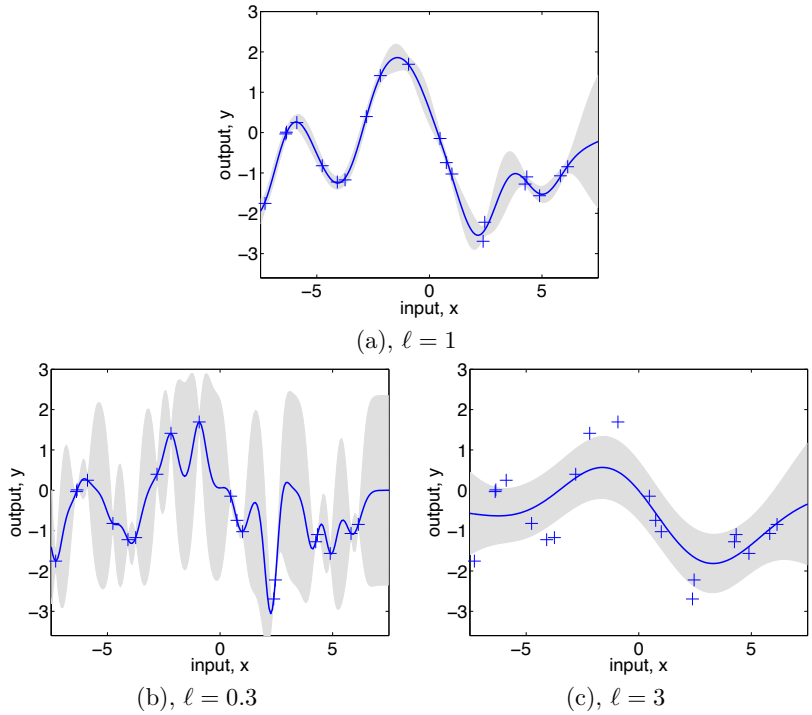
This result can also be obtained directly by observing that $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 I)$.

A practical implementation of Gaussian process regression (GPR) is shown in Algorithm 2.1. The algorithm uses Cholesky decomposition, instead of directly inverting the matrix, since it is faster and numerically more stable, see section A.4. The algorithm returns the predictive mean and variance for noise free test data—to compute the predictive distribution for noisy test data y_*, simply add the noise variance σ_n^2 to the predictive variance of f_*.

2.3 Varying the Hyperparameters CHAPTER 5

Typically the covariance functions that we use will have some free parameters. For example, the squared-exponential covariance function in one dimension has the following form

$$k_y(x_p, x_q) \;=\; \sigma_f^2 \exp\big(-\frac{1}{2\ell^2}(x_p - x_q)^2\big) + \sigma_n^2 \delta_{pq}. \tag{2.31}$$



(a), $\ell = 1$ (b), $\ell = 0.3$ (c), $\ell = 3$

Figure 2.5: (a) Data is generated from a GP with hyperparameters $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$, as shown by the + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 95% confidence region for the underlying function f (shown in grey). Panels (b) and (c) again show the 95% confidence region, but this time for hyperparameter values $(0.3, 1.08, 0.00005)$ and $(3.0, 1.16, 0.89)$ respectively.

hyperparameters

The covariance is denoted k_y as it is for the noisy targets y rather than for the underlying function f. Observe that the length-scale ℓ, the signal variance σ_f^2 and the noise variance σ_n^2 can be varied. In general we call the free parameters *hyperparameters*.[11]

In chapter 5 we will consider various methods for determining the hyperparameters from training data. However, in this section our aim is more simply to explore the effects of varying the hyperparameters on GP prediction. Consider the data shown by + signs in Figure 2.5(a). This was generated from a GP with the SE kernel with $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$. The figure also shows the 2 standard-deviation error bars for the predictions obtained using these values of the hyperparameters, as per eq. (2.24). Notice how the error bars get larger for input values that are distant from any training points. Indeed if the x-axis

[11] We refer to the parameters of the covariance function as hyperparameters to emphasize that they are parameters of a non-parametric model; in accordance with the weight-space view, section 2.1, the parameters (weights) of the underlying parametric model have been integrated out.

were extended one would see the error bars reflect the prior standard deviation of the process σ_f away from the data.

