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-

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



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 $\sigma_n^2$

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.

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

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

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 $\Sigma_p$ 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

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

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).

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



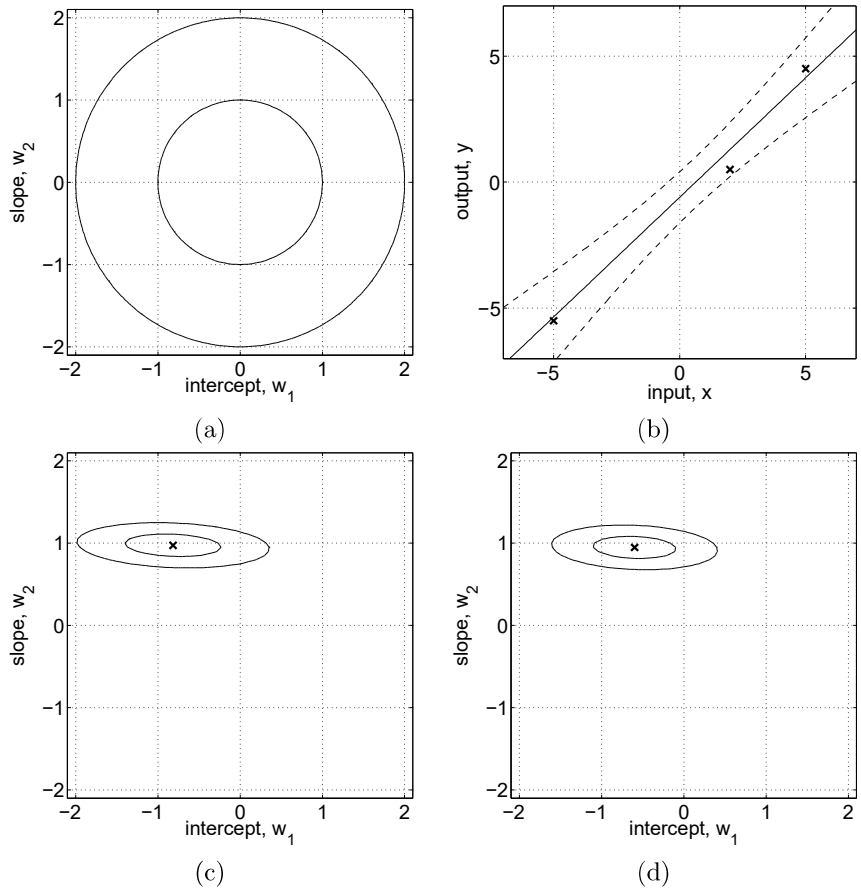



Figure 2.1: Example of Bayesian linear model $f(x) = w_1 + w_2 x$ with intercept $w_1$ and slope parameter $w_2$. Panel (a) shows the contours of the prior distribution $p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, I)$, eq. (2.4). Panel (b) shows three training points marked by crosses. Panel (c) shows 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 $\Sigma_p$). As $\Sigma_p$ 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 $\Sigma_{11}$ is the relevant submatrix of $\Sigma$, 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.

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).

basis functions

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.

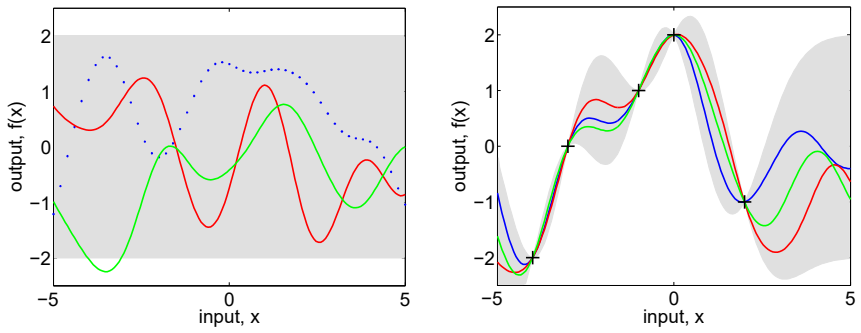
smoothness

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 $\ell$ 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 $\varepsilon$ with variance $\sigma_n^2$, the prior on the noisy observations becomes

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

where $\delta_{pq}$ 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

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

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

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

[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



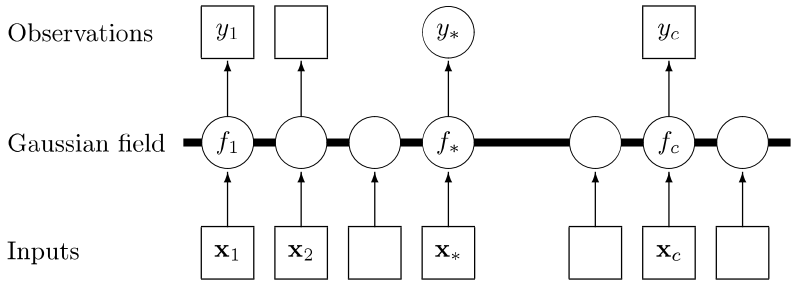



Figure 2.3: Graphical model (chain graph) for a GP for regression. Squares represent observed variables and circles represent unknowns. The thick horizontal bar represents a set of fully connected nodes. Note that an observation $y_i$ is conditionally independent of all other nodes given the corresponding 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

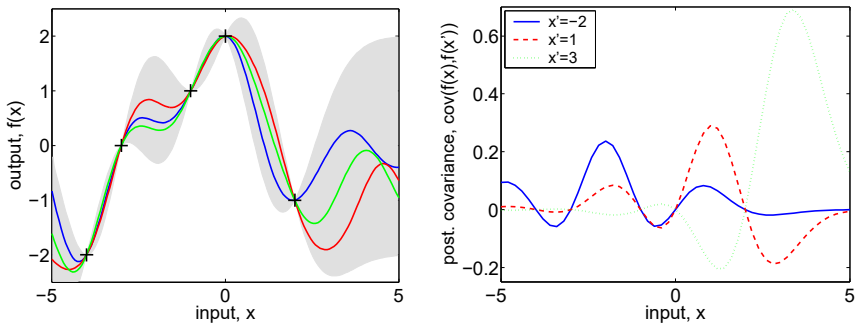
$$\bar{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*) \tag{2.27}$$

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

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






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

noisy predictions

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 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).

posterior process

marginal likelihood

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

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

## 2.3 Varying the Hyperparameters

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

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

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



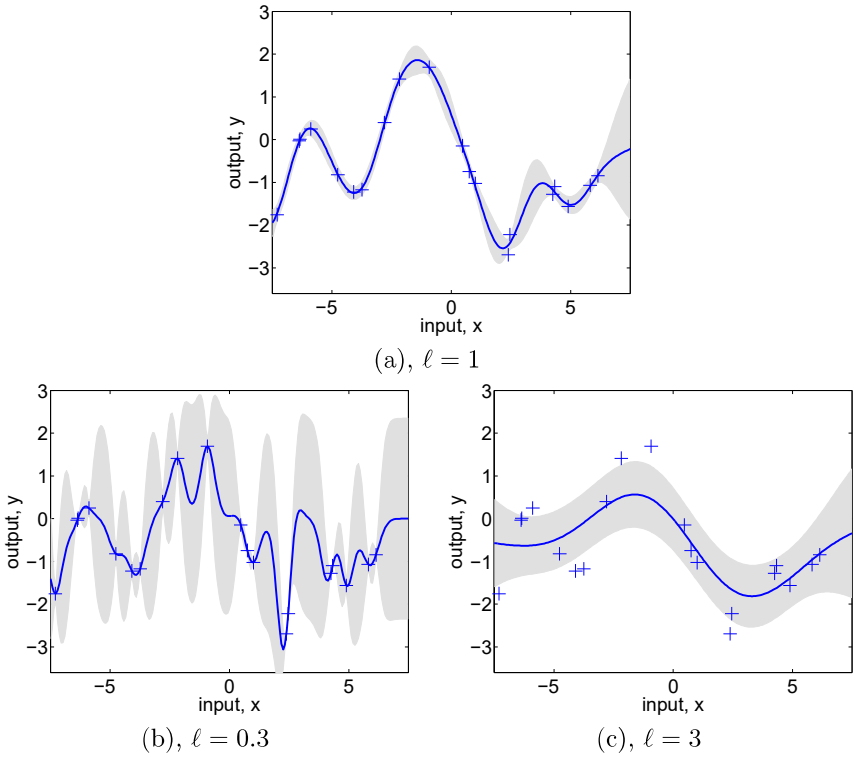



Figure 2.5: (a) Data is generated from a GP with hyperparameters $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$, as shown by the + symbols. Using Gaussian process prediction with these hyperparameters we obtain a 95% confidence region for the underlying function $f$ (shown in grey). Panels (b) and (c) again 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.

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 $\ell$, the signal variance $\sigma_f^2$ and the noise variance $\sigma_n^2$ can be varied. In general we call the free parameters *hyperparameters*.[11]

hyperparameters

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.

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



were extended one would see the error bars reflect the prior standard deviation of the process $\sigma_f$ 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.

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 short 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.

too long length-scale

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 $y_{\text{guess}}$ when the true value is $y_{\text{true}}$. 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 $y_{\text{guess}}$ which incurs the smallest loss, but how can we achieve that when we don't know $y_{\text{true}}$? 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 $y_{\text{guess}}$ 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 $y_{\text{guess}}$ 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 $\sigma_*^2$ 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 $\sigma_n^2$ 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 $\sigma_f^2$ and $\sigma_n^2$. 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 $\varepsilon$. 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 $\lambda_i$ is the $i$th 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] $\kappa$ 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 $\ell$ 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.

