

## 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.

## 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.

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.<sup>1</sup> In the following paragraphs we give example choices of parameterizations of distance measures for stationary covariance functions.

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

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

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

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

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  $\ell_1, \dots, \ell_D$  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

<sup>1</sup>This contrasts the use of the word in the SVM literature, where “training” usually refers to finding the support vectors for a fixed kernel.

<sup>2</sup>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.

enable interpretation

hyperparameters

training

characteristic  
length-scale  
automatic relevance  
determination

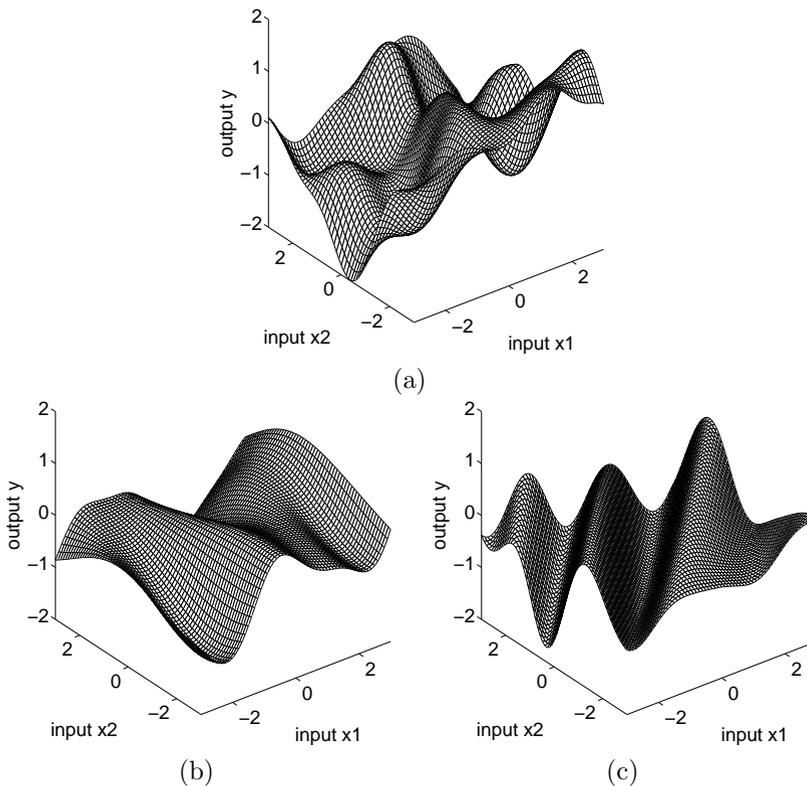


Figure 5.1: Functions with two dimensional input drawn at random from noise free squared exponential covariance function Gaussian processes, corresponding to the three different distance measures in eq. (5.2) respectively. The parameters were: (a)  $\ell = 1$ , (b)  $\ell = (1, 3)^\top$ , and (c)  $\Lambda = (1, -1)^\top$ ,  $\ell = (6, 6)^\top$ . 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

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,

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 are 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  $\theta$  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

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

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

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

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

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

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

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

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

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

level 3 inference

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

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

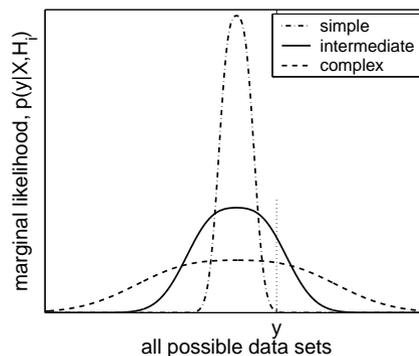


Figure 5.2: The marginal likelihood  $p(\mathbf{y}|X, \mathcal{H}_i)$  is the probability of the data, given the model. The number of data points  $n$  and the inputs  $X$  are fixed, and not shown. The horizontal axis is an idealized representation of all possible vectors of targets  $\mathbf{y}$ . 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

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.

automatic trade-off

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.<sup>3</sup>

## 5.3 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.

cross-validation

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 is 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)

<sup>3</sup>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."