If we set the length-scale shorter so that $\ell = 0.3$ and kept the other parameters the same, then generating from this process we would expect to see plots like those in Figure 2.5(a) except that the x-axis should be rescaled by a factor of 0.3; equivalently if the same x-axis was kept as in Figure 2.5(a) then a sample function would look much more wiggly.

too short length-scale

If we make predictions with a process with $\ell = 0.3$ on the data generated from the $\ell = 1$ process then we obtain the result in Figure 2.5(b). The remaining two parameters were set by optimizing the marginal likelihood, as explained in chapter 5. In this case the noise parameter is reduced to $\sigma_n = 0.00005$ as the greater flexibility of the "signal" means that the noise level can be reduced. This can be observed at the two datapoints near $x = 2.5$ in the plots. In Figure 2.5(a) ($\ell = 1$) these are essentially explained as a similar function value with differing noise. However, in Figure 2.5(b) ($\ell = 0.3$) the noise level is very low, so these two points have to be explained by a sharp variation in the value of the underlying function f. Notice also that the short length-scale means that the error bars in Figure 2.5(b) grow rapidly away from the datapoints.

too long length-scale

In contrast, we can set the length-scale longer, for example to $\ell = 3$, as shown in Figure 2.5(c). Again the remaining two parameters were set by optimizing the marginal likelihood. In this case the noise level has been increased to $\sigma_n = 0.89$ and we see that the data is now explained by a slowly varying function with a lot of noise.

Of course we can take the position of a quickly-varying signal with low noise, or a slowly-varying signal with high noise to extremes; the former would give rise to a white-noise process model for the signal, while the latter would give rise to a constant signal with added white noise. Under both these models the datapoints produced should look like white noise. However, studying Figure 2.5(a) we see that white noise is not a convincing model of the data, as the sequence of y's does not alternate sufficiently quickly but has correlations due to the variability of the underlying function. Of course this is relatively easy to see in one dimension, but methods such as the marginal likelihood discussed in chapter 5 generalize to higher dimensions and allow us to score the various models. In this case the marginal likelihood gives a clear preference for $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$ over the other two alternatives.

model comparison

2.4 Decision Theory for Regression

In the previous sections we have shown how to compute predictive distributions for the outputs y_* corresponding to the novel test input $\mathbf{x}_*$. The predictive distribution is Gaussian with mean and variance given by eq. (2.25) and eq. (2.26). In practical applications, however, we are often forced to make a decision about how to act, i.e. we need a point-like prediction which is optimal in some sense. To this end we need a *loss function*, $\mathcal{L}(y_{\text{true}}, y_{\text{guess}})$, which specifies the loss (or

optimal predictions

loss function

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

penalty) incurred by guessing the value y_{guess} when the true value is y_{true}. For example, the loss function could equal the absolute deviation between the guess and the truth.

non-Bayesian paradigm
Bayesian paradigm

Notice that we computed the predictive distribution without reference to the loss function. In non-Bayesian paradigms, the model is typically trained by minimizing the empirical risk (or loss). In contrast, in the Bayesian setting there is a clear separation between the likelihood function (used for training, in addition to the prior) and the loss function. The likelihood function describes how the noisy measurements are assumed to deviate from the underlying noise-free function. The loss function, on the other hand, captures the consequences of making a specific choice, given an actual true state. The likelihood and loss function need not have anything in common.[12]

expected loss, risk

Our goal is to make the point prediction y_{guess} which incurs the smallest loss, but how can we achieve that when we don't know y_{true}? Instead, we minimize the *expected loss* or *risk*, by averaging w.r.t. our model's opinion as to what the truth might be

$$\tilde{R}_{\mathcal{L}}(y_{\text{guess}}|\mathbf{x}_*) = \int \mathcal{L}(y_*, y_{\text{guess}}) p(y_*|\mathbf{x}_*, \mathcal{D})\, dy_*. \tag{2.32}$$

Thus our best guess, in the sense that it minimizes the expected loss, is

$$y_{\text{optimal}}|\mathbf{x}_* = \underset{y_{\text{guess}}}{\operatorname{argmin}}\, \tilde{R}_{\mathcal{L}}(y_{\text{guess}}|\mathbf{x}_*). \tag{2.33}$$

absolute error loss
squared error loss

In general the value of y_{guess} that minimizes the risk for the loss function $|y_{\text{guess}} - y_*|$ is the median of $p(y_*|\mathbf{x}_*, \mathcal{D})$, while for the squared loss $(y_{\text{guess}} - y_*)^2$ it is the mean of this distribution. When the predictive distribution is Gaussian the mean and the median coincide, and indeed for any symmetric loss function and symmetric predictive distribution we always get y_{guess} as the mean of the predictive distribution. However, in many practical problems the loss functions can be asymmetric, e.g. in safety critical applications, and point predictions may be computed directly from eq. (2.32) and eq. (2.33). A comprehensive treatment of decision theory can be found in Berger [1985].

2.5 An Example Application

robot arm

In this section we use Gaussian process regression to learn the inverse dynamics of a seven degrees-of-freedom SARCOS anthropomorphic robot arm. The task is to map from a 21-dimensional input space (7 joint positions, 7 joint velocities, 7 joint accelerations) to the corresponding 7 joint torques. This task has previously been used to study regression algorithms by Vijayakumar and Schaal [2000], Vijayakumar et al. [2002] and Vijayakumar et al. [2005].[13] Following

[12] Beware of fallacious arguments like: a Gaussian likelihood implies a squared error loss function.

