Contents

Chapter 1

Gaussian Processes

In this chapter Gaussian process regression is introduced. First, some concepts and terminology in Bayesian statistics are reviewed. The following section introduces the mathematical framework needed for Gaussian processes, and selected covariance functions are discussed. Concepts in Bayesian model selection are used as a basis to quantify and improve the quality of predictions. Finally, distributed Gaussian processes are introduced as a way of scaling Gaussian processes to larger datasets.

1.1 Introduction to Bayesian Statistics

There are two general philosophies in statistics, namely *Bayesian* and *frequentist* statistics. To understand where they differ, consider a statement statisticians from both branches would probably consider to be true

Statisticians use probability to describe uncertainty.

The difference between Bayesian and frequentist statistics is at the definition of the *uncertain*. Since uncertainty is described by probability this understanding must also vary, and one distinguishes between *objective* and *subjective* probability. Consider an example in which a statistician throws a die. Before throwing, he is uncertain about the outcome of the toss. This uncertainty related to the outcome is *objective*: no one can know if he will throw a 1 or a 4. On the other hand, he might also be uncertain about the underlying probability distribution of the toss. Is the die loaded? Is one of the edges sharper than the others? This uncertainty is *subjective*, as it may vary depending on how much information is available about the system, and how that information is used. One of the main critisisms of subjective probability posed by frequentists is that the final probability depends on who you ask.

1.1.1 Bayes' Theorem

To further illustrate the difference between frequentist and Bayesian statistics *Bayes' theorem* [1] is introduced. Bayes' theorem can be derived from the familiar rules of probability

$$P(X|I) + P(\bar{X}|I) = 1, \tag{1.1}$$

$$P(X, Y|I) = P(X|Y, I) \times P(Y|I), \tag{1.2}$$

commonly known as the *sum rule* and *product rule*, respectively. $P(X|I)$ is the probability of outcome X given the information I, and $P(X|Y, I)$ is the probability of outcome X given the information I *and* outcome Y. The bar over $\bar{X}$ means that the outcome X does *not* happen. The sum rule states that the total probability of the outcomes X and $\bar{X}$ is equal to 1. This is rather intuitive, considering that an event either takes place or not. The second rule, the product rule, states that the probability of both outcomes X and Y is equal to the probability of Y times the probability of X given that Y has ocurred. These expressions combine to give Bayes' theorem, first formulated by the reverend Thomas Bayes in 1763,

$$P(X|Y, I) = \frac{P(Y|X, I) \times P(X|I)}{P(Y|I)}. \tag{1.3}$$

This theorem states that the probability of X given Y equals the probability of Y given X times the probability of X, divided by the probability of Y. Surprisingly, there is nothing Bayesian about Bayes' theorem. It is merely a reformulation of the rules of logical consistent reasoning by Richard Cox in 1946 [6]. Laplace was the one to make Bayes' theorem Bayesian in the modern statistical sense, when he used the theorem to perform inference about probability distributions [3]. The resulting expression is the *posterior probability distribution*

$$P(\Theta|X, I) = \frac{P(X|\Theta, I)P(\Theta|I)}{P(X|I)}, \tag{1.4}$$

where Θ are the probability distribution parameters, X are the possible outcomes, $P(X|\Theta, I)$ and $P(\Theta|I)$ are the *likelihood* and *prior*, respectively, and $P(X|I)$ is a normalization constant called the *marginal likelihood* or evidence. The marginal likelihood is independent of the parameters

$$P(X|I) = \int P(X|\Theta, I)P(\Theta|I)d\Theta. \tag{1.5}$$

In other words, Eq. (1.4) states the probability of the parameters Θ given the knowledge of outcomes X.

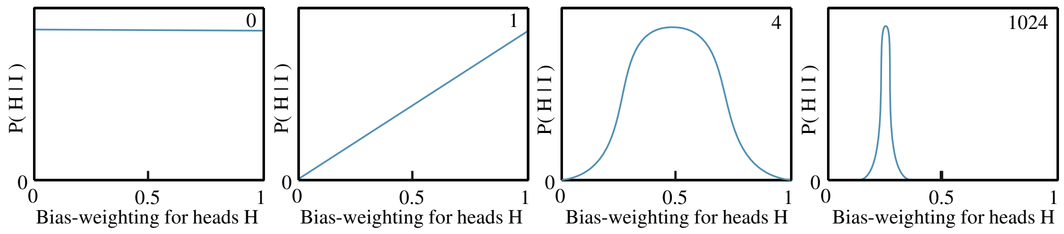


Figure 1.1: The posterior probability distribution of the bias-weighting of a coin for the uniform prior, $P(H|I)$. The first panel from the left is before the coin is tossed, the second panel is after 1 toss, the third is after 4 tosses, and the fourth is after 1024 tosses. The posterior converges towards a narrow peak at 0.25, so the coin is biased.

