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.

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.

two equivalent views

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 σ_n^2

bias, offset

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

likelihood

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

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

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

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

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

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

prior

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

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

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

posterior

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

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

marginal likelihood

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

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

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

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

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

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

MAP estimate

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

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

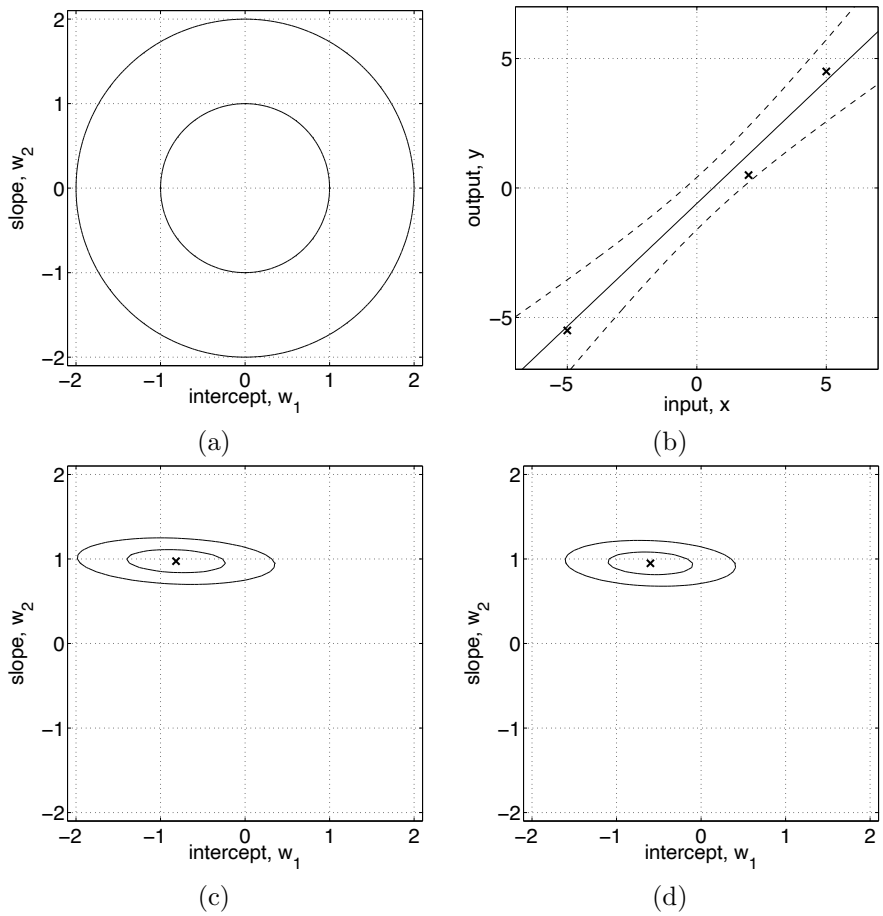


Figure 2.1: Example of Bayesian linear model $f(x) = w_1 + w_2 x$ with intercept w_1 and slope parameter w_2. Panel (a) shows the contours of the prior distribution $p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, I)$, eq. (2.4). Panel (b) shows three training points marked by crosses. Panel (c) shows contours 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 Σ_p). As Σ_p is positive definite we can define $\Sigma_p^{1/2}$ so that $(\Sigma_p^{1/2})^2 = \Sigma_p$; for example if the SVD (singular value decomposition) of $\Sigma_p = UDU^\top$, where D is diagonal, then one form for $\Sigma_p^{1/2}$ is $UD^{1/2}U^\top$. Then defining $\boldsymbol{\psi}(\mathbf{x}) = \Sigma_p^{1/2}\boldsymbol{\phi}(\mathbf{x})$ we obtain a simple dot product representation $k(\mathbf{x},\mathbf{x}') = \boldsymbol{\psi}(\mathbf{x})\cdot\boldsymbol{\psi}(\mathbf{x}')$.