[13] We thank Sethu Vijayakumar for providing us with the data.

this previous work we present results below on just one of the seven mappings, from the 21 input variables to the first of the seven torques.

One might ask why it is necessary to *learn* this mapping; indeed there exist physics-based rigid-body-dynamics models which allow us to obtain the torques from the position, velocity and acceleration variables. However, the real robot arm is actuated hydraulically and is rather lightweight and compliant, so the assumptions of the rigid-body-dynamics model are violated (as we see below). It is worth noting that the rigid-body-dynamics model is nonlinear, involving trigonometric functions and squares of the input variables. why learning?

An inverse dynamics model can be used in the following manner: a planning module decides on a trajectory that takes the robot from its start to goal states, and this specifies the desired positions, velocities and accelerations at each time. The inverse dynamics model is used to compute the torques needed to achieve this trajectory and errors are corrected using a feedback controller.

The dataset consists of 48,933 input-output pairs, of which 44,484 were used as a training set and the remaining 4,449 were used as a test set. The inputs were linearly rescaled to have zero mean and unit variance on the training set. The outputs were centered so as to have zero mean on the training set.

Given a prediction method, we can evaluate the quality of predictions in several ways. Perhaps the simplest is the squared error loss, where we compute the squared residual $(y_* - \bar{f}(\mathbf{x}_*))^2$ between the mean prediction and the target at each test point. This can be summarized by the mean squared error (MSE), by averaging over the test set. However, this quantity is sensitive to the overall scale of the target values, so it makes sense to normalize by the variance of the targets of the test cases to obtain the *standardized mean squared error* (SMSE). This causes the trivial method of guessing the mean of the training targets to have a SMSE of approximately 1. MSE SMSE

Additionally if we produce a predictive distribution at each test input we can evaluate the negative log probability of the target under the model.[14] As GPR produces a Gaussian predictive density, one obtains

$$-\log p(y_*|\mathcal{D}, \mathbf{x}_*) \;=\; \frac{1}{2}\log(2\pi\sigma_*^2) + \frac{(y_* - \bar{f}(\mathbf{x}_*))^2}{2\sigma_*^2}, \qquad (2.34)$$

where the predictive variance σ_*^2 for GPR is computed as $\sigma_*^2 = \mathbb{V}(f_*) + \sigma_n^2$, where $\mathbb{V}(f_*)$ is given by eq. (2.26); we must include the noise variance σ_n^2 as we are predicting the noisy target y_*. This loss can be standardized by subtracting the loss that would be obtained under the trivial model which predicts using a Gaussian with the mean and variance of the training data. We denote this the standardized log loss (SLL). The mean SLL is denoted MSLL. Thus the MSLL will be approximately zero for simple methods and negative for better methods. MSLL

A number of models were tested on the data. A linear regression (LR) model provides a simple baseline for the SMSE. By estimating the noise level from the

[14] It makes sense to use the *negative* log probability so as to obtain a loss, not a utility.

Method	SMSE	MSLL
LR	0.075	-1.29
RBD	0.104	–
LWPR	0.040	–
GPR	0.011	-2.25

Table 2.1: Test results on the inverse dynamics problem for a number of different methods. The "–" denotes a missing entry, caused by two methods not producing full predictive *distributions*, so MSLL could not be evaluated.

residuals on the training set one can also obtain a predictive variance and thus get a MSLL value for LR. The rigid-body-dynamics (RBD) model has a number of free parameters; these were estimated by Vijayakumar et al. [2005] using a least-squares fitting procedure. We also give results for the locally weighted projection regression (LWPR) method of Vijayakumar et al. [2005] which is an on-line method that cycles through the dataset multiple times. For the GP models it is computationally expensive to make use of all 44,484 training cases due to the $O(n^3)$ scaling of the basic algorithm. In chapter 8 we present several different approximate GP methods for large datasets. The result given in Table 2.1 was obtained with the subset of regressors (SR) approximation with a subset size of 4096. This result is taken from Table 8.1, which gives full results of the various approximation methods applied to the inverse dynamics problem. The squared exponential covariance function was used with a separate length-scale parameter for each of the 21 input dimensions, plus the signal and noise variance parameters σ_f^2 and σ_n^2. These parameters were set by optimizing the marginal likelihood eq. (2.30) on a subset of the data (see also chapter 5).

The results for the various methods are presented in Table 2.1. Notice that the problem is quite non-linear, so the linear regression model does poorly in comparison to non-linear methods.[15] The non-linear method LWPR improves over linear regression, but is outperformed by GPR.