A crucial parting of Bayesian statistics from frequentist statistics is at the introduction of the *prior*, which expresses a probability distribution on the *parameters* of the probability distribution before data. The prior and likelihood are discussed in the next section, while the marginal likelihood is revisited in Sec. 1.4.

1.1.2 Priors and Likelihood

The likelihood $P(X|\Theta, I)$ is simply the probability of the observations X given the parameters of the probability distribution Θ, and is revisited in Sec. 1.4.1. Conversely, the prior expresses a prior belief or assumption of the parameters, and has to be determined beforehand. As mentioned previously, the measure $P(\Theta|X, I)$ from Eq. (1.4) is called the posterior distribution. This can be thought of as the prior belief, modified by how well this belief fits the data,

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{marginal likelihood}}.$$

Consider an example. The statistician from before now sets about tossing a coin. Before tossing he assumes the probability of heads is uniformly distributed, and so adopts a flat, or uniform, prior probability distribution. The uniform distribution is illustrated in the first panel in Fig. 1.1. After one toss he gets heads, and the posterior changes to a function with high probability for heads, and low for tails, illustrated in the second panel. After four tosses, of which two gave heads and two gave tails, the posterior in the third panel shows an equal probabilty for heads and tails, with a wide distribution centered at 0.5. After several tosses the distribution converges to a narrow peak around 0.25, illustrated in the fourth panel. This indicates an unfair coin that is biased towards tails.

1.1.3 Best Estimate and Reliability

Given a posterior distribution $P(X|\mathcal{D}, I)$ over some random variable X where the prior, $P(X|I)$, has been modified by some data $\mathcal{D}$, it is important to decide how well the posterior fits the data. As will be shown, the posterior can be approximated by a Gaussian distribution, defined by the *best estimate* and *reliability* of the posterior. The best estimate X_0 is the outcome with the highest probability. In other words, it is the maximum of the posterior distribution

$$\frac{dP}{dX}\bigg|_{X_0} = 0, \qquad \frac{d^2P}{dX^2}\bigg|_{X_0} < 0, \tag{1.6}$$

where P is the posterior $P(X|\mathcal{D}, I)$. The second derivative must be negative to ensure that X_0 is, in fact, a maximum.

Once a best estimate is found, it is important to know how reliable it is. Reliability, or uncertainty, is connected to the width of the distribution. The width of the distribution tells how much the random variables X are smeared out around the mean value X_0. A narrow distribution has low uncertainty, while a wide distribution has large uncertainty. As an example, the third panel in Fig. 1.1 shows a distribution with a mean value of 0.5 with large uncertainty, while the fourth panel shows a distribution with mean 0.25 with small uncertainty.

The width is found by taking the logarithm [1] and Taylor expanding the posterior distribution $P(X|\mathcal{D}, I)$

$$L = L(X_0) + \frac{1}{2}\frac{d^2L}{dX^2}\bigg|_{X_0}(X - X_0)^2 + ..., \qquad L = \log_e\Big[P(X|\mathcal{D}, I)\Big] \tag{1.7}$$

The first term, $L(X_0)$, is just a constant. From Eq. (1.6) the condition of the best estimate is that $dL/dX|_{X_0} = 0$. The dominant term in determining the width is therefore the quadratic term in Eq. (1.7).

The Gaussian Distribution

Taking the exponential of Eq. (1.7) and ignoring higher order terms, the posterior can then be approximated as

$$P(X|\mathcal{D}, I) \approx A \exp\left[\frac{1}{2}\frac{d^2L}{dX^2}\bigg|_{X_0}(X - X_0)^2\right], \tag{1.8}$$

where $A = \exp\big[L(X_0)\big]$ is a constant.