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



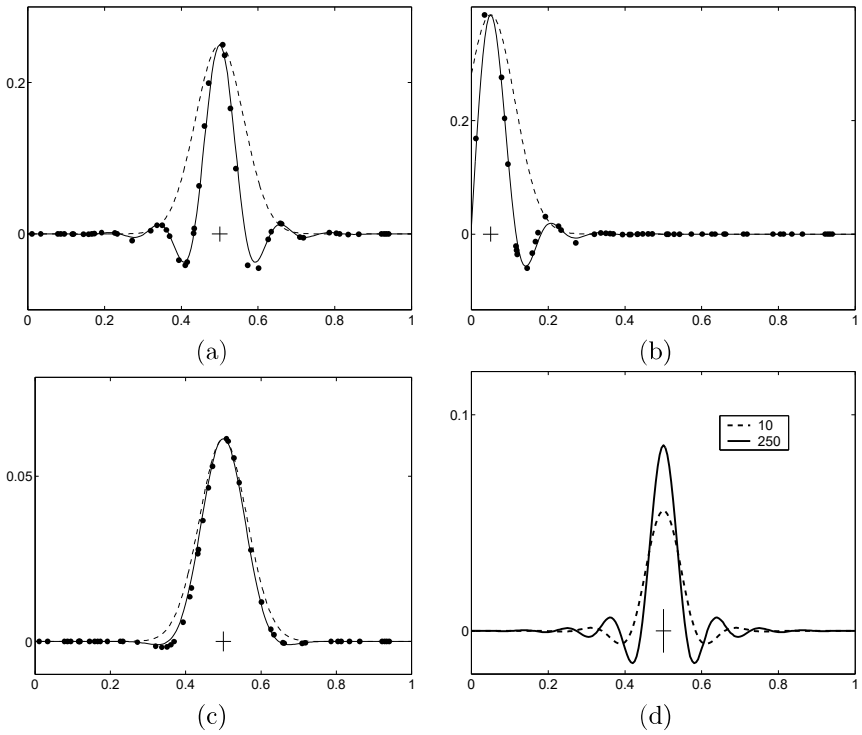



Figure 2.6: Panels (a)-(c) show the weight function $\mathbf{h}(x_*)$ (dots) corresponding to the $n = 50$ training points, the equivalent kernel (solid) and the original squared exponential kernel (dashed). Panel (d) shows the equivalent kernels for two different data densities. See text for further 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 $\lambda$ (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 $n_{\text{grid}}$ 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 $\sigma^2_{\text{grid}} = \sigma_n^2 n_{\text{grid}}/n$; for $n_{\text{grid}} > n$ this means that the effective variance at each of the $n_{\text{grid}}$ 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 $n_{\text{grid}}/n$ independent Gaussian observations with variance $\sigma^2_{\text{grid}}$ at a single $x$-position; this would be equivalent to one Gaussian observation with variance $\sigma_n^2$. 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.

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

fixed mean function

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

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 $K_y = K + \sigma_n^2 I$, 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

polynomial regression

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

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

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 $A = B^{-1} + HK_y^{-1}H^\top$ and $C = K_y^{-1}H^\top A^{-1}HK_y^{-1}$ 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 $A = HK_y^{-1}H^\top$ and $C = K_y^{-1}H^\top A^{-1}HK_y^{-1}$.

## 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

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



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.

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



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 $\text{var}_n(f(\mathbf{x}_*))$ 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 $\text{var}_{n-1}(f(\mathbf{x}_*))$. Show that $\text{var}_n(f(\mathbf{x}_*)) \leq \text{var}_{n-1}(f(\mathbf{x}_*))$, 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 $\text{var}_n(f(\mathbf{x}_*)) = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top (K + \sigma_n^2 I)^{-1} \mathbf{k}_*$. 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].

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

# Chapter 5

# Model Selection and Adaptation of Hyperparameters

In chapters 2 and 3 we have seen how to do regression and classification using a Gaussian process with a given fixed covariance function. However, in many practical applications, it may not be easy to specify all aspects of the covariance function with confidence. While some properties such as stationarity of the covariance function may be easy to determine from the context, we typically have only rather vague information about other properties, such as the value of free (hyper-) parameters, e.g. length-scales. In chapter 4 several examples of covariance functions were presented, many of which have large numbers of parameters. In addition, the exact form and possible free parameters of the likelihood function may also not be known in advance. Thus in order to turn Gaussian processes into powerful practical tools it is essential to develop methods that address the model selection problem. We interpret the model selection problem rather broadly, to include all aspects of the model including the discrete choice of the functional form for the covariance function as well as values for any hyperparameters.

model selection

In section 5.1 we outline the model selection problem. In the following sections different methodologies are presented: in section 5.2 Bayesian principles are covered, and in section 5.3 cross-validation is discussed, in particular the leave-one-out estimator. In the remaining two sections the different methodologies are applied specifically to learning in GP models, for regression in section 5.4 and classification in section 5.5.

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



## 5.1 The Model Selection Problem

In order for a model to be a practical tool in an application, one needs to make decisions about the details of its specification. Some properties may be easy to specify, while we typically have only vague information available about other aspects. We use the term model selection to cover both discrete choices and the setting of continuous (hyper-) parameters of the covariance functions. In fact, model selection can help both to refine the predictions of the model, and give a valuable interpretation to the user about the properties of the data, e.g. that a non-stationary covariance function may be preferred over a stationary one.

enable interpretation

A multitude of possible families of covariance functions exists, including squared exponential, polynomial, neural network, etc., see section 4.2 for an overview. Each of these families typically have a number of free *hyperparameters* whose values also need to be determined. Choosing a covariance function for a particular application thus comprises both setting of hyperparameters within a family, and comparing across different families. Both of these problems will be treated by the same methods, so there is no need to distinguish between them, and we will use the term "model selection" to cover both meanings. We will refer to the selection of a covariance function and its parameters as *training* of a Gaussian process.[1] In the following paragraphs we give example choices of parameterizations of distance measures for stationary covariance functions.

hyperparameters

training

Covariance functions such as the squared exponential can be parameterized in terms of hyperparameters. For example

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^\top M (\mathbf{x}_p - \mathbf{x}_q)\right) + \sigma_n^2 \delta_{pq}, \tag{5.1}$$

where $\boldsymbol{\theta} = (\{M\}, \sigma_f^2, \sigma_n^2)^\top$ is a vector containing all the hyperparameters,[2] and $\{M\}$ denotes the parameters in the symmetric matrix $M$. Possible choices for the matrix $M$ include

$$M_1 = \ell^{-2} I, \qquad M_2 = \mathrm{diag}(\boldsymbol{\ell})^{-2}, \qquad M_3 = \Lambda\Lambda^\top + \mathrm{diag}(\boldsymbol{\ell})^{-2}, \tag{5.2}$$

where $\boldsymbol{\ell}$ is a vector of positive values, and $\Lambda$ is a $D \times k$ matrix, $k < D$. The properties of functions with these covariance functions depend on the values of the hyperparameters. For many covariance functions it is easy to interpret the meaning of the hyperparameters, which is of great importance when trying to understand your data. For the squared exponential covariance function eq. (5.1) with distance measure $M_2$ from eq. (5.2), the $\ell_1, \ldots, \ell_D$ hyperparameters play the rôle of *characteristic length-scales*; loosely speaking, how far do you need to move (along a particular axis) in input space for the function values to become uncorrelated. Such a covariance function implements automatic relevance determination (ARD) [Neal, 1996], since the inverse of the length-scale determines how relevant an input is: if the length-scale has a very large value, the

characteristic length-scale

automatic relevance determination

[1] This contrasts the use of the word in the SVM literature, where "training" usually refers to finding the support vectors for a fixed kernel.

[2] Sometimes the noise level parameter, $\sigma_n^2$ is not considered a hyperparameter; however it plays an analogous role and is treated in the same way, so we simply consider it a hyperparameter.

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



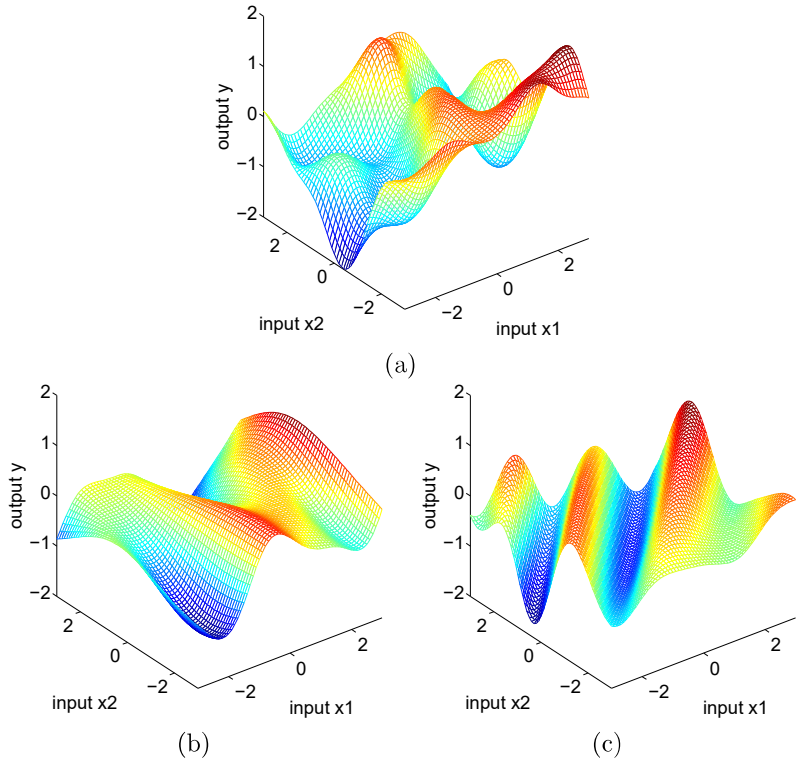



Figure 5.1: Functions with two dimensional input drawn at random from noise free squared exponential covariance function Gaussian processes, corresponding to the three different distance measures in eq. (5.2) respectively. The parameters were: (a) $\ell = 1$, (b) $\boldsymbol{\ell} = (1, 3)^\top$, and (c) $\Lambda = (1, -1)^\top$, $\boldsymbol{\ell} = (6, 6)^\top$. In panel (a) the two inputs are equally important, while in (b) the function varies less rapidly as a function of $x_2$ than $x_1$. In (c) the $\Lambda$ column gives the direction of most rapid variation .

covariance will become almost independent of that input, effectively removing it from the inference. ARD has been used successfully for removing irrelevant input by several authors, e.g. Williams and Rasmussen [1996]. We call the parameterization of $M_3$ in eq. (5.2) the *factor analysis distance* due to the analogy with the (unsupervised) factor analysis model which seeks to explain the data through a low rank plus diagonal decomposition. For high dimensional datasets the $k$ columns of the $\Lambda$ matrix could identify a few directions in the input space with specially high "relevance", and their lengths give the inverse characteristic length-scale for those directions.

factor analysis distance

In Figure 5.1 we show functions drawn at random from squared exponential covariance function Gaussian processes, for different choices of $M$. In panel (a) we get an isotropic behaviour. In panel (b) the characteristic length-scale is different along the two input axes; the function varies rapidly as a function of $x_1$, but less rapidly as a function of $x_2$. In panel (c) the direction of most rapid variation is perpendicular to the direction $(1, 1)$. As this figure illustrates,

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



there is plenty of scope for variation even inside a single family of covariance functions. Our task is, based on a set of training data, to make inferences about the form and parameters of the covariance function, or equivalently, about the relationships in the data.

It should be clear from the above example that model selection is essentially open ended. Even for the squared exponential covariance function, there is a huge variety of possible distance measures. However, this should not be a cause for despair, rather seen as a possibility to learn. It requires, however, a systematic and practical approach to model selection. In a nutshell we need to be able to compare two (or more) methods differing in values of particular parameters, or the shape of the covariance function, or compare a Gaussian process model to any other kind of model. Although there are endless variations in the suggestions for model selection in the literature three general principles cover most: (1) compute the probability of the model given the data, (2) estimate the generalization error and (3) bound the generalization error. We use the term *generalization error* to mean the average error on unseen test examples (from the same distribution as the training cases). Note that the training error is usually a poor proxy for the generalization error, since the model may fit the noise in the training set (over-fit), leading to low training error but poor generalization performance.