2.6 Smoothing, Weight Functions and Equivalent Kernels

linear smoother

Gaussian process regression aims to reconstruct the underlying signal f by removing the contaminating noise ε. To do this it computes a weighted average of the noisy observations $\mathbf{y}$ as $\bar{f}(\mathbf{x}_*) = \mathbf{k}(\mathbf{x}_*)^\top (K + \sigma_n^2 I)^{-1}\mathbf{y}$; as $\bar{f}(\mathbf{x}_*)$ is a *linear* combination of the y values, Gaussian process regression is a *linear smoother* (see Hastie and Tibshirani [1990, sec. 2.8] for further details). In this section we study smoothing first in terms of a matrix analysis of the predictions at the training points, and then in terms of the equivalent kernel.

[15] It is perhaps surprising that RBD does worse than linear regression. However, Stefan Schaal (pers. comm., 2004) states that the RBD parameters were optimized on a very large dataset, of which the training data used here is subset, and if the RBD model were optimized w.r.t. this training set one might well expect it to outperform linear regression.

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

The predicted mean values $\bar{\mathbf{f}}$ at the training points are given by

$$\bar{\mathbf{f}} \;=\; K(K+\sigma_n^2 I)^{-1}\mathbf{y}. \tag{2.35}$$

Let K have the eigendecomposition $K = \sum_{i=1}^n \lambda_i \mathbf{u}_i \mathbf{u}_i^\top$, where λ_i is the ith eigenvalue and $\mathbf{u}_i$ is the corresponding eigenvector. As K is real and symmetric positive semidefinite, its eigenvalues are real and non-negative, and its eigenvectors are mutually orthogonal. Let $\mathbf{y} = \sum_{i=1}^n \gamma_i \mathbf{u}_i$ for some coefficients $\gamma_i = \mathbf{u}_i^\top \mathbf{y}$. Then eigendecomposition

$$\bar{\mathbf{f}} \;=\; \sum_{i=1}^n \frac{\gamma_i \lambda_i}{\lambda_i + \sigma_n^2}\mathbf{u}_i. \tag{2.36}$$

Notice that if $\lambda_i/(\lambda_i+\sigma_n^2) \ll 1$ then the component in $\mathbf{y}$ along $\mathbf{u}_i$ is effectively eliminated. For most covariance functions that are used in practice the eigenvalues are larger for more slowly varying eigenvectors (e.g. fewer zero-crossings) so that this means that high-frequency components in $\mathbf{y}$ are smoothed out. The effective number of parameters or degrees of freedom of the smoother is defined as $\mathrm{tr}(K(K+\sigma_n^2 I)^{-1}) = \sum_{i=1}^n \lambda_i/(\lambda_i+\sigma_n^2)$, see Hastie and Tibshirani [1990, sec. 3.5]. Notice that this counts the number of eigenvectors which are not eliminated. degrees of freedom

We can define a vector of functions $\mathbf{h}(\mathbf{x}_*) = (K+\sigma_n^2 I)^{-1}\mathbf{k}(\mathbf{x}_*)$. Thus we have $\bar{f}(\mathbf{x}_*) = \mathbf{h}(\mathbf{x}_*)^\top \mathbf{y}$, making it clear that the mean prediction at a point $\mathbf{x}_*$ is a linear combination of the target values $\mathbf{y}$. For a fixed test point $\mathbf{x}_*$, $\mathbf{h}(\mathbf{x}_*)$ gives the vector of weights applied to targets $\mathbf{y}$. $\mathbf{h}(\mathbf{x}_*)$ is called the *weight function* [Silverman, 1984]. As Gaussian process regression is a linear smoother, the weight function does not depend on $\mathbf{y}$. Note the difference between a linear *model*, where the prediction is a linear combination of the *inputs*, and a linear *smoother*, where the prediction is a linear combination of the training set targets. weight function

Understanding the form of the weight function is made complicated by the matrix inversion of $K+\sigma_n^2 I$ and the fact that K depends on the specific locations of the n datapoints. Idealizing the situation one can consider the observations to be "smeared out" in $\mathbf{x}$-space at some density of observations. In this case analytic tools can be brought to bear on the problem, as shown in section 7.1. By analogy to kernel smoothing, Silverman [1984] called the idealized weight function the *equivalent kernel*; see also Girosi et al. [1995, sec. 2.1]. equivalent kernel

A kernel smoother centres a kernel function[16] κ on $\mathbf{x}_*$ and then computes $\kappa_i = \kappa(|\mathbf{x}_i - \mathbf{x}_*|/\ell)$ for each data point $(\mathbf{x}_i, y_i)$, where ℓ is a length-scale. The Gaussian is a commonly used kernel function. The prediction for $f(\mathbf{x}_*)$ is computed as $\hat{f}(\mathbf{x}_*) = \sum_{i=1}^n w_i y_i$ where $w_i = \kappa_i / \sum_{j=1}^n \kappa_j$. This is also known as the Nadaraya-Watson estimator, see e.g. Scott [1992, sec. 8.1]. kernel smoother

The weight function and equivalent kernel for a Gaussian process are illustrated in Figure 2.6 for a one-dimensional input variable x. We have used the squared exponential covariance function and have set the length-scale $\ell = 0.0632$ (so that $\ell^2 = 0.004$). There are $n = 50$ training points spaced randomly along

[16] Note that this kernel function does not need to be a valid covariance function.