Equation (1.8) is now in the shape of a *Gaussian distribution*, given by

$$P(X|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(X-\mu)^2}{2\sigma^2}\right], \tag{1.9}$$

[1] L is a monotonic function of P, so the maximum of L is at the maximum of P.

where μ and σ^2 are the two parameters of the Gaussian distribution mentioned above. An example of a Gaussian distribution is shown in Fig. 1.2. μ is the *mean* value of X, which will be written as $m(X)$, and σ^2 is shorthand for the *variance* of the distribution around the mean $m(X)$, which will be written as $\mathbb{V}(X)$. The variance is discussed further below. The mean and variance for the approximated Gaussian distribution of $P(X|\mathcal{D}, I)$ are then given by

$$m(X) = X_0, \; \mathbb{V}(X) = \Big(-\frac{d^2L}{dX^2}\Big)^{-1/2}. \tag{1.10}$$

The Gaussian distribution is also referred to as the *normal distribution*, and a Gaussian distribution with mean $m(X) = \mu$ and variance $\mathbb{V}(X) = \sigma^2$ is therefore written as $\mathcal{N}(\mu, \sigma^2)$. A random variable X drawn from this distribution is denoted by a tilde, $X \sim \mathcal{N}(\mu, \sigma^2)$. The Gaussian distribution is symmetric with respect to the maximum at the mean μ, and has a full width at half maximum (FWHM) at around 2.35σ, where $\sigma = \sqrt{\sigma^2}$ is the standard deviation.

In Gaussian process regression of a function $f(X)$ the Gaussian distribution is central, as the basic idea is to predict a Gaussian distribution over function values $f(X)$ for each X. Gaussian processes are discussed in Sec. 1.3, so for now it will suffice to sum up that the quality of a posterior distribution, $P(X|\mathcal{D}, I)$, can be summed up in two measures: the best estimate and the reliability. These can be seen as the mean and variance of a Gaussian distribution $\mathcal{N}(m(X), \mathbb{V}(X))$,

$$m(X) \; : \; \text{Mean of } X, \tag{1.11}$$

$$\mathbb{V}(X) \; : \; \text{Variance of } X. \tag{1.12}$$

Variance

Now that some basics of Gaussian processes have been covered, a more formal definition of the variance is in order. The variance $\mathbb{V}(X)$ is defined as the expectation value of the square of deviations from the mean. The expectation value of a random variable X with a probability distribution $P(X|I)$ is here written as $\mathbb{E}[X]$, and defined as

$$\mathbb{E}[X] = \int XP(X|I)dX \; : \; \text{Expectation value of } X. \tag{1.13}$$

For the posterior probability distribution $P(X|\mathcal{D}, I)$ the variance of the random variable X is then given by [6]

$$\mathbb{V}(X) = \mathbb{E}\big[(X - X_0)^2\big] = \int\int (X - X_0)^2 P(X|\mathcal{D}, I)dX. \tag{1.14}$$

The variance in X is often denoted $\sigma_X^2 = \mathbb{V}(X)$, and its square root is the *standard deviation* $\sqrt{\sigma_X^2} = \sigma_X$.

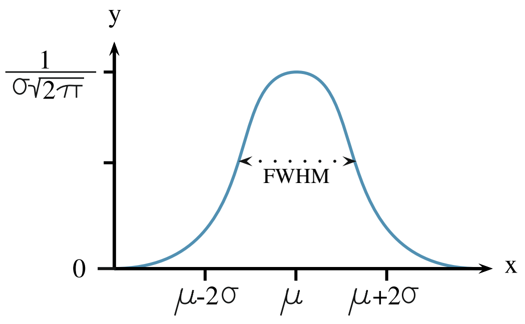


Figure 1.2: A Gaussian probability distribution. The maximum is at the mean value μ, with a full width at half maximum (FWHM) at around 2.35σ. Figure from [6].

1.1.4 Covariance

In distributions over several random variables, $P(X_i|\mathcal{D}, I)$, varying one variable can affect the variance of another variable. This is called *covariance*. For these distributions the equations are not as simple to solve as in Eq. (1.7). In the case of several random variables X_i, a set of *simultaneous equations* must be solved to get the best estimate

$$\frac{dP}{dX_i}\bigg|_{X_{0j}} = 0, \qquad\qquad \frac{d^2P}{dX_i^2}\bigg|_{X_{0j}} < 0 \tag{1.15}$$

To simplify expressions consider the problem in two dimensions, so that $\{X_i\} = (X, Y)$. Analogously to Eq. (1.7), the Taylor expansion of $L = \log_e \Big[P(X,Y|I)\Big]$ is found

$$\begin{aligned} L =& L(X_0, Y_0) + \frac{1}{2}\Big[\frac{d^2L}{dX^2}\bigg|_{X_0,Y_0}(X - X_0)^2 \\ &+ \frac{d^2L}{dY^2}\bigg|_{X_0,Y_0}(Y - Y_0)^2 + 2\frac{d^2L}{dXdY}\bigg|_{X_0,Y_0}(X - X_0)(Y - Y_0)\Big] + ... \end{aligned} \tag{1.16}$$