In the next section we describe the Bayesian view on model selection, which involves the computation of the probability of the model given the data, based on the marginal likelihood. In section 5.3 we cover cross-validation, which estimates the generalization performance. These two paradigms are applied to Gaussian process models in the remainder of this chapter. The probably approximately correct (PAC) framework is an example of a bound on the generalization error, and is covered in section 7.4.2.

## 5.2 Bayesian Model Selection

In this section we give a short outline description of the main ideas in Bayesian model selection. The discussion will be general, but focusses on issues which will be relevant for the specific treatment of Gaussian process models for regression in section 5.4 and classification in section 5.5.

hierarchical models

It is common to use a hierarchical specification of models. At the lowest level are the parameters, $\mathbf{w}$. For example, the parameters could be the parameters in a linear model, or the weights in a neural network model. At the second level are hyperparameters $\boldsymbol{\theta}$ which control the distribution of the parameters at the bottom level. For example the "weight decay" term in a neural network, or the "ridge" term in ridge regression are hyperparameters. At the top level we may have a (discrete) set of possible model structures, $\mathcal{H}_i$, under consideration.

We will first give a "mechanistic" description of the computations needed for Bayesian inference, and continue with a discussion providing the intuition about what is going on. Inference takes place one level at a time, by applying

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



the rules of probability theory, see e.g. MacKay [1992b] for this framework and MacKay [1992a] for the context of neural networks. At the bottom level, the *posterior* over the parameters is given by Bayes' rule

level 1 inference

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)}, \tag{5.3}$$

where $p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)$ is the *likelihood* and $p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)$ is the parameter *prior*. The prior encodes as a probability distribution our knowledge about the parameters prior to seeing the data. If we have only vague prior information about the parameters, then the prior distribution is chosen to be broad to reflect this. The posterior combines the information from the prior and the data (through the likelihood). The normalizing constant in the denominator of eq. (5.3) $p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)$ is independent of the parameters, and called the *marginal likelihood* (or evidence), and is given by

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)\, d\mathbf{w}. \tag{5.4}$$

At the next level, we analogously express the posterior over the hyperparameters, where the marginal likelihood from the first level plays the rôle of the likelihood

level 2 inference

$$p(\boldsymbol{\theta}|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)}, \tag{5.5}$$

where $p(\boldsymbol{\theta}|\mathcal{H}_i)$ is the *hyper-prior* (the prior for the hyperparameters). The normalizing constant is given by

$$p(\mathbf{y}|X, \mathcal{H}_i) = \int p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)d\boldsymbol{\theta}. \tag{5.6}$$

At the top level, we compute the posterior for the model

level 3 inference

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)}, \tag{5.7}$$

where $p(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)$. We note that the implementation of Bayesian inference calls for the evaluation of several integrals. Depending on the details of the models, these integrals may or may not be analytically tractable and in general one may have to resort to analytical approximations or Markov chain Monte Carlo (MCMC) methods. In practice, especially the evaluation of the integral in eq. (5.6) may be difficult, and as an approximation one may shy away from using the hyperparameter posterior in eq. (5.5), and instead maximize the marginal likelihood in eq. (5.4) w.r.t. the hyperparameters, $\boldsymbol{\theta}$. This approximation is known as type II maximum likelihood (ML-II). Of course, one should be careful with such an optimization step, since it opens up the possibility of overfitting, especially if there are many hyperparameters. The integral in eq. (5.6) can then be approximated using a local expansion around the maximum (the Laplace approximation). This approximation will be good if the posterior for $\boldsymbol{\theta}$ is fairly well peaked, which is more often the case for the

ML-II

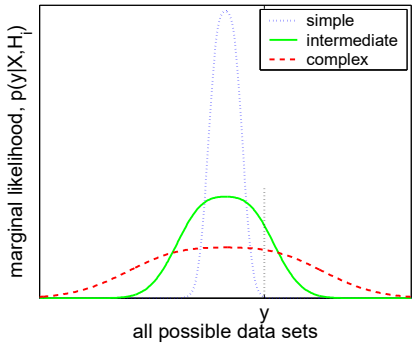



Figure 5.2: The marginal likelihood $p(\mathbf{y}|X, \mathcal{H}_i)$ is the probability of the data, given the model. The number of data points $n$ and the inputs $X$ are fixed, and not shown. The horizontal axis is an idealized representation of all possible vectors of targets $\mathbf{y}$. The marginal likelihood for models of three different complexities are shown. Note, that since the marginal likelihood is a probability distribution, it must normalize to unity. For a particular dataset indicated by $\mathbf{y}$ and a dotted line, the marginal likelihood prefers a model of intermediate complexity over too simple or too complex alternatives.

hyperparameters than for the parameters themselves, see MacKay [1999] for an illuminating discussion. The prior over models $\mathcal{H}_i$ in eq. (5.7) is often taken to be flat, so that a priori we do not favour one model over another. In this case, the probability for the model is proportional to the expression from eq. (5.6).

It is primarily the marginal likelihood from eq. (5.4) involving the integral over the parameter space which distinguishes the Bayesian scheme of inference from other schemes based on optimization. It is a property of the marginal likelihood that it automatically incorporates a trade-off between model fit and model complexity. This is the reason why the marginal likelihood is valuable in solving the model selection problem.

In Figure 5.2 we show a schematic of the behaviour of the marginal likelihood for three different model complexities. Let the number of data points $n$ and the inputs $X$ be fixed; the horizontal axis is an idealized representation of all possible vectors of targets $\mathbf{y}$, and the vertical axis plots the marginal likelihood $p(\mathbf{y}|X, \mathcal{H}_i)$. A simple model can only account for a limited range of possible sets of target values, but since the marginal likelihood is a probability distribution over $\mathbf{y}$ it must normalize to unity, and therefore the data sets which the model *does* account for have a large value of the marginal likelihood. Conversely for a complex model: it is capable of accounting for a wider range of data sets, and consequently the marginal likelihood doesn't attain such large values as for the simple model. For example, the simple model could be a linear model, and the complex model a large neural network. The figure illustrates why the marginal likelihood doesn't simply favour the models that fit the training data the best. This effect is called Occam's razor after William of Occam 1285-1349, whose principle: "plurality should not be assumed without necessity" he used to encourage simplicity in explanations. See also Rasmussen and Ghahramani [2001] for an investigation into Occam's razor in statistical models.

Occam's razor

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



automatic trade-off

Notice that the trade-off between data-fit and model complexity is automatic; there is no need to set a parameter externally to fix the trade-off. Do not confuse the automatic Occam's razor principle with the use of priors in the Bayesian method. Even if the priors are "flat" over complexity, the marginal likelihood will still tend to favour the least complex model able to explain the data. Thus, a model complexity which is well suited to the data can be selected using the marginal likelihood.

In the preceding paragraphs we have thought of the specification of a model as the model structure as well as the parameters of the priors, etc. If it is unclear how to set some of the parameters of the prior, one can treat these as hyperparameters, and do model selection to determine how to set them. At the same time it should be emphasized that the priors correspond to (probabilistic) assumptions about the data. If the priors are grossly at odds with the distribution of the data, inference will still take place under the assumptions encoded by the prior, see the step-function example in section 5.4.3. To avoid this situation, one should be careful not to employ priors which are too narrow, ruling out reasonable explanations of the data.[3]

## 5.3 Cross-validation

cross-validation

In this section we consider how to use methods of cross-validation (CV) for model selection. The basic idea is to split the training set into two disjoint sets, one which is actually used for training, and the other, the *validation* set, which is used to monitor performance. The performance on the validation set is used as a proxy for the generalization error and model selection is carried out using this measure.

In practice a drawback of hold-out method is that only a fraction of the full data set can be used for training, and that if the validation set it small, the performance estimate obtained may have large variance. To minimize these problems, CV is almost always used in the $k$-fold cross-validation setting: the data is split into $k$ disjoint, equally sized subsets; validation is done on a single subset and training is done using the union of the remaining $k-1$ subsets, the entire procedure being repeated $k$ times, each time with a different subset for validation. Thus, a large fraction of the data can be used for training, and all cases appear as validation cases. The price is that $k$ models must be trained instead of one. Typical values for $k$ are in the range 3 to 10.

$k$-fold cross-validation

An extreme case of $k$-fold cross-validation is obtained for $k = n$, the number of training cases, also known as leave-one-out cross-validation (LOO-CV). Often the computational cost of LOO-CV ("training" $n$ models) is prohibitive, but in certain cases, such as Gaussian process regression, there are computational shortcuts.

leave-one-out cross-validation (LOO-CV)

[3] This is known as Cromwell's dictum [Lindley, 1985] after Oliver Cromwell who on August 5th, 1650 wrote to the synod of the Church of Scotland: "I beseech you, in the bowels of Christ, consider it possible that you are mistaken."

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



other loss functions

Cross-validation can be used with any loss function. Although the squared error loss is by far the most common for regression, there is no reason not to allow other loss functions. For probabilistic models such as Gaussian processes it is natural to consider also cross-validation using the negative log probability loss. Craven and Wahba [1979] describe a variant of cross-validation using squared error known as generalized cross-validation which gives different weightings to different datapoints so as to achieve certain invariance properites. See Wahba [1990, sec. 4.3] for further details.

## 5.4 Model Selection for GP Regression

We apply Bayesian inference in section 5.4.1 and cross-validation in section 5.4.2 to Gaussian process regression with Gaussian noise. We conclude in section 5.4.3 with some more detailed examples of how one can use the model selection principles to tailor covariance functions.

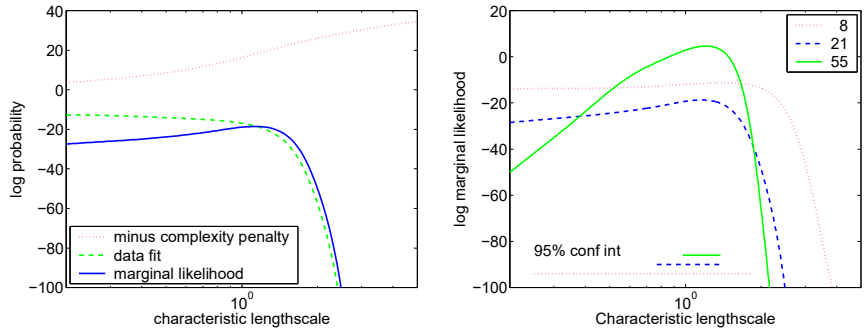
### 5.4.1 Marginal Likelihood

Bayesian principles provide a persuasive and consistent framework for inference. Unfortunately, for most interesting models for machine learning, the required computations (integrals over parameter space) are analytically intractable, and good approximations are not easily derived. Gaussian process regression models with Gaussian noise are a rare exception: integrals over the parameters are analytically tractable and at the same time the models are very flexible. In this section we first apply the general Bayesian inference principles from section 5.2 to the specific Gaussian process model, in the simplified form where hyperparameters are optimized over. We derive the expressions for the marginal likelihood and interpret these.

model parameters

Since a Gaussian process model is a non-parametric model, it may not be immediately obvious what the parameters of the model are. Generally, one may regard the noise-free latent function values at the training inputs $\mathbf{f}$ as the parameters. The more training cases there are, the more parameters. Using the weight-space view, developed in section 2.1, one may equivalently think of the parameters as being the weights of the linear model which uses the basis-functions $\phi$, which can be chosen as the eigenfunctions of the covariance function. Of course, we have seen that this view is inconvenient for nondegenerate covariance functions, since these would then have an infinite number of weights.