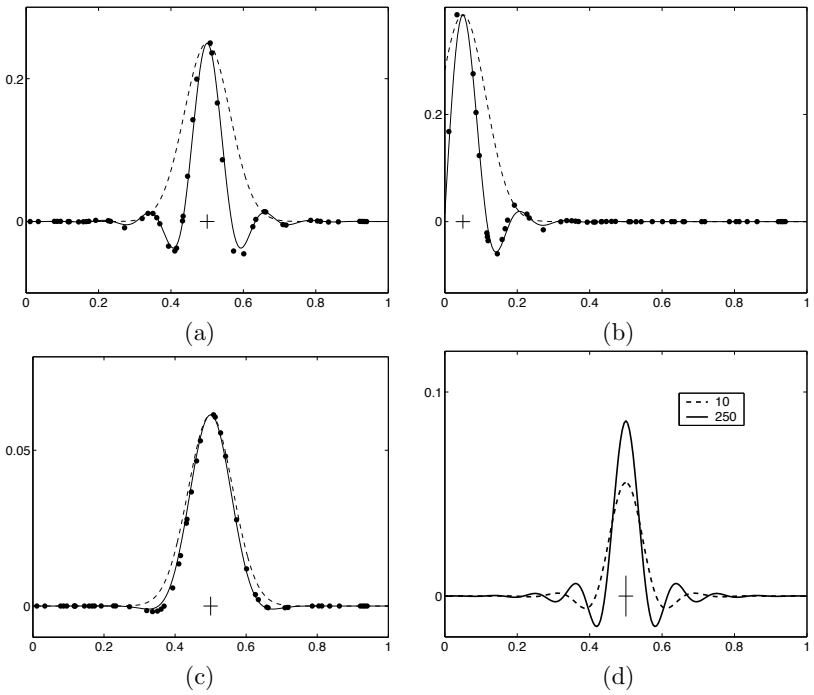


Figure 2.6: Panels (a)-(c) show the weight function $\mathbf{h}(x_*)$ (dots) corresponding to the $n = 50$ training points, the equivalent kernel (solid) and the original squared exponential kernel (dashed). Panel (d) shows the equivalent kernels for two different data densities. See text for further details. The small cross at the test point is to scale in all four plots.

the x-axis. Figures 2.6(a) and 2.6(b) show the weight function and equivalent kernel for $x_* = 0.5$ and $x_* = 0.05$ respectively, for $\sigma_n^2 = 0.1$. Figure 2.6(c) is also for $x_* = 0.5$ but uses $\sigma_n^2 = 10$. In each case the dots correspond to the weight function $\mathbf{h}(x_*)$ and the solid line is the equivalent kernel, whose construction is explained below. The dashed line shows a squared exponential kernel centered on the test point, scaled to have the same height as the maximum value in the equivalent kernel. Figure 2.6(d) shows the variation in the equivalent kernel as a function of n, the number of datapoints in the unit interval.

Many interesting observations can be made from these plots. Observe that the equivalent kernel has (in general) a shape quite different to the original SE kernel. In Figure 2.6(a) the equivalent kernel is clearly oscillatory (with negative sidelobes) and has a higher spatial frequency than the original kernel. Figure 2.6(b) shows similar behaviour although due to edge effects the equivalent kernel is truncated relative to that in Figure 2.6(a). In Figure 2.6(c) we see that at higher noise levels the negative sidelobes are reduced and the width of the equivalent kernel is similar to the original kernel. Also note that the overall height of the equivalent kernel in (c) is reduced compared to that in (a) and

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

(b)—it averages over a wider area. The more oscillatory equivalent kernel for lower noise levels can be understood in terms of the eigenanalysis above; at higher noise levels only the large λ (slowly varying) components of $\mathbf{y}$ remain, while for smaller noise levels the more oscillatory components are also retained.

In Figure 2.6(d) we have plotted the equivalent kernel for $n = 10$ and $n = 250$ datapoints in $[0, 1]$; notice how the width of the equivalent kernel decreases as n increases. We discuss this behaviour further in section 7.1.