There are now four partial derivatives, reduced to three using the rules for mixed partial derivatives $\frac{\partial^2}{\partial X \partial Y} = \frac{\partial^2}{\partial Y \partial X}$. Writing the quadratic terms of Eq. (1.16) in matrix form gives

$$Q = \begin{pmatrix} X - X_0 & Y - Y_0 \end{pmatrix} \begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} X - X_0 \\ Y - Y_0 \end{pmatrix}, \tag{1.17}$$

where the matrix elements are

$$A = \frac{\partial^2 L}{\partial X^2}\bigg|_{X_0,Y_0}, \qquad B = \frac{\partial^2 L}{\partial Y^2}\bigg|_{X_0,Y_0}, \qquad C = \frac{\partial^2 L}{\partial X \partial Y}\bigg|_{X_0,Y_0}. \tag{1.18}$$

The expression for the variance of X for the distribution $P(X,Y|I)$ is very similar to Eq. (1.14), except for an additional integral over the random variable Y

$$\mathbb{V}(X) = \sigma_X^2 = \mathbb{E}\big[(X-X_0)^2\big] = \int\int (X-X_0)^2 P(X,Y|\mathcal{D},I)dXdY. \tag{1.19}$$

A similar expression, σ_Y^2, can be found for Y by substituting X and Y.

The simultaneous deviations of the random variables X and Y is the aforementioned covariance, often written as σ_{XY}^2. In two dimensions the covariance is given by

$$\sigma_{XY}^2 = \mathbb{E}\big[(X-X_0)(Y-Y_0)\big] = \int\int (X-X_0)(Y-Y_0)P(X,Y|\mathcal{D},I)dXdY. \tag{1.20}$$

The covariance indicates how an over- or underestimation of one random variable affects another. If, for example, an overestimation of X leads to an overestimation of Y, the covariance is positive. An example of positive covariance is shown in the third panel of Fig. 1.3. If the overestimation of X has little or no effect on the estimation of Y, the covariance is negligible or zero $|\sigma_{XY}| \ll \sqrt{\sigma_X^2\sigma_Y^2}$, as seen in the first panel of Fig. 1.3. The second panel shows negaitve covariance.

Covariance Matrix

The variances and covariances are the elements of the *covariance matrix*. For N random variables $X_1, ..., X_N$ the covariance matrix is an $N \times N$-matrix. The covariance matrix for X and Y is here denoted $\text{cov}(X,Y)$, and it can be shown that [6]

$$\text{cov}(X,Y) = \begin{pmatrix} \sigma_X^2 & \sigma_{XY}^2 \\ \sigma_{XY}^2 & \sigma_Y^2 \end{pmatrix} = -\begin{pmatrix} A & C \\ C & B \end{pmatrix}^{-1}. \tag{1.21}$$

To sum up, the posterior probability distribution over a random variable X is denoted $P(X|\mathcal{D},I)$, and its best estimate and the associated reliability can be approximated as a Gaussian distribution $\mathcal{N}(m(X),\mathbb{V}(X))$. The Gaussian distribution is defined by the mean value $m(X)$ and the variance $\mathbb{V}(X)$. For distributions over several random variables X_i one can also find the covariance $\sigma_{X_iX_j}^2$, and all variances and covariances are contained in the covariance matrix $\text{cov}(X_i)$. In the next section *covariance functions* are introduced, which are used to calculate the elements of the covariance matrix.

1.2 Covariance Functions

As mentioned in Sec. 1.1.4, the elements of a covariance matrix can be determined by *covariance functions*, or *kernels*. A function that maps two arguments

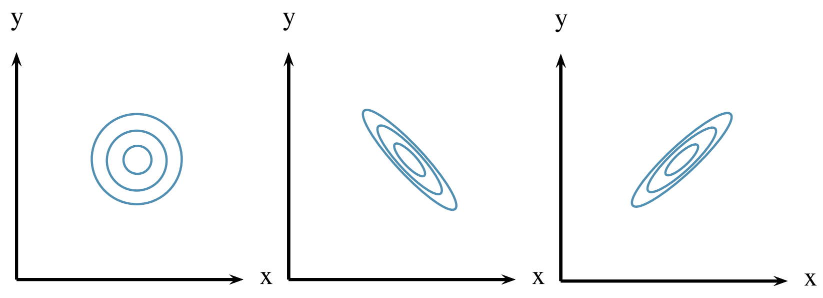


Figure 1.3: A schematic illustration of covariance and correlation. (a) The contours of a posterior pdf with zero covariance, where the inferred values of X and Y are uncorrelated. (b) The corresponding plot when the covariance is large and negative; (c) The case of positive correlation.