We proceed by applying eq. (5.3) and eq. (5.4) for the 1st level of inference—which we find that we have already done back in chapter 2! The predictive distribution from eq. (5.3) is given for the weight-space view in eq. (2.11) and eq. (2.12) and equivalently for the function-space view in eq. (2.22). The *marginal likelihood* (or evidence) from eq. (5.4) was computed in eq. (2.30),




(a) (b)

Figure 5.3: Panel (a) shows a decomposition of the log marginal likelihood into its constituents: data-fit and complexity penalty, as a function of the characteristic length-scale. The training data is drawn from a Gaussian process with SE covariance function and parameters $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$, the same as in Figure 2.5, and we are fitting only the length-scale parameter $\ell$ (the two other parameters have been set in accordance with the generating process). Panel (b) shows the log marginal likelihood as a function of the characteristic length-scale for different sizes of training sets. Also shown, are the 95% confidence intervals for the posterior length-scales.

and we re-state the result here

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) \;=\; -\frac{1}{2}\mathbf{y}^\top K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi, \qquad (5.8)$$

where $K_y = K_f + \sigma_n^2 I$ is the covariance matrix for the noisy targets $\mathbf{y}$ (and $K_f$ is the covariance matrix for the noise-free latent $\mathbf{f}$), and we now explicitly write the marginal likelihood conditioned on the hyperparameters (the parameters of the covariance function) $\boldsymbol{\theta}$. From this perspective it becomes clear why we call eq. (5.8) the log *marginal likelihood*, since it is obtained through marginalization over the latent function. Otherwise, if one thinks entirely in terms of the function-space view, the term "marginal" may appear a bit mysterious, and similarly the "hyper" from the $\boldsymbol{\theta}$ parameters of the covariance function.[4]

marginal likelihood

The three terms of the marginal likelihood in eq. (5.8) have readily interpretable rôles: the only term involving the observed targets is the data-fit $-\mathbf{y}^\top K_y^{-1}\mathbf{y}/2$; $\log|K_y|/2$ is the complexity penalty depending only on the covariance function and the inputs and $n\log(2\pi)/2$ is a normalization constant. In Figure 5.3(a) we illustrate this breakdown of the log marginal likelihood. The data-fit decreases monotonically with the length-scale, since the model becomes less and less flexible. The negative complexity penalty increases with the length-scale, because the model gets less complex with growing length-scale. The marginal likelihood itself peaks at a value close to 1. For length-scales somewhat longer than 1, the marginal likelihood decreases rapidly (note the

interpretation

[4] Another reason that we like to stick to the term "marginal likelihood" is that it is the likelihood of a non-parametric model, i.e. a model which requires access to all the training data when making predictions; this contrasts the situation for a parametric model, which "absorbs" the information from the training data into its (posterior) parameter (distribution). This difference makes the two "likelihoods" behave quite differently as a function of $\boldsymbol{\theta}$.

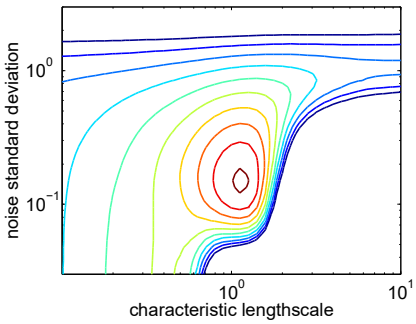



Figure 5.4: Contour plot showing the log marginal likelihood as a function of the characteristic length-scale and the noise level, for the same data as in Figure 2.5 and Figure 5.3. The signal variance hyperparameter was set to $\sigma_f^2 = 1$. The optimum is close to the parameters used when generating the data. Note, the two ridges, one for small noise and length-scale $\ell = 0.4$ and another for long length-scale and noise $\sigma_n^2 = 1$. The contour lines spaced 2 units apart in log probability density.

log scale!), due to the poor ability of the model to explain the data, compare to Figure 2.5(c). For smaller length-scales the marginal likelihood decreases somewhat more slowly, corresponding to models that do accommodate the data, but waste predictive mass at regions far away from the underlying function, compare to Figure 2.5(b).

In Figure 5.3(b) the dependence of the log marginal likelihood on the characteristic length-scale is shown for different numbers of training cases. Generally, the more data, the more peaked the marginal likelihood. For very small numbers of training data points the slope of the log marginal likelihood is very shallow as when only a little data has been observed, both very short and intermediate values of the length-scale are consistent with the data. With more data, the complexity term gets more severe, and discourages too short length-scales.

marginal likelihood gradient

To set the hyperparameters by maximizing the marginal likelihood, we seek the partial derivatives of the marginal likelihood w.r.t. the hyperparameters. Using eq. (5.8) and eq. (A.14-A.15) we obtain

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|X,\boldsymbol{\theta}) &= \frac{1}{2}\mathbf{y}^\top K^{-1}\frac{\partial K}{\partial \theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\operatorname{tr}\left(K^{-1}\frac{\partial K}{\partial \theta_j}\right) \\ &= \frac{1}{2}\operatorname{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^\top - K^{-1})\frac{\partial K}{\partial \theta_j}\right) \quad \text{where} \quad \boldsymbol{\alpha} = K^{-1}\mathbf{y}. \end{aligned} \tag{5.9}$$

The complexity of computing the marginal likelihood in eq. (5.8) is dominated by the need to invert the $K$ matrix (the log determinant of $K$ is easily computed as a by-product of the inverse). Standard methods for matrix inversion of positive definite symmetric matrices require time $\mathcal{O}(n^3)$ for inversion of an $n$ by $n$ matrix. Once $K^{-1}$ is known, the computation of the derivatives in eq. (5.9) requires only time $\mathcal{O}(n^2)$ per hyperparameter.[5] Thus, the computational over-

[5] Note that matrix-by-matrix products in eq. (5.9) should not be computed directly: in the first term, do the vector-by-matrix multiplications first; in the trace term, compute only the diagonal terms of the product.

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



head of computing derivatives is small, so using a gradient based optimizer is advantageous.

Estimation of $\boldsymbol{\theta}$ by optimzation of the marginal likelihood has a long history in spatial statistics, see e.g. Mardia and Marshall [1984]. As $n$ increases, one would hope that the data becomes increasingly informative about $\boldsymbol{\theta}$. However, it is necessary to contrast what Stein [1999, sec. 3.3] calls fixed-domain asymptotics (where one gets increasingly dense observations within some region) with increasing-domain asymptotics (where the size of the observation region grows with $n$). Increasing-domain asymptotics are a natural choice in a time-series context but fixed-domain asymptotics seem more natural in spatial (and machine learning) settings. For further discussion see Stein [1999, sec. 6.4].

Figure 5.4 shows an example of the log marginal likelihood as a function of the characteristic length-scale and the noise standard deviation hyperparameters for the squared exponential covariance function, see eq. (5.1). The signal variance $\sigma_f^2$ was set to 1.0. The marginal likelihood has a clear maximum around the hyperparameter values which were used in the Gaussian process from which the data was generated. Note that for long length-scales and a noise level of $\sigma_n^2 = 1$, the marginal likelihood becomes almost independent of the length-scale; this is caused by the model explaining everything as noise, and no longer needing the signal covariance. Similarly, for small noise and a length-scale of $\ell = 0.4$, the marginal likelihood becomes almost independent of the noise level; this is caused by the ability of the model to exactly interpolate the data at this short length-scale. We note that although the model in this hyperparameter region explains all the data-points exactly, this model is still disfavoured by the marginal likelihood, see Figure 5.2.

multiple local maxima

There is no guarantee that the marginal likelihood does not suffer from multiple local optima. Practical experience with simple covariance functions seem to indicate that local maxima are not a devastating problem, but certainly they do exist. In fact, every local maximum corresponds to a particular interpretation of the data. In Figure 5.5 an example with two local optima is shown, together with the corresponding (noise free) predictions of the model at each of the two local optima. One optimum corresponds to a relatively complicated model with low noise, whereas the other corresponds to a much simpler model with more noise. With only 7 data points, it is not possible for the model to confidently reject either of the two possibilities. The numerical value of the marginal likelihood for the more complex model is about 60% higher than for the simple model. According to the Bayesian formalism, one ought to weight predictions from alternative explanations according to their posterior probabilities. In practice, with data sets of much larger sizes, one often finds that one local optimum is orders of magnitude more probable than other local optima, so averaging together alternative explanations may not be necessary. However, care should be taken that one doesn't end up in a bad local optimum.

multi-task learning

Above we have described how to adapt the parameters of the covariance function given one dataset. However, it may happen that we are given several datasets all of which are assumed to share the same hyperparameters; this is known as *multi-task learning*, see e.g. Caruana [1997]. In this case one can

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



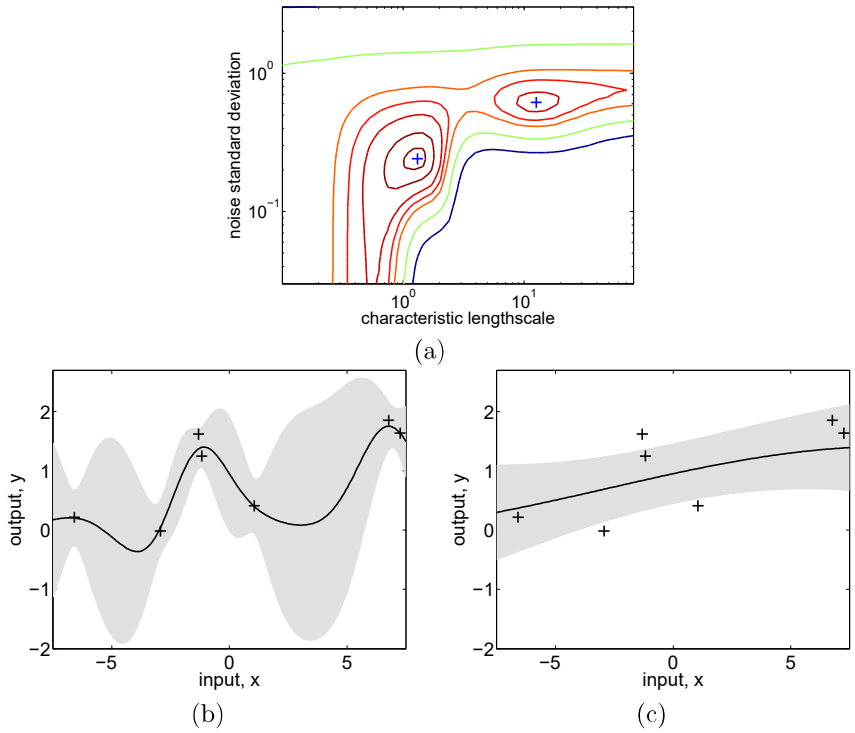



Figure 5.5: Panel (a) shows the marginal likelihood as a function of the hyperparameters $\ell$ (length-scale) and $\sigma_n^2$ (noise standard deviation), where $\sigma_f^2 = 1$ (signal standard deviation) for a data set of 7 observations (seen in panels (b) and (c)). There are two local optima, indicated with '+': the global optimum has low noise and a short length-scale; the local optimum has a high noise and a long length scale. In (b) and (c) the inferred underlying functions (and 95% confidence intervals) are shown for each of the two solutions. In fact, the data points were generated by a Gaussian process with $(\ell, \sigma_f^2, \sigma_n^2) = (1, 1, 0.1)$ in eq. (5.1).

simply sum the log marginal likelihoods of the individual problems and optimize this sum w.r.t. the hyperparameters [Minka and Picard, 1999].