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 properties. 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),

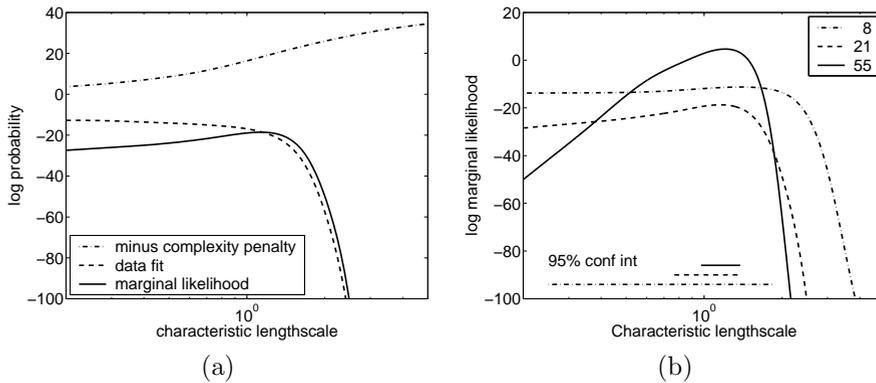


Figure 5.3: Panel (a) shows a decomposition of the log marginal likelihood into its constituents: data-fit and complexity penalty, as a function of the characteristic length-scale. The training data is drawn from a Gaussian process with SE covariance function and parameters  $(\ell, \sigma_f, \sigma_n) = (1, 1, 0.1)$ , 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

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

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.<sup>4</sup>

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

<sup>4</sup>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  $\boldsymbol{\theta}$ .

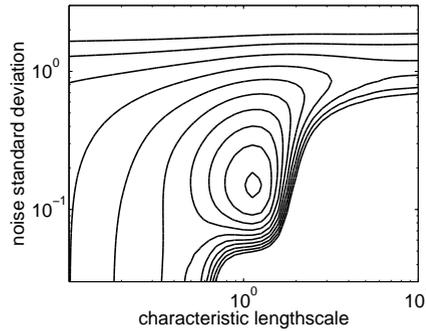


Figure 5.4: Contour plot showing the log marginal likelihood as a function of the characteristic length-scale and the noise level, for the same data as in Figure 2.5 and Figure 5.3. The signal variance hyperparameter was set to  $\sigma_f^2 = 1$ . The optimum is close to the parameters used when generating 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.

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

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

The computational 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.<sup>5</sup> Thus,

<sup>5</sup>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.

marginal likelihood  
gradient

the computational overhead of computing derivatives is small, so using a gradient based optimizer is advantageous.

Estimation of  $\theta$  by optimization 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  $\theta$ . 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.

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.

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

multiple local maxima

multi-task learning

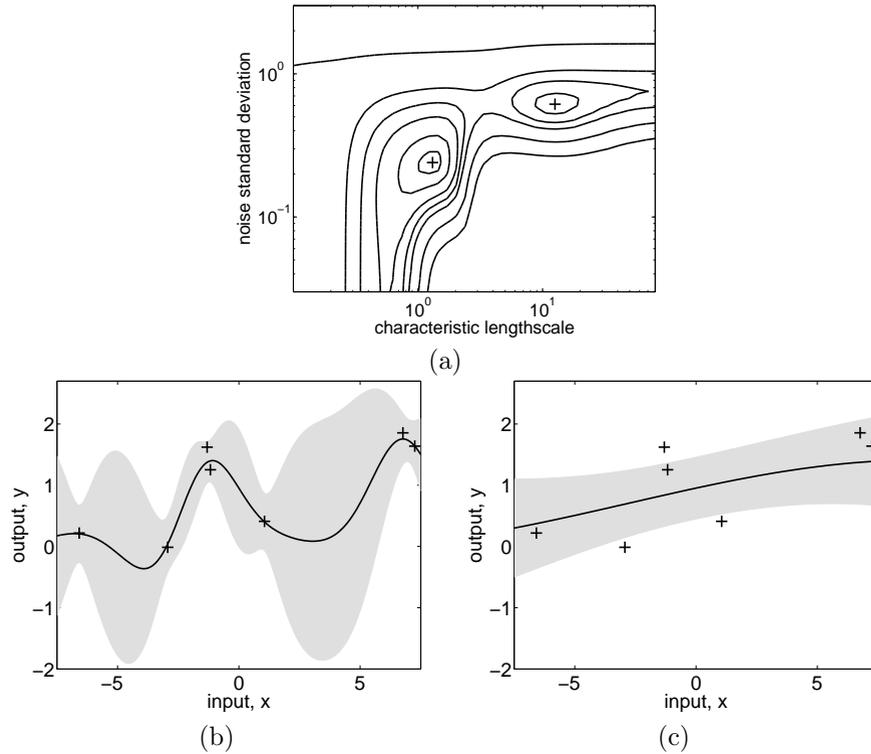


Figure 5.5: Panel (a) shows the marginal likelihood as a function of the hyperparameters  $\ell$  (length-scale) and  $\sigma_n^2$  (noise standard deviation), where  $\sigma_f^2 = 1$  (signal standard deviation) for a data set of 7 observations (seen in panels (b) and (c)). There are two local optima, 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