$\mathbf{x} \in \mathcal{X}$, $\mathbf{x}' \in \mathcal{X}$ into $\mathbb{R}$ is generally called a kernel k. Covariance functions are symmetric kernels, meaning that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. In this project both kernel and covariance function are taken to mean covariance function. As will be discussed, in Gaussian processes the covariance between two function values $f(\mathbf{x})$ and $f(\mathbf{x}')$, where $\mathbf{x}$ and $\mathbf{x}'$ are input vectors, is decided by the covariance of the input vectors, given by a kernel $\text{cov}(f(\mathbf{x}), f(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}')$. As previously mentioned, the matrix containing all the covariance elements is called the *covariance matrix*, or the Gram matrix K, whose elements are given by

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{1.22}$$

Note that a covariance matrix calculated using a covariance function k is here denoted by a capital K.

A kernel function that only depends on the difference between two points, $\mathbf{x} - \mathbf{x}'$, is called *stationary*. This implies that the function is invariant to translations in input space. If, in addition, it only depends on the length $r = |\mathbf{x} - \mathbf{x}'|$, the function is *isotropic* [2]. Isotropic functions are commonly referred to as *radial basis functions* (RBFs). The covariance function can also depend on the dot product, $\mathbf{x} \cdot \mathbf{x}'$, and is then called a *dot product* covariance function. The most important covariance functions for this project are the squared exponential covariance function and the Matérn class of covariance functions.

1.2.1 The Squared Exponential Covariance Function

The *squared exponential covariance function* (SE) has the form

$$k_{SE}(r) = \exp\left(-\frac{r^2}{2\ell^2}\right), \tag{1.23}$$

[2]Invariant to rigid rotations in input space.

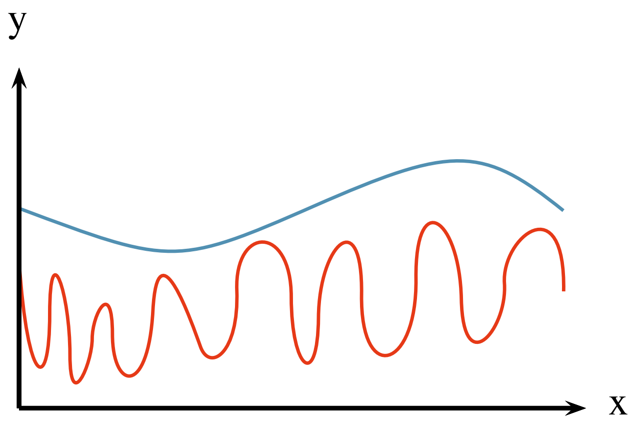


Figure 1.4: The effect of varying the length scale ℓ. A long length scale (blue) gives a smooth, slowly varying function, while a short length scale (red) gives a more staccato, quickly varying function.

where ℓ is the *characteristic length scale*. The length scale determines the smoothness of the function. For a large length scale one should expect a very slowly varying function, while a shorter length scale means a more rapidly varying function, see the illustration in Fig. 1.4. The SE is infinitely differentiable and therefore very smooth.

The SE is implemented in `scikit-learn` under the name radial basis function (RBF), and may be called in the following way for length scale 10, with bounds on the length scale $[0.01, 100]$

```
from sklearn.gaussian_process.kernels import RBF
rbf = RBF(length_scale=10, length_scale_bounds=(1e-2, 1e2))
```

1.2.2 The Matérn Class of Covariance Functions

The *Matérn class of covariance functions* is given by

$$k_{Matérn}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\ell} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{\ell} \right), \tag{1.24}$$

where $\nu, \ell > 0$, and K_ν is a modified Bessel function. The hyperparameter ν controls the smoothness of the function, as opposed to the SE kernel which is by definition very smooth. For $\nu \to \infty$ this becomes the SE kernel, and for $\nu = 1/2$ is becomes the very rough absolute exponential kernel $k(r) = \exp(-r/\ell)$. In the case of half integer ν, $\nu = p + \frac{1}{2}$, the covariance function is simply the product of an exponential and a polynomial

$$k_{\nu=p+\frac{1}{2}} = \exp\Big(-\frac{\sqrt{2\nu}r}{\ell}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu}r}{\ell}\Big)^{p-i}. \tag{1.25}$$