The plots of equivalent kernels in Figure 2.6 were made by using a dense grid of n_{grid} points on $[0, 1]$ and then computing the smoother matrix $K(K + \sigma^2_{\text{grid}} I)^{-1}$. Each row of this matrix is the equivalent kernel at the appropriate location. However, in order to get the scaling right one has to set $\sigma^2_{\text{grid}} = \sigma^2_n n_{\text{grid}}/n$; for $n_{\text{grid}} > n$ this means that the effective variance at each of the n_{grid} points is larger, but as there are correspondingly more points this effect cancels out. This can be understood by imagining the situation if there were n_{grid}/n independent Gaussian observations with variance σ^2_{grid} at a single x-position; this would be equivalent to one Gaussian observation with variance σ^2_n. In effect the n observations have been smoothed out uniformly along the interval. The form of the equivalent kernel can be obtained analytically if we go to the continuum limit and look to smooth a noisy function. The relevant theory and some example equivalent kernels are given in section 7.1.

2.7 Incorporating Explicit Basis Functions *

It is common but by no means necessary to consider GPs with a zero mean function. Note that this is not necessarily a drastic limitation, since the mean of the *posterior* process is not confined to be zero. Yet there are several reasons why one might wish to explicitly model a mean function, including interpretability of the model, convenience of expressing prior information and a number of analytical limits which we will need in subsequent chapters. The use of explicit basis functions is a way to specify a non-zero mean over functions, but as we will see in this section, one can also use them to achieve other interesting effects.

fixed mean function

Using a *fixed* (deterministic) mean function $m(\mathbf{x})$ is trivial: Simply apply the usual zero mean GP to the *difference* between the observations and the fixed mean function. With

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}),\, k(\mathbf{x}, \mathbf{x}')\big), \tag{2.37}$$

the predictive mean becomes

$$\bar{\mathbf{f}}_* = \mathbf{m}(X_*) + K(X_*, X)K_y^{-1}(\mathbf{y} - \mathbf{m}(X)), \tag{2.38}$$

where $K_y = K + \sigma^2_n I$, and the predictive variance remains unchanged from eq. (2.24).

However, in practice it can often be difficult to specify a fixed mean function. In many cases it may be more convenient to specify a few fixed basis functions,

C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. www.GaussianProcess.org/gpml

stochastic mean function

whose coefficients, $\boldsymbol{\beta}$, are to be inferred from the data. Consider

$$g(\mathbf{x}) \;=\; f(\mathbf{x}) + \mathbf{h}(\mathbf{x})^\top \boldsymbol{\beta}, \quad \text{where} \quad f(\mathbf{x}) \;\sim\; \mathcal{GP}\big(0, k(\mathbf{x}, \mathbf{x}')\big), \tag{2.39}$$

here $f(\mathbf{x})$ is a zero mean GP, $\mathbf{h}(\mathbf{x})$ are a set of fixed basis functions, and $\boldsymbol{\beta}$ are additional parameters. This formulation expresses that the data is close to a global linear model with the residuals being modelled by a GP. This idea was explored explicitly as early as 1975 by Blight and Ott [1975], who used the GP to model the residuals from a polynomial regression, i.e. $\mathbf{h}(x) = (1, x, x^2, \ldots)$. When fitting the model, one could optimize over the parameters $\boldsymbol{\beta}$ jointly with the hyperparameters of the covariance function. Alternatively, if we take the prior on $\boldsymbol{\beta}$ to be Gaussian, $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, B)$, we can also integrate out these parameters. Following O'Hagan [1978] we obtain another GP

polynomial regression

$$g(\mathbf{x}) \;\sim\; \mathcal{GP}\big(\mathbf{h}(\mathbf{x})^\top \mathbf{b},\; k(\mathbf{x}, \mathbf{x}') + \mathbf{h}(\mathbf{x})^\top B \mathbf{h}(\mathbf{x}')\big), \tag{2.40}$$

now with an added contribution in the covariance function caused by the uncertainty in the parameters of the mean. Predictions are made by plugging the mean and covariance functions of $g(\mathbf{x})$ into eq. (2.39) and eq. (2.24). After rearranging, we obtain

$$\begin{aligned} \bar{\mathbf{g}}(X_*) \;&=\; H_*^\top \bar{\boldsymbol{\beta}} + K_*^\top K_y^{-1}(\mathbf{y} - H^\top \bar{\boldsymbol{\beta}}) \;=\; \bar{\mathbf{f}}(X_*) + R^\top \bar{\boldsymbol{\beta}}, \\ \operatorname{cov}(\mathbf{g}_*) \;&=\; \operatorname{cov}(\mathbf{f}_*) + R^\top (B^{-1} + H K_y^{-1} H^\top)^{-1} R, \end{aligned} \tag{2.41}$$

where the H matrix collects the $\mathbf{h}(\mathbf{x})$ vectors for all training (and H_* all test) cases, $\bar{\boldsymbol{\beta}} = (B^{-1} + H K_y^{-1} H^\top)^{-1}(H K_y^{-1}\mathbf{y} + B^{-1}\mathbf{b})$, and $R = H_* - H K_y^{-1} K_*$. Notice the nice interpretation of the mean expression, eq. (2.41) top line: $\bar{\boldsymbol{\beta}}$ is the mean of the global linear model parameters, being a compromise between the data term and prior, and the predictive mean is simply the mean linear output plus what the GP model predicts from the residuals. The covariance is the sum of the usual covariance term and a new non-negative contribution.