### 5.4.2 Cross-validation

negative log validation density loss

The predictive log probability when leaving out training case $i$ is

$$\log p(y_i|X, \mathbf{y}_{-i}, \boldsymbol{\theta}) = -\frac{1}{2}\log\sigma_i^2 - \frac{(y_i - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2}\log 2\pi, \tag{5.10}$$

where the notation $\mathbf{y}_{-i}$ means all targets *except* number $i$, and $\mu_i$ and $\sigma_i^2$ are computed according to eq. (2.23) and (2.24) respectively, in which the training sets are taken to be $(X_{-i}, \mathbf{y}_{-i})$. Accordingly, the LOO log predictive probability is

$$L_{\mathrm{LOO}}(X, \mathbf{y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(y_i|X, \mathbf{y}_{-i}, \boldsymbol{\theta}), \tag{5.11}$$

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



see [Geisser and Eddy, 1979] for a discussion of this and related approaches. $L_{\mathrm{LOO}}$ in eq. (5.11) is sometimes called the log *pseudo*-likelihood. Notice, that in each of the $n$ LOO-CV rotations, inference in the Gaussian process model (with fixed hyperparameters) essentially consists of computing the inverse covariance matrix, to allow predictive mean and variance in eq. (2.23) and (2.24) to be evaluated (i.e. there is no parameter-fitting, such as there would be in a parametric model). The key insight is that when repeatedly applying the prediction eq. (2.23) and (2.24), the expressions are almost identical: we need the inverses of covariance matrices with a single column and row removed in turn. This can be computed efficiently from the inverse of the complete covariance matrix using inversion by partitioning, see eq. (A.11-A.12). A similar insight has also been used for spline models, see e.g. Wahba [1990, sec. 4.2]. The approach was used for hyperparameter selection in Gaussian process models in Sundararajan and Keerthi [2001]. The expressions for the LOO-CV predictive mean and variance are

pseudo-likelihood

$$\mu_i \;=\; y_i - [K^{-1}\mathbf{y}]_i/[K^{-1}]_{ii}, \qquad \text{and} \qquad \sigma_i^2 \;=\; 1/[K^{-1}]_{ii}, \tag{5.12}$$

where careful inspection reveals that the mean $\mu_i$ is in fact independent of $y_i$ as indeed it should be. The computational expense of computing these quantities is $\mathcal{O}(n^3)$ once for computing the inverse of $K$ plus $\mathcal{O}(n^2)$ for the entire LOO-CV procedure (when $K^{-1}$ is known). Thus, the computational overhead for the LOO-CV quantities is negligible. Plugging these expressions into eq. (5.10) and (5.11) produces a performance estimator which we can optimize w.r.t. hyperparameters to do model selection. In particular, we can compute the partial derivatives of $L_{\mathrm{LOO}}$ w.r.t. the hyperparameters (using eq. (A.14)) and use conjugate gradient optimization. To this end, we need the partial derivatives of the LOO-CV predictive mean and variances from eq. (5.12) w.r.t. the hyperparameters

$$\frac{\partial \mu_i}{\partial \theta_j} \;=\; \frac{[Z_j\boldsymbol{\alpha}]_i}{[K^{-1}]_{ii}} - \frac{\boldsymbol{\alpha}_i[Z_jK^{-1}]_{ii}}{[K^{-1}]_{ii}^2}, \qquad \frac{\partial \sigma_i^2}{\partial \theta_j} \;=\; \frac{[Z_jK^{-1}]_{ii}}{[K^{-1}]_{ii}^2}, \tag{5.13}$$

where $\boldsymbol{\alpha} = K^{-1}\mathbf{y}$ and $Z_j = K^{-1}\frac{\partial K}{\partial \theta_j}$. The partial derivatives of eq. (5.11) are obtained by using the chain-rule and eq. (5.13) to give

$$\begin{aligned} \frac{\partial L_{\mathrm{LOO}}}{\partial \theta_j} \;&=\; \sum_{i=1}^{n} \frac{\partial \log p(y_i|X,\mathbf{y}_{-i},\boldsymbol{\theta})}{\partial \mu_i}\frac{\partial \mu_i}{\partial \theta_j} + \frac{\partial \log p(y_i|X,\mathbf{y}_{-i},\boldsymbol{\theta})}{\partial \sigma_i^2}\frac{\partial \sigma_i^2}{\partial \theta_j} \\ &=\; \sum_{i=1}^{n} \Big(\alpha_i[Z_j\boldsymbol{\alpha}]_i - \frac{1}{2}\Big(1+\frac{\alpha_i^2}{[K^{-1}]_{ii}}\Big)[Z_jK^{-1}]_{ii}\Big)/[K^{-1}]_{ii}. \end{aligned} \tag{5.14}$$

The computational complexity is $\mathcal{O}(n^3)$ for computing the inverse of $K$, and $\mathcal{O}(n^3)$ *per hyperparameter*[6] for the derivative eq. (5.14). Thus, the computational burden of the derivatives is greater for the LOO-CV method than for the method based on marginal likelihood, eq. (5.9).

[6] Computation of the matrix-by-matrix product $K^{-1}\frac{\partial K}{\partial \theta_j}$ for each hyperparameter is unavoidable.

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



LOO-CV with squared error loss

In eq. (5.10) we have used the log of the validation density as a cross-validation measure of fit (or equivalently, the negative log validation density as a loss function). One could also envisage using other loss functions, such as the commonly used squared error. However, this loss function is only a function of the predicted mean and ignores the validation set variances. Further, since the mean prediction eq. (2.23) is independent of the scale of the covariances (i.e. you can multiply the covariance of the signal *and* noise by an arbitrary positive constant without changing the mean predictions), one degree of freedom is left undetermined[7] by a LOO-CV procedure based on squared error loss (or any other loss function which depends only on the mean predictions). But, of course, the full predictive distribution does depend on the scale of the covariance function. Also, computation of the derivatives based on the squared error loss has similar computational complexity as the negative log validation density loss. In conclusion, it seems unattractive to use LOO-CV based on squared error loss for hyperparameter selection.

Comparing the pseudo-likelihood for the LOO-CV methodology with the marginal likelihood from the previous section, it is interesting to ask under which circumstances each method might be preferable. Their computational demands are roughly identical. This issue has not been studied much empirically. However, it is interesting to note that the marginal likelihood tells us the probability of the observations *given the assumptions of the model*. This contrasts with the frequentist LOO-CV value, which gives an estimate for the (log) predictive probability, whether or not the assumptions of the model may be fulfilled. Thus Wahba [1990, sec. 4.8] has argued that CV procedures should be more robust against model mis-specification.

### 5.4.3 Examples and Discussion

In the following we give three examples of model selection for regression models. We first describe a 1-d modelling task which illustrates how special covariance functions can be designed to achieve various useful effects, and can be evaluated using the marginal likelihood. Secondly, we make a short reference to the model selection carried out for the robot arm problem discussed in chapter 2 and again in chapter 8. Finally, we discuss an example where we deliberately choose a covariance function that is not well-suited for the problem; this is the so-called mis-specified model scenario.

#### Mauna Loa Atmospheric Carbon Dioxide

We will use a modelling problem concerning the concentration of $CO_2$ in the atmosphere to illustrate how the marginal likelihood can be used to set multiple hyperparameters in hierarchical Gaussian process models. A complex covariance function is derived by combining several different kinds of simple covariance functions, and the resulting model provides an excellent fit to the data as well

[7] In the special case where we know either the signal or the noise variance there is no indeterminancy.

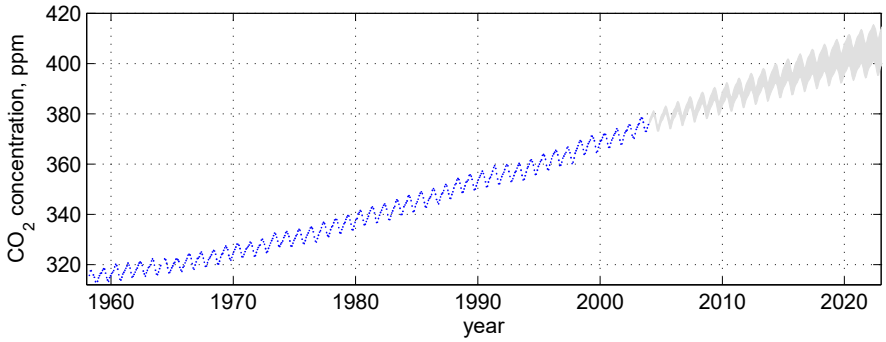



Figure 5.6: The 545 observations of monthly averages of the atmospheric concentration of $CO_2$ made between 1958 and the end of 2003, together with 95% predictive confidence region for a Gaussian process regression model, 20 years into the future. Rising trend and seasonal variations are clearly visible. Note also that the confidence interval gets wider the further the predictions are extrapolated.

as insights into its properties by interpretation of the adapted hyperparameters. Although the data is one-dimensional, and therefore easy to visualize, a total of 11 hyperparameters are used, which in practice rules out the use of cross-validation for setting parameters, except for the gradient-based LOO-CV procedure from the previous section.