In machine learning the two most common cases are for $\nu = 3/2$ and $\nu = 5/2$

$$k_{\nu=3/2}(r) = \Big(1+\frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{1.26}$$

$$k_{\nu=5/2}(r) = \Big(1+\frac{\sqrt{5}r}{\ell}+\frac{5r^2}{3\ell^2}\Big)\exp\Big(-\frac{\sqrt{5}r}{\ell}\Big). \tag{1.27}$$

In `scikit-learn` the hyperparameter ν is fixed, and so not optimized during training. The Matérn kernel is considered more appropriate for physical processes [5], and may be called in `scikit-learn` in the following way for length scale 10, length scale bounds $[0.01, 100]$ and $\nu = 3/2$

```
from sklearn.gaussian_process.kernels import Matern
matern = Matern(length_scale=10, length_scale_bounds=(1e-2,
    1e2), nu=1.5)
```

For values not in $\nu = [0.5, 1.5, 2.5, \infty]$ `scikit-learn` evaluates Bessel functions explicitly, which increases the computational cost by a factor as high as 10.

Noise

The covariance function can also contain information about noise in the data, represented by a constant term added to the diagonal of the covariance matrix

$$k(\mathbf{x}_i, \mathbf{x}_j)_{noise} = C\delta_{ij}, \tag{1.28}$$

where $C \in \mathbb{R}$ is a real, constant number, and δ_{ij} is the Dirac delta function. The noise is assumed to follow a Gaussian distribution with mean 0 and variance α, $\mathcal{N}(0, \alpha)$. In `scikit-learn` this can be implemented either by giving a fixed noise level `alpha` to the regressor function, or by using the `WhiteKernel`, which estimates the noise level from the data. This kernel is implemented in `scikit-learn` in the following way for noise level 0.001 with bounds $[10^{-10}, 1]$

```
from sklearn.gaussian_processes.kernels import WhiteKernel
whitenoise = WhiteKernel(noise_level=0.001,
    noise_level_bounds=(1e-10,1))
```

Hyperparameters

Each kernel has a vector of hyperparameters, *e.g.* $\boldsymbol{\theta} = (\{M\}, \sigma_f^2, \sigma_n^2)$ for the squared exponential kernel with noise

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j))^T M(\mathbf{x}_i - \mathbf{x}_j) + \sigma_n^2 \delta_{ij}, \tag{1.29}$$

where σ_f is a scaling factor, M contains the characteristic length scales ℓ and σ_n^2 is the Gaussian noise parameter. The matrix M can have several forms, amongst them

$$M_1 = \ell^{-2}\mathbb{1}, \qquad M_2 = \mathrm{diag}(\boldsymbol{\ell})^{-2}. \tag{1.30}$$

Choosing $\boldsymbol{\ell}$ to be a vector in stead of a scalar is in many cases useful, especially if the input vector $\mathbf{x}$ contains values of different scales, *e.g.* $\mathbf{x} = (x_1, x_2)$ where $x_1 \in [0, 1]$ and $x_2 \in [200, 3000]$. The length scale can be set to a vector in `scikit-learn` by giving the `length_scale` parameter as a `numpy` array of the same dimension as the input vector $\mathbf{x}$.

Other Covariance Functions

There are several covariance functions that are not discussed here. Kernels can be multiplied and summed to form new kernels, making the space of possible kernels infinite. For further details see chapter 4 in [5].

1.3 Gaussian Process Regression

Gaussian processes (GP) is a supervised machine learning method, designed to solve regression and probabilistic classification problems. Only regression is considered here. This section begins by introducing Gaussian processes and important notation, first in a general sense, and then in the function space view. Then distributions over functions are considered, followed by a short discussion on how functions can be drawn from these distributions. Finally, a quick overview of the noise-free model and the Gaussian noise nodel, and their covariance matrices, are given.