Exploring the limit of the above expressions as the prior on the $\boldsymbol{\beta}$ parameter becomes vague, $B^{-1} \to O$ (where O is the matrix of zeros), we obtain a predictive distribution which is independent of $\mathbf{b}$

$$\begin{aligned} \bar{\mathbf{g}}(X_*) \;&=\; \bar{\mathbf{f}}(X_*) + R^\top \bar{\boldsymbol{\beta}}, \\ \operatorname{cov}(\mathbf{g}_*) \;&=\; \operatorname{cov}(\mathbf{f}_*) + R^\top (H K_y^{-1} H^\top)^{-1} R, \end{aligned} \tag{2.42}$$

where the limiting $\bar{\boldsymbol{\beta}} = (H K_y^{-1} H^\top)^{-1} H K_y^{-1} \mathbf{y}$. Notice that predictions under the limit $B^{-1} \to O$ should not be implemented naïvely by plugging the modified covariance function from eq. (2.40) into the standard prediction equations, since the entries of the covariance function tend to infinity, thus making it unsuitable for numerical implementation. Instead eq. (2.42) must be used. Even if the non-limiting case is of interest, eq. (2.41) is numerically preferable to a direct implementation based on eq. (2.40), since the global linear part will often add some very large eigenvalues to the covariance matrix, affecting its condition number.

2.7.1 Marginal Likelihood

In this short section we briefly discuss the marginal likelihood for the model with a Gaussian prior $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, B)$ on the explicit parameters from eq. (2.40), as this will be useful later, particularly in section 6.3.1. We can express the marginal likelihood from eq. (2.30) as

$$\begin{aligned} \log p(\mathbf{y}|X, \mathbf{b}, B) \;=\; & -\tfrac{1}{2}(H^\top \mathbf{b} - \mathbf{y})^\top (K_y + H^\top B H)^{-1}(H^\top \mathbf{b} - \mathbf{y}) \\ & -\tfrac{1}{2}\log|K_y + H^\top B H| - \tfrac{n}{2}\log 2\pi, \end{aligned} \tag{2.43}$$

where we have included the explicit mean. We are interested in exploring the limit where $B^{-1} \to O$, i.e. when the prior is vague. In this limit the mean of the prior is irrelevant (as was the case in eq. (2.42)), so without loss of generality (for the limiting case) we assume for now that the mean is zero, $\mathbf{b} = \mathbf{0}$, giving

$$\begin{aligned} \log p(\mathbf{y}|X, \mathbf{b}{=}\mathbf{0}, B) \;=\; & -\tfrac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} + \tfrac{1}{2}\mathbf{y}^\top C \mathbf{y} \\ & -\tfrac{1}{2}\log|K_y| - \tfrac{1}{2}\log|B| - \tfrac{1}{2}\log|A| - \tfrac{n}{2}\log 2\pi, \end{aligned} \tag{2.44}$$

where $A = B^{-1} + H K_y^{-1} H^\top$ and $C = K_y^{-1} H^\top A^{-1} H K_y^{-1}$ and we have used the matrix inversion lemma, eq. (A.9) and eq. (A.10).

We now explore the behaviour of the log marginal likelihood in the limit of vague priors on $\boldsymbol{\beta}$. In this limit the variances of the Gaussian in the directions spanned by columns of $H^\top$ will become infinite, and it is clear that this will require special treatment. The log marginal likelihood consists of three terms: a quadratic form in $\mathbf{y}$, a log determinant term, and a term involving $\log 2\pi$. Performing an eigendecomposition of the covariance matrix we see that the contributions to quadratic form term from the infinite-variance directions will be zero. However, the log determinant term will tend to minus infinity. The standard solution [Wahba, 1985, Ansley and Kohn, 1985] in this case is to project $\mathbf{y}$ onto the directions orthogonal to the span of $H^\top$ and compute the marginal likelihood in this subspace. Let the rank of $H^\top$ be m. Then as shown in Ansley and Kohn [1985] this means that we must discard the terms $-\frac{1}{2}\log|B| - \frac{m}{2}\log 2\pi$ from eq. (2.44) to give

$$\log p(\mathbf{y}|X) \;=\; -\tfrac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} + \tfrac{1}{2}\mathbf{y}^\top C \mathbf{y} - \tfrac{1}{2}\log|K_y| - \tfrac{1}{2}\log|A| - \tfrac{n-m}{2}\log 2\pi, \tag{2.45}$$

where $A = H K_y^{-1} H^\top$ and $C = K_y^{-1} H^\top A^{-1} H K_y^{-1}$.