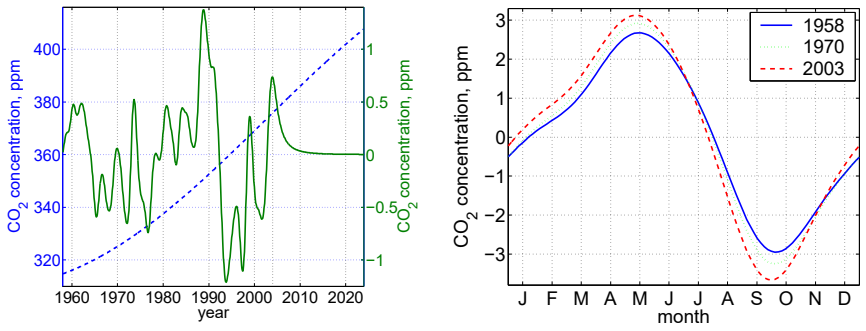
The data [Keeling and Whorf, 2004] consists of monthly average atmospheric $CO_2$ concentrations (in parts per million by volume (ppmv)) derived from in situ air samples collected at the Mauna Loa Observatory, Hawaii, between 1958 and 2003 (with some missing values).[8] The data is shown in Figure 5.6. Our goal is the model the $CO_2$ concentration as a function of time $x$. Several features are immediately apparent: a long term rising trend, a pronounced seasonal variation and some smaller irregularities. In the following we suggest contributions to a combined covariance function which takes care of these individual properties. This is meant primarily to illustrate the power and flexibility of the Gaussian process framework—it is possible that other choices would be more appropriate for this data set.

To model the long term smooth rising trend we use a squared exponential (SE) covariance term, with two hyperparameters controlling the amplitude $\theta_1$ and characteristic length-scale $\theta_2$ smooth trend

$$k_1(x, x') = \theta_1^2 \exp\Big(-\frac{(x - x')^2}{2\theta_2^2}\Big). \tag{5.15}$$

Note that we just use a smooth trend; actually enforcing the trend a priori to be *increasing* is probably not so simple and (hopefully) not desirable. We can use the periodic covariance function from eq. (4.31) with a period of one year to model the seasonal variation. However, it is not clear that the seasonal trend is seasonal component

[8] The data is available from http://cdiac.esd.ornl.gov/ftp/trends/co2/maunaloa.co2.




(a) (b)

Figure 5.7: Panel (a): long term trend, dashed, left hand scale, predicted by the squared exponential contribution; superimposed is the medium term trend, full line, right hand scale, predicted by the rational quadratic contribution; the vertical dash-dotted line indicates the upper limit of the training data. Panel (b) shows the seasonal variation over the year for three different years. The concentration peaks in mid May and has a low in the beginning of October. The seasonal variation is smooth, but not of exactly sinusoidal shape. The peak-to-peak amplitude increases from about 5.5 ppm in 1958 to about 7 ppm in 2003, but the shape does not change very much. The characteristic decay length of the periodic component is inferred to be 90 years, so the seasonal trend changes rather slowly, as also suggested by the gradual progression between the three years shown.

exactly periodic, so we modify eq. (4.31) by taking the product with a squared exponential component (using the product construction from section 4.2.4), to allow a *decay* away from exact periodicity

$$k_2(x,x') \;=\; \theta_3^2 \exp\Big(-\frac{(x-x')^2}{2\theta_4^2} - \frac{2\sin^2(\pi(x-x'))}{\theta_5^2}\Big), \tag{5.16}$$

where $\theta_3$ gives the magnitude, $\theta_4$ the *decay-time* for the periodic component, and $\theta_5$ the smoothness of the periodic component; the period has been fixed to one (year). The seasonal component in the data is caused primarily by different rates of $CO_2$ uptake for plants depending on the season, and it is probably reasonable to assume that this pattern may itself change slowly over time, partially due to the elevation of the $CO_2$ level itself; if this effect turns out not to be relevant, then it can be effectively removed at the fitting stage by allowing $\theta_4$ to become very large.

medium term irregularities

To model the (small) medium term irregularities a rational quadratic term is used, eq. (4.19)

$$k_3(x,x') \;=\; \theta_6^2\Big(1+\frac{(x-x')^2}{2\theta_8\theta_7^2}\Big)^{-\theta_8}, \tag{5.17}$$

where $\theta_6$ is the magnitude, $\theta_7$ is the typical length-scale and $\theta_8$ is the shape parameter determining diffuseness of the length-scales, see the discussion on page 87. One could also have used a squared exponential form for this component, but it turns out that the rational quadratic works better (gives higher marginal likelihood), probably because it can accommodate several length-scales.

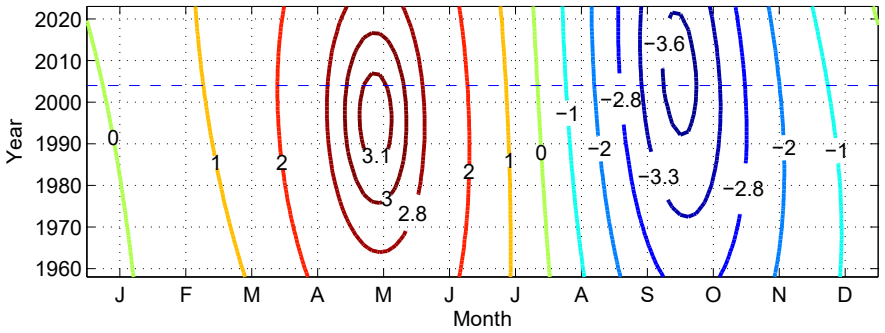



Figure 5.8: The time course of the seasonal effect, plotted in a months vs. year plot (with wrap-around continuity between the edges). The labels on the contours are in ppmv of $CO_2$. The training period extends up to the dashed line. Note the slow development: the height of the May peak may have started to recede, but the low in October may currently (2005) be deepening further. The seasonal effects from three particular years were also plotted in Figure 5.7(b).

noise terms

Finally we specify a noise model as the sum of a squared exponential contribution and an independent component

$$k_4(x_p, x_q) \;=\; \theta_9^2 \exp\Big(-\frac{(x_p - x_q)^2}{2\theta_{10}^2}\Big) + \theta_{11}^2 \delta_{pq}, \tag{5.18}$$

where $\theta_9$ is the magnitude of the correlated noise component, $\theta_{10}$ is its length-scale and $\theta_{11}$ is the magnitude of the independent noise component. Noise in the series could be caused by measurement inaccuracies, and by local short-term weather phenomena, so it is probably reasonable to assume at least a modest amount of correlation in time. Notice that the correlated noise component, the first term of eq. (5.18), has an identical expression to the long term component in eq. (5.15). When optimizing the hyperparameters, we will see that one of these components becomes large with a long length-scale (the long term trend), while the other remains small with a short length-scale (noise). The fact that we have chosen to call one of these components 'signal' and the other one 'noise' is only a question of interpretation. Presumably we are less interested in very short-term effect, and thus call it noise; if on the other hand we were interested in this effect, we would call it signal.

The final covariance function is

$$k(x, x') \;=\; k_1(x, x') + k_2(x, x') + k_3(x, x') + k_4(x, x'), \tag{5.19}$$

with hyperparameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{11})^\top$. We first subtract the empirical mean of the data (341 ppm), and then fit the hyperparameters by optimizing the marginal likelihood using a conjugate gradient optimizer. To avoid bad local minima (e.g. caused by swapping rôles of the rational quadratic and squared exponential terms) a few random restarts are tried, picking the run with the best marginal likelihood, which was $\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -108.5$.

parameter estimation

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



We now examine and interpret the hyperparameters which optimize the marginal likelihood. The long term trend has a magnitude of $\theta_1 = 66$ ppm and a length scale of $\theta_2 = 67$ years. The mean predictions inside the range of the training data and extending for 20 years into the future are depicted in Figure 5.7 (a). In the same plot (with right hand axis) we also show the medium term effects modelled by the rational quadratic component with magnitude $\theta_6 = 0.66$ ppm, typical length $\theta_7 = 1.2$ years and shape $\theta_8 = 0.78$. The very small shape value allows for covariance at many different length-scales, which is also evident in Figure 5.7 (a). Notice that beyond the edge of the training data the mean of this contribution smoothly decays to zero, but of course it still has a contribution to the uncertainty, see Figure 5.6.

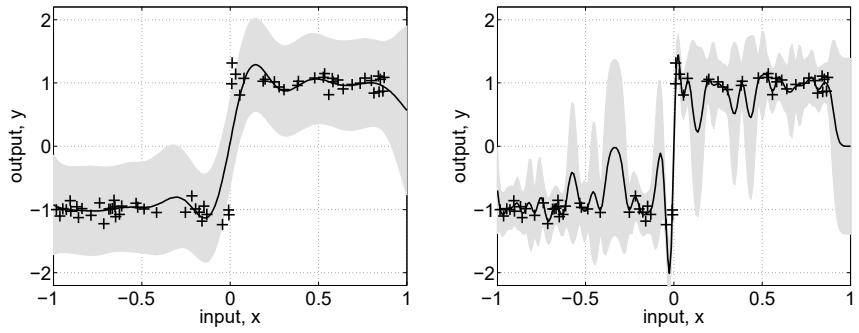
The hyperparameter values for the decaying periodic contribution are: magnitude $\theta_3 = 2.4$ ppm, decay-time $\theta_4 = 90$ years, and the smoothness of the periodic component is $\theta_5 = 1.3$. The quite long decay-time shows that the data have a very close to periodic component in the short term. In Figure 5.7 (b) we show the mean periodic contribution for three years corresponding to the beginning, middle and end of the training data. This component is not exactly sinusoidal, and it changes its shape slowly over time, most notably the amplitude is increasing, see Figure 5.8.

For the noise components, we get the amplitude for the correlated component $\theta_9 = 0.18$ ppm, a length-scale of $\theta_{10} = 1.6$ months and an independent noise magnitude of $\theta_{11} = 0.19$ ppm. Thus, the correlation length for the noise component is indeed inferred to be short, and the total magnitude of the noise is just $\sqrt{\theta_9^2 + \theta_{11}^2} = 0.26$ ppm, indicating that the data can be explained very well by the model. Note also in Figure 5.6 that the model makes relatively confident predictions, the 95% confidence region being 16 ppm wide at a 20 year prediction horizon.

In conclusion, we have seen an example of how non-trivial structure can be inferred by using composite covariance functions, and that the ability to leave hyperparameters to be determined by the data is useful in practice. Of course a serious treatment of such data would probably require modelling of other effects, such as demographic and economic indicators too. Finally, one may want to use a real time-series approach (not just a regression from time to $CO_2$ level as we have done here), to accommodate causality, etc. Nevertheless, the ability of the Gaussian process to avoid simple parametric assumptions and still build in a lot of structure makes it, as we have seen, a very attractive model in many application domains.

### Robot Arm Inverse Dynamics

We have discussed the use of GPR for the SARCOS robot arm inverse dynamics problem in section 2.5. This example is also further studied in section 8.3.7 where a variety of approximation methods are compared, because the size of the training set (44, 484 examples) precludes the use of simple GPR due to its $\mathcal{O}(n^2)$ storage and $\mathcal{O}(n^3)$ time complexity.