kernel

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

kernel trick

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

2.2 Function-space View

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

Gaussian process

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

covariance and mean function

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

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

and will write the Gaussian process as

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

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

index set ≡ input domain

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

marginalization property

finite index set

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

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

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

Bayesian linear model is a Gaussian process

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

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

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

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

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

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

basis functions

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

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

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

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

smoothness

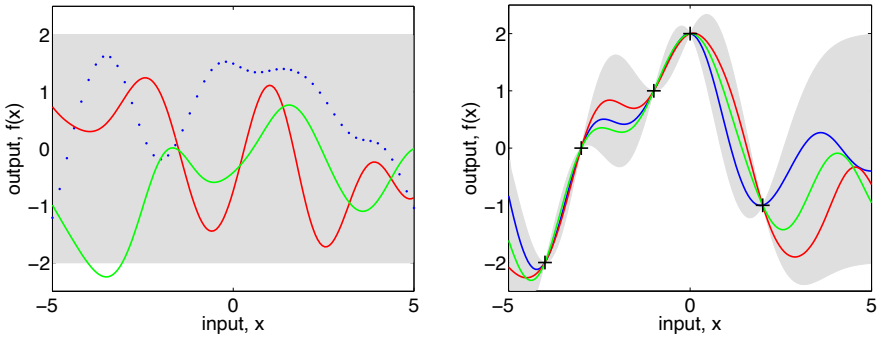
characteristic length-scale

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

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

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

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



(a), prior (b), posterior

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

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

magnitude

Prediction with Noise-free Observations

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

joint prior

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

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

graphical rejection

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

noise-free predictive distribution

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

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

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

Prediction using Noisy Observations

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

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

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

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

predictive distribution

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

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

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

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

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

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

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

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

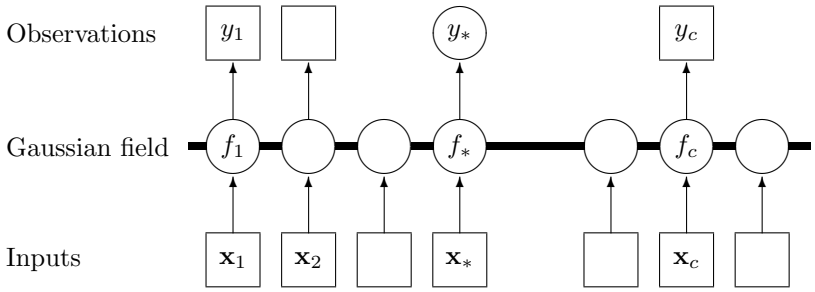


Figure 2.3: Graphical model (chain graph) for a GP for regression. Squares represent observed variables and circles represent unknowns. The thick horizontal bar represents a set of fully connected nodes. Note that an observation y_i is conditionally independent of all other nodes given the corresponding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

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

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

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

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

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

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

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

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

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

The plots of equivalent kernels in Figure 2.6 were made by using a dense grid of n_{grid} points on $[0, 1]$ and then computing the smoother matrix $K(K + \sigma^2_{\text{grid}} I)^{-1}$. Each row of this matrix is the equivalent kernel at the appropriate location. However, in order to get the scaling right one has to set $\sigma^2_{\text{grid}} = \sigma_n^2 n_{\text{grid}}/n$; for $n_{\text{grid}} > n$ this means that the effective variance at each of the n_{grid} points is larger, but as there are correspondingly more points this effect cancels out. This can be understood by imagining the situation if there were n_{grid}/n independent Gaussian observations with variance σ^2_{grid} at a single x-position; this would be equivalent to one Gaussian observation with variance σ_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 to model the residuals from a polynomial regression, i.e. $\mathbf{h}(x) = (1, x, x^2, \ldots)$. When fitting the model, one could optimize over the parameters $\boldsymbol{\beta}$ jointly with the hyperparameters of the covariance function. Alternatively, if we take the prior on $\boldsymbol{\beta}$ to be Gaussian, $\boldsymbol{\beta} \sim \mathcal{N}(\mathbf{b}, B)$, we can also integrate out these parameters. Following O'Hagan [1978] we obtain another GP

polynomial regression

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

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

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

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

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

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

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