2.8 History and Related Work

Prediction with Gaussian processes is certainly not a very recent topic, especially for time series analysis; the basic theory goes back at least as far as the work of Wiener [1949] and Kolmogorov [1941] in the 1940's. Indeed Lauritzen [1981] discusses relevant work by the Danish astronomer T. N. Thiele dating from 1880.

time series

geostatistics

Gaussian process prediction is also well known in the geostatistics field (see, e.g. Matheron, 1973; Journel and Huijbregts, 1978) where it is known as *kriging*,[17] and in meteorology [Thompson, 1956, Daley, 1991] although this literature naturally has focussed mostly on two- and three-dimensional input spaces. Whittle [1963, sec. 5.4] also suggests the use of such methods for spatial prediction. Ripley [1981] and Cressie [1993] provide useful overviews of Gaussian process prediction in spatial statistics.

kriging

Gradually it was realized that Gaussian process prediction could be used in a general regression context. For example O'Hagan [1978] presents the general theory as given in our equations 2.23 and 2.24, and applies it to a number of one-dimensional regression problems. Sacks et al. [1989] describe GPR in the context of computer experiments (where the observations y are noise free) and discuss a number of interesting directions such as the optimization of parameters in the covariance function (see our chapter 5) and experimental design (i.e. the choice of $\mathbf{x}$-points that provide most information on f). The authors describe a number of computer simulations that were modelled, including an example where the response variable was the clock asynchronization in a circuit and the inputs were six transistor widths. Santner et al. [2003] is a recent book on the use of GPs for the design and analysis of computer experiments.

computer experiments

machine learning

Williams and Rasmussen [1996] described Gaussian process regression in a machine learning context, and described optimization of the parameters in the covariance function, see also Rasmussen [1996]. They were inspired to use Gaussian process by the connection to infinite neural networks as described in section 4.2.3 and in Neal [1996]. The "kernelization" of linear ridge regression described above is also known as *kernel ridge regression* see e.g. Saunders et al. [1998].

Relationships between Gaussian process prediction and regularization theory, splines, support vector machines (SVMs) and relevance vector machines (RVMs) are discussed in chapter 6.

2.9 Exercises

1. Replicate the generation of random functions from Figure 2.2. Use a regular (or random) grid of scalar inputs and the covariance function from eq. (2.16). Hints on how to generate random samples from multi-variate Gaussian distributions are given in section A.2. Invent some training data points, and make random draws from the resulting GP posterior using eq. (2.19).

2. In eq. (2.11) we saw that the predictive variance at $\mathbf{x}_*$ under the feature space regression model was $\mathrm{var}(f(\mathbf{x}_*)) = \boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1}\boldsymbol{\phi}(\mathbf{x}_*)$. Show that $\mathrm{cov}(f(\mathbf{x}_*), f(\mathbf{x}'_*)) = \boldsymbol{\phi}(\mathbf{x}_*)^\top A^{-1}\boldsymbol{\phi}(\mathbf{x}'_*)$. Check that this is compatible with the expression given in eq. (2.24).

[17] Matheron named the method after the South African mining engineer D. G. Krige.

3. The Wiener process is defined for $x \geq 0$ and has $f(0) = 0$. (See section B.2.1 for further details.) It has mean zero and a non-stationary covariance function $k(x, x') = \min(x, x')$. If we condition on the Wiener process passing through $f(1) = 0$ we obtain a process known as the Brownian bridge (or *tied-down* Wiener process). Show that this process has covariance $k(x, x') = \min(x, x') - xx'$ for $0 \leq x, x' \leq 1$ and mean 0. Write a computer program to draw samples from this process at a finite grid of x points in $[0, 1]$.

4. Let $\mathrm{var}_n(f(\mathbf{x}_*))$ be the predictive variance of a Gaussian process regression model at $\mathbf{x}_*$ given a dataset of size n. The corresponding predictive variance using a dataset of only the first $n-1$ training points is denoted $\mathrm{var}_{n-1}(f(\mathbf{x}_*))$. Show that $\mathrm{var}_n(f(\mathbf{x}_*)) \leq \mathrm{var}_{n-1}(f(\mathbf{x}_*))$, i.e. that the predictive variance at $\mathbf{x}_*$ cannot increase as more training data is obtained. One way to approach this problem is to use the partitioned matrix equations given in section A.3 to decompose $\mathrm{var}_n(f(\mathbf{x}_*)) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*$. An alternative information theoretic argument is given in Williams and Vivarelli [2000]. Note that while this conclusion is true for Gaussian process priors and Gaussian noise models it does not hold generally, see Barber and Saad [1996].