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

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

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

see [Geisser and Eddy, 1979] for a discussion of this and related approaches.  $L_{\text{LOO}}$  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

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

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  $L_{\text{LOO}}$  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

$$\frac{\partial \mu_i}{\partial \theta_j} = \frac{[Z_j \boldsymbol{\alpha}]_i}{[K^{-1}]_{ii}} - \frac{\boldsymbol{\alpha}_i [Z_j K^{-1}]_{ii}}{[K^{-1}]_{ii}^2}, \quad \frac{\partial \sigma_i^2}{\partial \theta_j} = \frac{[Z_j K^{-1}]_{ii}}{[K^{-1}]_{ii}^2}, \quad (5.13)$$

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

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

The computational complexity is  $\mathcal{O}(n^3)$  for computing the inverse of  $K$ , and  $\mathcal{O}(n^3)$  *per hyperparameter*<sup>6</sup> 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).

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

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<sup>7</sup> 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<sub>2</sub> 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

<sup>7</sup>In the special case where we know either the signal or the noise variance there is no indeterminacy.

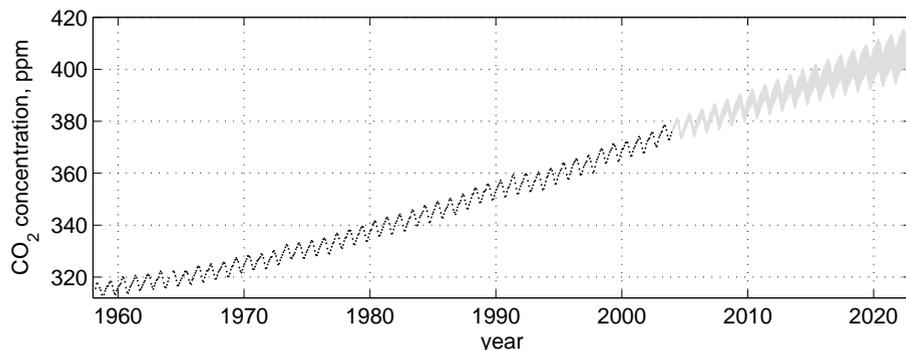


Figure 5.6: The 545 observations of monthly averages of the atmospheric concentration of CO<sub>2</sub> made between 1958 and the end of 2003, together with 95% predictive confidence region for a Gaussian process regression model, 20 years into the future. Rising trend and seasonal variations are clearly 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<sub>2</sub> 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).<sup>8</sup> The data is shown in Figure 5.6. Our goal is the model the CO<sub>2</sub> 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  $\theta_2$

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

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 exactly periodic, so we modify eq. (4.31) by taking the product with a squared

smooth trend

seasonal component

<sup>8</sup>The data is available from <http://cdiac.esd.ornl.gov/ftp/trends/co2/maunaloa.co2>.

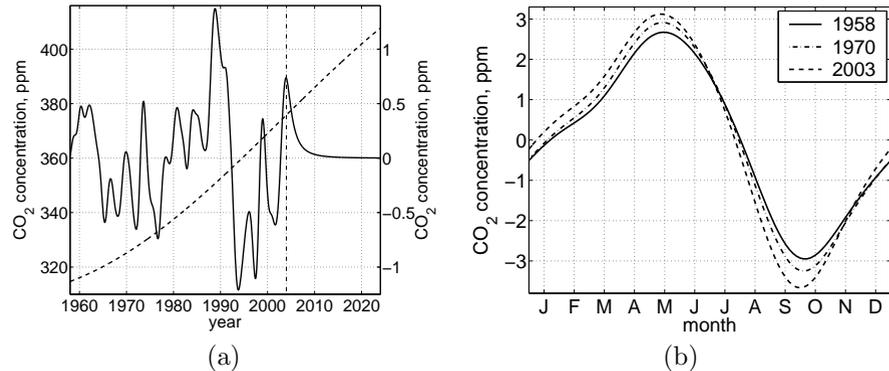


Figure 5.7: Panel (a): long term trend, dashed, left hand scale, predicted by the squared exponential contribution; superimposed is the medium term trend, full line, right hand scale, predicted by the rational quadratic contribution; the vertical dash-dotted line indicates the upper limit of the training 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.