Gaussian Processes

Consider a set of points $\mathcal{D} = \{\mathbf{x}_i, y_i\}$, where y is some (possibly noisy) function of $\mathbf{x}$, $y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon$. Here, ε is the noise and $f(\mathbf{x})$ is the true value of the function. These points are illustrated by the black dots in Fig. 1.5. In machine learning $\mathcal{D}$ is the *training data*, as it is used to train the model. It consists of *features*, which are the components of the input vectors $\mathbf{x}_i$, and *targets*, which are the function

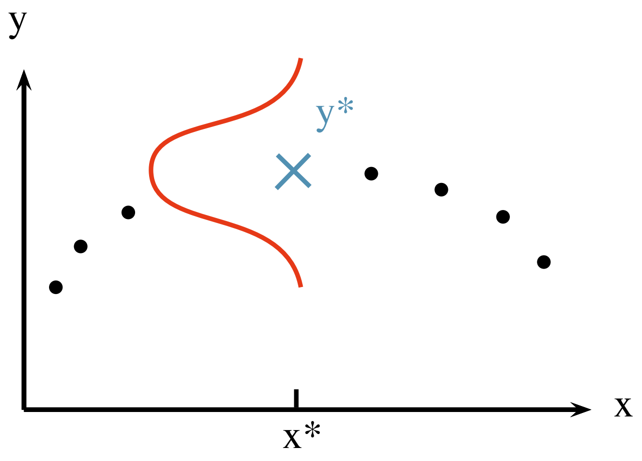


Figure 1.5: An illustration of a GP prediction of the target value y^* (blue cross), given the known set of points $\{x_i, y_i\}$ (black dots). The prediction is a Gaussian distribution in y with mean y^* and variance σ^2. The Gaussian distribution is drawn in red with y on the vertical axis, with uncertainty in the y-direction.

values y_i. The set of points is discrete, so there is some $\mathbf{x}^*$ for which the target y^* is unknown. The test point $\mathbf{x}^*$ is marked on the x-axis in Fig. 1.5.

Gaussian processes (GP) predict a Gaussian distribution *over function values* at this point $\mathbf{x}^*$. The distribution for a single test point $\mathbf{x}^*$ has a corresponding mean $m(\mathbf{x}^*)$ and variance $\mathbb{V}(\mathbf{x}^*)$. Note that the mean $m(\mathbf{x}^*)$ *is not the mean of the input vector* $\mathbf{x}^*$, but rather *the mean of function values* $f(\mathbf{x})$ *evaluated at* $\mathbf{x}^*$. The GP prediction for the target value $y^* = f(\mathbf{x}^*)$ is the mean $m(\mathbf{x}^*)$. Similarly, the variance, $\mathbb{V}(\mathbf{x}^*)$, is in fact the variance in function values $f(\mathbf{x}^*)$, or the width of the Gaussian distribution (red line) in the y-direction in Fig. 1.5. The mean value y^* is drawn as a blue cross in Fig. 1.5. As will be shown, the predicted target y^* is a linear combination of the known targets y_i, where the weights are controlled by the covariances between training points $\mathbf{x}_i$ and the test point $\mathbf{x}^*$.

Some Notation

As Gaussian process regression is notoriously confusing, it is helpful to begin with some notation.

As discussed in Sec. 1.3, the Gaussian distribution with a mean μ and variance σ^2 is denoted $\mathcal{N}(\mu, \sigma^2)$. The distribution can be over a single random variable, *e.g.* f, or a finite set of random variables, f_i. A Gaussian distribution over several random variables is called a *multivariate Gaussian distribution*. The n random variables f_i, $i = 1, ..., n$ drawn from a multivariate Gaussian distribution make up the n-dimensional vector $\mathbf{f}$. The mean values, μ_i, are then contained in the n-dimensional *mean vector* $\bar{\mathbf{f}}$. The variance, σ^2, is replaced by an $n \times n$-dimensional covariance matrix, $\text{cov}(\mathbf{f}, \mathbf{f})$. The multivariate Gaussian distribution over $\mathbf{f}$ is written as

$$\mathbf{f} \sim \mathcal{N}\big(\bar{\mathbf{f}},\ \text{cov}(\mathbf{f}, \mathbf{f})\big). \tag{1.31}$$

For n points $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i$ is an m-dimensional feature vector and y_i is the target, the features comprise the $n \times m$-matrix X. The targets make up the corresponding n-dimensional vector $\mathbf{y}$. A central assumption in Gaussian processes is that the covariance between targets y_i, y_j is given by the covariance of their features $\mathbf{x}_i$, $\mathbf{x}_j$

$$\text{cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j), \tag{1.32}$$

where $k(\mathbf{x}_i, \mathbf{x}_j)$ is a covariance function, as described in Sec. 1.2. In Gaussian processes the distribution over target values $\mathbf{f}$ will therefore be written

$$\mathbf{f} \sim \mathcal{N}\big(\ \bar{\mathbf{f}}, K(X, X)\ \big), \tag{1.33}$$

where $K(X, X)$ is the covariance matrix containing the covariances of the features contained in X, calculated using the covariance function $k(\mathbf{x}, \mathbf{x}')$.