(a) (b)

Figure 5.9: Mis-specification example. Fit to 64 datapoints drawn from a step function with Gaussian noise with standard deviation $\sigma_n = 0.1$. The Gaussian process models are using a squared exponential covariance function. Panel (a) shows the mean and 95% confidence interval for the noisy signal in grey, when the hyperparameters are chosen to maximize the marginal likelihood. Panel (b) shows the resulting model when the hyperparameters are chosen using leave-one-out cross-validation (LOO-CV). Note that the marginal likelihood chooses a high noise level and long length-scale, whereas LOO-CV chooses a smaller noise level and shorter length-scale. It is not immediately obvious which fit it worse.

One of the techniques considered in section 8.3.7 is the subset of datapoints (SD) method, where we simply discard some of the data and only make use of $m < n$ training examples. Given a subset of the training data of size $m$ selected at random, we adjusted the hyperparameters by optimizing either the marginal likelihood or $L_{\mathrm{LOO}}$. As ARD was used, this involved adjusting $D + 2 = 23$ hyperparameters. This process was repeated 10 times with different random subsets of the data selected for both $m = 1024$ and $m = 2048$. The results show that the predictive accuracy obtained from the two optimization methods is very similar on both standardized mean squared error (SMSE) and mean standardized log loss (MSLL) criteria, but that the marginal likelihood optimization is much quicker.

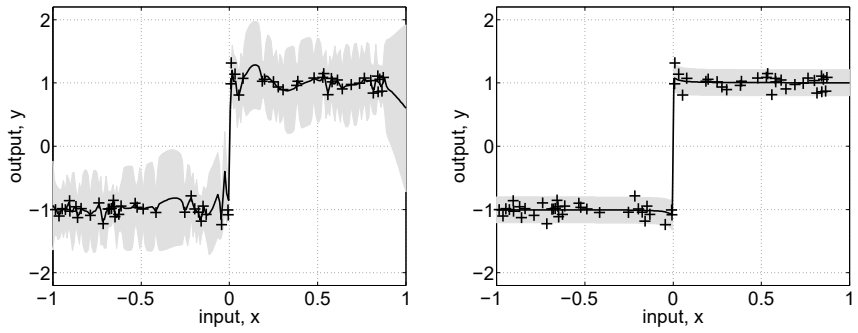
### Step function example illustrating mis-specification

In this section we discuss the *mis-specified* model scenario, where we attempt to learn the hyperparameters for a covariance function which is not very well suited to the data. The mis-specification arises because the data comes from a function which has either zero or very low probability under the GP prior. One could ask why it is interesting to discuss this scenario, since one should surely simply avoid choosing such a model in practice. While this is true in theory, for practical reasons such as the convenience of using standard forms for the covariance function or because vague prior information, one inevitably ends up in a situation which resembles some level of mis-specification.

As an example, we use data from a noisy step function and fit a GP model with a squared exponential covariance function, Figure 5.9. There is mis-specification because it would be very unlikely that samples drawn from a GP

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






(a) (b)

Figure 5.10: Same data as in Figure 5.9. Panel (a) shows the result of using a covariance function which is the sum of two squared-exponential terms. Although this is still a stationary covariance function, it gives rise to a higher marginal likelihood than for the squared-exponential covariance function in Figure 5.9(a), and probably also a better fit. In panel (b) the neural network covariance function eq. (4.29) was used, providing a much larger marginal likelihood and a very good fit.

with the stationary SE covariance function would look like a step function. For short length-scales samples can vary quite quickly, but they would tend to vary rapidly all over, not just near the step. Conversely a stationary SE covariance function with a long length-scale could model the flat parts of the step function but not the rapid transition. Note that Gibbs' covariance function eq. (4.32) would be one way to achieve the desired effect. It is interesting to note the differences between the model optimized with marginal likelihood in Figure 5.9(a), and one optimized with LOO-CV in panel (b) of the same figure. See exercise 5.6.2 for more on how these two criteria weight the influence of the prior.

For comparison, we show the predictive distribution for two other covariance functions in Figure 5.10. In panel (a) a sum of two squared exponential terms were used in the covariance. Notice that this covariance function is still stationary, but it is more flexible than a single squared exponential, since it has two magnitude and two length-scale parameters. The predictive distribution looks a little bit better, and the value of the log marginal likelihood improves from $-37.7$ in Figure 5.9(a) to $-26.1$ in Figure 5.10(a). We also tried the neural network covariance function from eq. (4.29), which is ideally suited to this case, since it allows saturation at different values in the positive and negative directions of $x$. As shown in Figure 5.10(b) the predictions are also near perfect, and the log marginal likelihood is much larger at 50.2.

## 5.5 Model Selection for GP Classification

In this section we compute the derivatives of the approximate marginal likelihood for the Laplace and EP methods for binary classification which are needed for training. We also give the detailed algorithms for these, and briefly discuss the possible use of cross-validation and other methods for training binary GP

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



classifiers.

### 5.5.1 Derivatives of the Marginal Likelihood for Laplace's Approximation *

Recall from section 3.4.4 that the approximate log marginal likelihood was given in eq. (3.32) as

$$\log q(\mathbf{y}|X,\boldsymbol{\theta}) \;=\; -\frac{1}{2}\hat{\mathbf{f}}^\top K^{-1}\hat{\mathbf{f}} + \log p(\mathbf{y}|\hat{\mathbf{f}}) - \frac{1}{2}\log|B|, \tag{5.20}$$

where $B = I + W^{\frac{1}{2}}KW^{\frac{1}{2}}$ and $\hat{\mathbf{f}}$ is the maximum of the posterior eq. (3.12) found by Newton's method in Algorithm 3.1, and $W$ is the diagonal matrix $W = -\nabla\nabla\log p(\mathbf{y}|\hat{\mathbf{f}})$. We can now optimize the approximate marginal likelihood $q(\mathbf{y}|X,\boldsymbol{\theta})$ w.r.t. the hyperparameters, $\boldsymbol{\theta}$. To this end we seek the partial derivatives of $\partial q(\mathbf{y}|X,\boldsymbol{\theta})/\partial\theta_j$. The covariance matrix $K$ is a function of the hyperparameters, but $\hat{\mathbf{f}}$ and therefore $W$ are also implicitly functions of $\boldsymbol{\theta}$, since when $\boldsymbol{\theta}$ changes, the optimum of the posterior $\hat{\mathbf{f}}$ also changes. Thus

$$\frac{\partial\log q(\mathbf{y}|X,\boldsymbol{\theta})}{\partial\theta_j} \;=\; \frac{\partial\log q(\mathbf{y}|X,\boldsymbol{\theta})}{\partial\theta_j}\bigg|_{\text{explicit}} + \sum_{i=1}^{n}\frac{\partial\log q(\mathbf{y}|X,\boldsymbol{\theta})}{\partial\hat{f}_i}\frac{\partial\hat{f}_i}{\partial\theta_j}, \tag{5.21}$$

by the chain rule. Using eq. (A.14) and eq. (A.15) the explicit term is given by

$$\frac{\partial\log q(\mathbf{y}|X,\boldsymbol{\theta})}{\partial\theta_j}\bigg|_{\text{explicit}} \;=\; \frac{1}{2}\hat{\mathbf{f}}^\top K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\hat{\mathbf{f}} - \frac{1}{2}\operatorname{tr}\Big((W^{-1}+K)^{-1}\frac{\partial K}{\partial\theta_j}\Big). \tag{5.22}$$

When evaluating the remaining term from eq. (5.21), we utilize the fact that $\hat{\mathbf{f}}$ is the maximum of the posterior, so that $\partial\Psi(\mathbf{f})/\partial\mathbf{f} = \mathbf{0}$ at $\mathbf{f} = \hat{\mathbf{f}}$, where the (un-normalized) log posterior $\Psi(\mathbf{f})$ is defined in eq. (3.12); thus the implicit derivatives of the two first terms of eq. (5.20) vanish, leaving only

$$\begin{aligned}\frac{\partial\log q(\mathbf{y}|X,\boldsymbol{\theta})}{\partial\hat{f}_i} \;&=\; -\frac{1}{2}\frac{\partial\log|B|}{\partial\hat{f}_i} \;=\; -\frac{1}{2}\operatorname{tr}\Big((K^{-1}+W)^{-1}\frac{\partial W}{\partial\hat{f}_i}\Big)\\ &=\; -\frac{1}{2}\big[(K^{-1}+W)^{-1}\big]_{ii}\frac{\partial^3}{\partial\hat{f}_i^3}\log p(\mathbf{y}|\hat{\mathbf{f}}).\end{aligned} \tag{5.23}$$

In order to evaluate the derivative $\partial\hat{\mathbf{f}}/\partial\theta_j$, we differentiate the self-consistent eq. (3.17) $\hat{\mathbf{f}} = K\nabla\log p(\mathbf{y}|\hat{\mathbf{f}})$ to obtain

$$\frac{\partial\hat{\mathbf{f}}}{\partial\theta_j} \;=\; \frac{\partial K}{\partial\theta_j}\nabla\log p(\mathbf{y}|\hat{\mathbf{f}}) + K\frac{\partial\nabla\log p(\mathbf{y}|\hat{\mathbf{f}})}{\partial\hat{\mathbf{f}}}\frac{\partial\hat{\mathbf{f}}}{\partial\theta_j} \;=\; (I+KW)^{-1}\frac{\partial K}{\partial\theta_j}\nabla\log p(\mathbf{y}|\hat{\mathbf{f}}), \tag{5.24}$$

where we have used the chain rule $\partial/\partial\theta_j = \partial\hat{\mathbf{f}}/\partial\theta_j\cdot\partial/\partial\hat{\mathbf{f}}$ and the identity $\partial\nabla\log p(\mathbf{y}|\hat{\mathbf{f}})/\partial\hat{\mathbf{f}} = -W$. The desired derivatives are obtained by plugging eq. (5.22-5.24) into eq. (5.21).