exponential component (using the product construction from section 4.2.4), to allow a *decay* away from exact periodicity

$$k_2(x, x') = \theta_3^2 \exp\left(-\frac{(x-x')^2}{2\theta_4^2} - \frac{2\sin^2(\pi(x-x'))}{\theta_5^2}\right), \quad (5.16)$$

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<sub>2</sub> 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<sub>2</sub> 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)

$$k_3(x, x') = \theta_6^2 \left(1 + \frac{(x-x')^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}, \quad (5.17)$$

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.

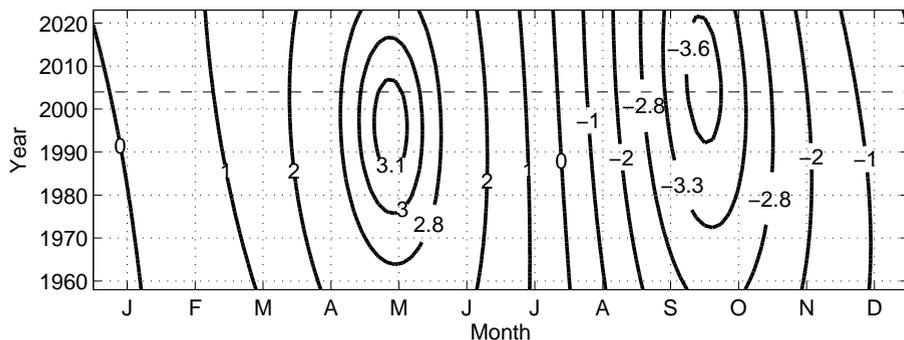


Figure 5.8: The time course of the seasonal effect, plotted in a months vs. year plot (with wrap-around continuity between the edges). The labels on the contours are in ppmv of CO<sub>2</sub>. The training period extends up to the dashed line. Note the slow development: the height of the May peak may 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).

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

noise terms

$$k_4(x, x') = \theta_9^2 \exp\left(-\frac{(x - x')^2}{2\theta_{10}^2}\right) + \theta_{11}^2 \delta_{xx'}, \quad (5.18)$$

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

$$k(x, x') = k_1(x, x') + k_2(x, x') + k_3(x, x') + k_4(x, x'), \quad (5.19)$$

with hyperparameters  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_{11})^\top$ . 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

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<sub>2</sub> 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.

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

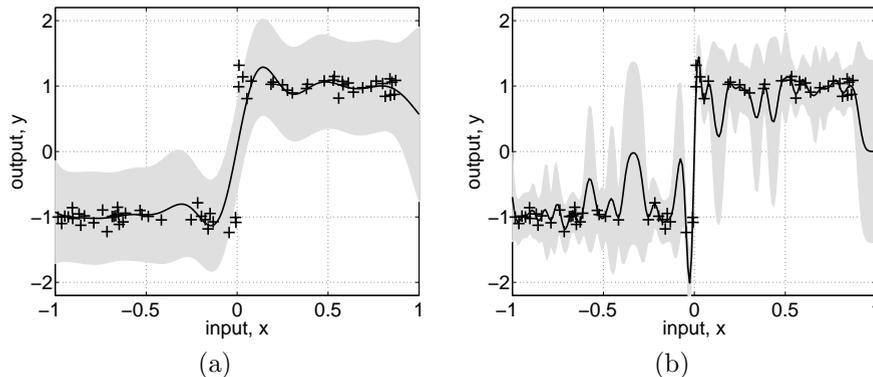


Figure 5.9: Mis-specification example. Fit to 64 datapoints drawn from a step function with Gaussian noise with standard deviation  $\sigma_n = 0.1$ . The Gaussian process models are using a squared exponential covariance function. Panel (a) shows the mean and 95% confidence interval for the noisy signal 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.

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  $L_{\text{LOO}}$ . 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 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

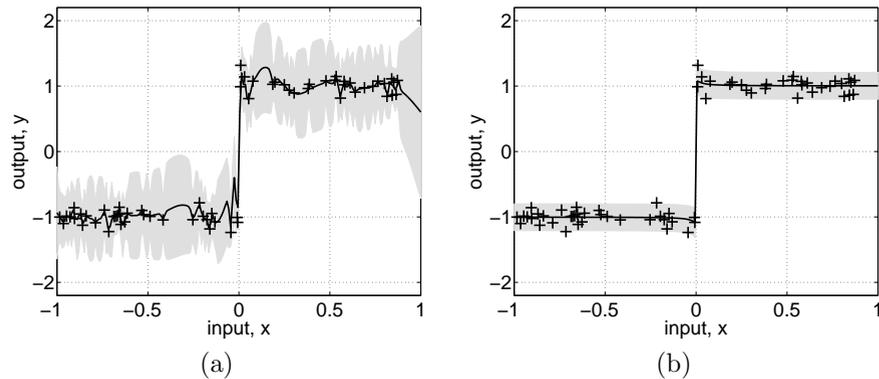


Figure 5.10: Same data as in Figure 5.9. Panel (a) shows the result of using a covariance function which is the sum of two squared-exponential terms. Although this is still a stationary covariance function, it gives rise to a higher marginal likelihood than for the squared-exponential covariance function 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.

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 a 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 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

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

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

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

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

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

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

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

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

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

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).

#### 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-

<b>input:</b> $X$ (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 $\boldsymbol{\theta}$
$\mathbf{f} := \text{mode}(K, \mathbf{y}, p(\mathbf{y} \mathbf{f}))$	locate the posterior mode using Algorithm 3.1
4: $W := -\nabla\nabla \log p(\mathbf{y} \mathbf{f})$	
$L := \text{cholesky}(I + W^{\frac{1}{2}}KW^{\frac{1}{2}})$	solve $LL^T = B = I + W^{\frac{1}{2}}KW^{\frac{1}{2}}$
6: $E := -\frac{1}{2}\mathbf{a}^T\mathbf{f} + \log p(\mathbf{y} \mathbf{f}) - \sum \log(\text{diag}(L))$	eq. (5.20)
$Z := W^{\frac{1}{2}}L^T \setminus (L \setminus W^{\frac{1}{2}})$	$Z = W^{\frac{1}{2}}(I + W^{\frac{1}{2}}KW^{\frac{1}{2}})^{-1}W^{\frac{1}{2}}$
8: $C := L \setminus (W^{\frac{1}{2}}K)$	} eq. (5.23)
$\mathbf{s}_2 := -\frac{1}{2} \text{diag}(\text{diag}(K) - \text{diag}(C^T C)) \nabla^3 \log p(\mathbf{y} \mathbf{f})$	
10: <b>for</b> $j := 1 \dots \text{dim}(\boldsymbol{\theta})$ <b>do</b>	
$C := \partial K / \partial \theta_j$	compute derivative matrix from $X$ and $\boldsymbol{\theta}$
12: $\mathbf{s}_1 := \frac{1}{2}\mathbf{a}^T C \mathbf{a} - \frac{1}{2} \text{tr}(ZC)$	eq. (5.22)
$\mathbf{b} := C \nabla \log p(\mathbf{y} \mathbf{f})$	} eq. (5.24)
14: $\mathbf{s}_3 := \mathbf{b} - KZ\mathbf{b}$	
$\nabla_j E := \mathbf{s}_1 + \mathbf{s}_2^T \mathbf{s}_3$	eq. (5.21)
16: <b>end for</b>	
<b>return:</b> $E$ (log marginal likelihood), $\nabla E$ (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 11 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).

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

$$Z = (W^{-1} + K)^{-1} = W^{\frac{1}{2}}(I + W^{\frac{1}{2}}KW^{\frac{1}{2}})^{-1}W^{\frac{1}{2}}, \quad (5.25)$$

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. (5.22-5.22) is straightforward, and for eq. (5.24) we apply the matrix inversion lemma (eq. (A.9)) to  $B^{-1} = (I + W^{\frac{1}{2}}KW^{\frac{1}{2}})^{-1}$  to obtain  $I - KZ$ , 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  $Z$  in line 7 is also  $\mathcal{O}(n^3)$ , all other computations being at most  $\mathcal{O}(n^2)$  per hyperparameter.

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

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

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

$$\frac{\partial Z_{\text{EP}}}{\partial \theta_j} = \frac{1}{2} \text{tr} \left( (\mathbf{b}\mathbf{b}^{\top} - \tilde{S}^{\frac{1}{2}} B^{-1} \tilde{S}^{\frac{1}{2}}) \frac{\partial K}{\partial \theta_j} \right), \quad (5.27)$$

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

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

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

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

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,  $\theta_1, \theta_2, \dots$ . 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.

alignment