**input**: $X$ (inputs), $\mathbf{y}$ ($\pm 1$ targets), $\boldsymbol{\theta}$ (hypers), $p(\mathbf{y}|\mathbf{f})$ (likelihood function)
2: compute $K$ — compute covariance matrix from $X$ and $\boldsymbol{\theta}$
$(\mathbf{f},\ \mathbf{a}) := \text{mode}\big(K, \mathbf{y}, p(\mathbf{y}|\mathbf{f})\big)$ — locate posterior mode using Algorithm 3.1
4: $W := -\nabla\nabla \log p(\mathbf{y}|\mathbf{f})$
$L := \text{cholesky}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})$ — solve $LL^\top = B = I + W^{\frac{1}{2}} K W^{\frac{1}{2}}$
6: $\log Z := -\frac{1}{2}\mathbf{a}^\top \mathbf{f} + \log p(\mathbf{y}|\mathbf{f}) - \sum \log(\text{diag}(L))$ — eq. (5.20)
$R := W^{\frac{1}{2}} L^\top \backslash (L \backslash W^{\frac{1}{2}})$ — $R = W^{\frac{1}{2}}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})^{-1} W^{\frac{1}{2}}$
8: $C := L \backslash (W^{\frac{1}{2}} K)$
$\mathbf{s}_2 := -\frac{1}{2}\,\text{diag}\big(\text{diag}(K) - \text{diag}(C^\top C)\big)\nabla^3 \log p(\mathbf{y}|\mathbf{f})$ } eq. (5.23)
10: **for** $j := 1 \ldots \dim(\boldsymbol{\theta})$ **do**
$C := \partial K / \partial \theta_j$ — compute derivative matrix from $X$ and $\boldsymbol{\theta}$
12: $s_1 := \frac{1}{2}\mathbf{a}^\top C \mathbf{a} - \frac{1}{2}\,\text{tr}(RC)$ — eq. (5.22)
$\mathbf{b} := C \nabla \log p(\mathbf{y}|\mathbf{f})$
14: $\mathbf{s}_3 := \mathbf{b} - KR\mathbf{b}$ } eq. (5.24)
$\nabla_j \log Z := s_1 + \mathbf{s}_2^\top \mathbf{s}_3$ — eq. (5.21)
16: **end for**
**return**: $\log Z$ (log marginal likelihood), $\nabla \log Z$ (partial derivatives)

Algorithm 5.1: Compute the approximate log marginal likelihood and its derivatives w.r.t. the hyperparameters for binary Laplace GPC for use by an optimization routine, such as conjugate gradient optimization. In line 3 Algorithm 3.1 on page 46 is called to locate the posterior mode. In line 12 only the diagonal elements of the matrix product should be computed. In line 15 the notation $\nabla_j$ means the partial derivative w.r.t. the $j$'th hyperparameter. An actual implementation may also return the value of $\mathbf{f}$ to be used as an initial guess for the subsequent call (as an alternative the zero initialization in line 2 of Algorithm 3.1).

### Details of the Implementation

The implementation of the log marginal likelihood and its partial derivatives w.r.t. the hyperparameters is shown in Algorithm 5.1. It is advantageous to re-write the equations from the previous section in terms of well-conditioned symmetric positive definite matrices, whose solutions can be obtained by Cholesky factorization, combining numerical stability with computational speed.

In detail, the matrix of central importance turns out to be

$$R \;=\; (W^{-1} + K)^{-1} \;=\; W^{\frac{1}{2}}(I + W^{\frac{1}{2}} K W^{\frac{1}{2}})^{-1} W^{\frac{1}{2}}, \qquad (5.25)$$

where the right hand side is suitable for numerical evaluation as in line 7 of Algorithm 5.1, reusing the Cholesky factor $L$ from the Newton scheme above. Remember that $W$ is diagonal so eq. (5.25) does not require any real matrix-by-matrix products. Rewriting eq. (5.22-5.23) is straightforward, and for eq. (5.24) we apply the matrix inversion lemma (eq. (A.9)) to $(I + KW)^{-1}$ to obtain $I - KR$, which is used in the implementation.

The computational complexity is dominated by the Cholesky factorization in line 5 which takes $n^3/6$ operations per iteration of the Newton scheme. In addition the computation of $R$ in line 7 is also $\mathcal{O}(n^3)$, all other computations being at most $\mathcal{O}(n^2)$ per hyperparameter.

**input**: $X$ (inputs), $\mathbf{y}$ ($\pm 1$ targets), $\boldsymbol{\theta}$ (hyperparameters)
2: compute $K$ — compute covariance matrix from $X$ and $\boldsymbol{\theta}$
$(\tilde{\boldsymbol{\nu}},\ \tilde{\boldsymbol{\tau}},\ \log Z_{\mathrm{EP}}) := \mathrm{EP}\,(K, \mathbf{y})$ — run the EP Algorithm 3.5
4: $L := \mathrm{cholesky}(I + \tilde{S}^{\frac{1}{2}} K \tilde{S}^{\frac{1}{2}})$ — solve $LL^\top = B = I + \tilde{S}^{\frac{1}{2}} K \tilde{S}^{\frac{1}{2}}$
$\mathbf{b} := \tilde{\boldsymbol{\nu}} - \tilde{S}^{\frac{1}{2}} L \backslash (L^\top \backslash \tilde{S}^{\frac{1}{2}} K \tilde{\boldsymbol{\nu}})$ — $\mathbf{b}$ from under eq. (5.27)
6: $R := \mathbf{b}\mathbf{b}^\top - \tilde{S}^{\frac{1}{2}} L^\top \backslash (L \backslash \tilde{S}^{\frac{1}{2}})$ — $R = \mathbf{b}\mathbf{b}^\top - \tilde{S}^{\frac{1}{2}} B^{-1} \tilde{S}^{\frac{1}{2}}$
**for** $j := 1 \ldots \dim(\boldsymbol{\theta})$ **do**
8: $C := \partial K / \partial \theta_j$ — compute derivative matrix from $X$ and $\boldsymbol{\theta}$
$\nabla_j \log Z_{\mathrm{EP}} := \frac{1}{2} \mathrm{tr}(RC)$ — eq. (5.27)
10: **end for**
**return**: $\log Z_{\mathrm{EP}}$ (log marginal likelihood), $\nabla \log Z_{\mathrm{EP}}$ (partial derivatives)

Algorithm 5.2: Compute the log marginal likelihood and its derivatives w.r.t. the hyperparameters for EP binary GP classification for use by an optimization routine, such as conjugate gradient optimization. $\tilde{S}$ is a diagonal precision matrix with entries $\tilde{S}_{ii} = \tilde{\tau}_i$. In line 3 Algorithm 3.5 on page 58 is called to compute parameters of the EP approximation. In line 9 only the diagonal of the matrix product should be computed and the notation $\nabla_j$ means the partial derivative w.r.t. the $j$'th hyperparameter. The computational complexity is dominated by the Cholesky factorization in line 4 and the solution in line 6, both of which are $\mathcal{O}(n^3)$.

### 5.5.2 Derivatives of the Marginal Likelihood for EP *

Optimization of the EP approximation to the marginal likelihood w.r.t. the hyperparameters of the covariance function requires evaluation of the partial derivatives from eq. (3.65). Luckily, it turns out that implicit terms in the derivatives caused by the solution of EP being a function of the hyperparameters is exactly zero. We will not present the proof here, see Seeger [2005]. Consequently, we only have to take account of the explicit dependencies

$$\begin{aligned} \frac{\partial \log Z_{\mathrm{EP}}}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\Big( -\frac{1}{2}\tilde{\boldsymbol{\mu}}^\top (K + \tilde{\Sigma})^{-1} \tilde{\boldsymbol{\mu}} - \frac{1}{2} \log |K + \tilde{\Sigma}| \Big) \qquad (5.26) \\ &= \frac{1}{2}\tilde{\boldsymbol{\mu}}^\top (K + \tilde{S}^{-1})^{-1} \frac{\partial K}{\partial \theta_j} (K + \tilde{S}^{-1})^{-1} \tilde{\boldsymbol{\mu}} - \frac{1}{2} \mathrm{tr}\big((K + \tilde{S}^{-1})^{-1} \frac{\partial K}{\partial \theta_j}\big). \end{aligned}$$

In Algorithm 5.2 the derivatives from eq. (5.26) are implemented using

$$\frac{\partial \log Z_{\mathrm{EP}}}{\partial \theta_j} = \frac{1}{2} \mathrm{tr}\Big( \big(\mathbf{b}\mathbf{b}^\top - \tilde{S}^{\frac{1}{2}} B^{-1} S^{\frac{1}{2}}\big) \frac{\partial K}{\partial \theta_j} \Big), \qquad (5.27)$$

where $\mathbf{b} = (I - \tilde{S}^{\frac{1}{2}} B^{-1} \tilde{S}^{\frac{1}{2}} K)\tilde{\boldsymbol{\nu}}$.

### 5.5.3 Cross-validation

Whereas the LOO-CV estimates were easily computed for regression through the use of rank-one updates, it is not so obvious how to generalize this to classification. Opper and Winther [2000, sec. 5] use the cavity distributions of their mean-field approach as LOO-CV estimates, and one could similarly use the cavity distributions from the closely-related EP algorithm discussed in

section 3.6. Although technically the cavity distribution for site $i$ could depend on the label $y_i$ (because the algorithm uses all cases when converging to its fixed point), this effect is probably very small and indeed Opper and Winther [2000, sec. 8] report very high precision for these LOO-CV estimates. As an alternative $k$-fold CV could be used explicitly for some moderate value of $k$.

**Other Methods for Setting Hyperparameters**

Above we have considered setting hyperparameters by optimizing the marginal likelihood or cross-validation criteria. However, some other criteria have been proposed in the literature. For example Cristianini et al. [2002] define the *alignment* between a Gram matrix $K$ and the corresponding $+1/-1$ vector of targets $\mathbf{y}$ as

alignment

$$A(K, \mathbf{y}) = \frac{\mathbf{y}^\top K \mathbf{y}}{n \|K\|_F}, \tag{5.28}$$

where $\|K\|_F$ denotes the Frobenius norm of the matrix $K$, as defined in eq. (A.16). Lanckriet et al. [2004] show that if $K$ is a convex combination of Gram matrices $K_i$ so that $K = \sum_i \nu_i K_i$ with $\nu_i \geq 0$ for all $i$ then the optimization of the alignment score w.r.t. the $\nu_i$'s can be achieved by solving a semidefinite programming problem.

### 5.5.4 Example

For an example of model selection, refer to section 3.7. Although the experiments there were done by exhaustively evaluating the marginal likelihood for a whole grid of hyperparameter values, the techniques described in this chapter could be used to locate the same solutions more efficiently.

## 5.6 Exercises

1. The optimization of the marginal likelihood w.r.t. the hyperparameters is generally not possible in closed form. Consider, however, the situation where one hyperparameter, $\theta_0$ gives the overall scale of the covariance

$$k_y(\mathbf{x}, \mathbf{x}') = \theta_0 \tilde{k}_y(\mathbf{x}, \mathbf{x}'), \tag{5.29}$$

   where $k_y$ is the covariance function for the noisy targets (i.e. including noise contributions) and $\tilde{k}_y(\mathbf{x}, \mathbf{x}')$ may depend on further hyperparameters, $\theta_1, \theta_2, \ldots$. Show that the marginal likelihood can be optimized w.r.t. $\theta_0$ in closed form.

2. Consider the difference between the log marginal likelihood given by: $\sum_i \log p(y_i | \{y_j, j < i\})$, and the LOO-CV using log probability which is given by $\sum_i \log p(y_i | \{y_j, j \neq i\})$. From the viewpoint of the marginal likelihood the LOO-CV conditions *too much* on the data. Show that the *expected* LOO-CV loss is greater than the expected marginal likelihood.