Finally, a Gaussian process will be denoted $\mathcal{GP}$. It may be difficult to distinguish between a Gaussian *distribution*, $\mathcal{N}$, and a Gaussian *process*, $\mathcal{GP}$. The difference can be thought of as the difference between a finite collection of function values $f(\mathbf{x}_i)$, and the continuous function, $f(\mathbf{x})$. The former can be viewed as a vector $\mathbf{f}$, and can be drawn from a distribution such as the one in Eq. (1.33), $\mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}, K(X, X))$. The latter is a *function*, drawn from a distribution *over functions*, where the mean $m(\mathbf{x})$ and covariances $k(\mathbf{x}, \mathbf{x}')$ are functions as well. A function $f(\mathbf{x})$ drawn from a Gaussian process is written as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{1.34}$$

Function Space View

Gaussian processes provide distributions over functions $\mathcal{GP}$. It is therefore useful to consider the problem in the function space view introduced in [5]. For a function $f(\mathbf{x})$ the Gaussian process mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ are defined as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{1.35}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))]. \tag{1.36}$$

In other words, the mean $m(\mathbf{x})$ is the expected value of the function $f(\mathbf{x})$ at $\mathbf{x}$, and the covariance function is the expected value of the simultaneous deviations of $f(\mathbf{x})$ from the mean $m(\mathbf{x})$ at $\mathbf{x}$, and $f(\mathbf{x}')$ from the mean $m(\mathbf{x}')$ at $\mathbf{x}'$. As mentioned, the mean and covariance are now *functions* of the input vector $\mathbf{x}$. This means that for every input vector $\mathbf{x}$, there is a Gaussian distribution over function values with a mean $m(\mathbf{x})$ and covariance given by $k(\mathbf{x}, \mathbf{x}')$. This is a generalization of the single test point in Fig. 1.5, where every point $\mathbf{x}^*$ gets a similar distribution.

The collection of Gaussian distributions over functions that are now a function of the input vector $\mathbf{x}$, are the Gaussian processes, $\mathcal{GP}$. In the same way that a random variable is drawn from a distribution, random functions can be drawn from the $\mathcal{GP}$,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{1.37}$$

How functions are drawn from distributions is discussed in more detail in a later section.

The covariance between two points $f(\mathbf{x})$, $f(\mathbf{x}')$ is determined by the covariance function $k(\mathbf{x}, \mathbf{x}')$, as discussed in Sec. 1.2. Since the distribution $\mathcal{GP}$ is over function values, and so the covariance function calculates the covariance between the *function values*, $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, and *not* the input vectors, $\mathbf{x}_i$ and $\mathbf{x}_j$. Rather, the covariance between two function values is a function of the input vectors. Consider again the illustration in Fig. 1.4. The variation in function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$ depends on the distance between the points $\mathbf{x}_i$ and $\mathbf{x}_j$, and the characteristic length scale of the process. If the length scale is large, the two input vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ can be far away, and still have similar function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$. For short length scales, however, nearby points can have very different function values, because the function varies rapidly.

Distributions over Functions

As metioned above, functions can be drawn from the Gaussian process with mean $m(\mathbf{x})$ and covariance $k(\mathbf{x}, \mathbf{x}')$, $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. All functions $f(\mathbf{x})$ that are drawn from this distribution must have a covariance between function values $f(\mathbf{x})$, $f(\mathbf{x}')$ determined by the covariance function $k(\mathbf{x}, \mathbf{x}')$. If for example the covariance function has a large length scale, quickly varying functions are not allowed. The mean of the distribution, $m(\mathbf{x})$, is *not* the mean value of each function $f(\mathbf{x})$ drawn from $\mathcal{GP}$, but rather the mean function value you would get in $\mathbf{x}$ if you drew enough functions from $\mathcal{GP}$. Drawing functions from a distribution in this way will be referred to as *drawing samples*.

Gaussian processes are Bayesian in that there is a prior and a posterior distribution over functions, where the posterior is obtained by conditioning the prior on the training data. Consider the $n^* \times m$-matrix of test points X^*, containing n^* test points $\mathbf{x}_i^*$, with unknown function values $f(\mathbf{x}^*)$. Using the kernel on the matrix X^* gives the covariance matrix, as discussed in Sec. 1.2. The covariance

[3]The mean does not have to be zero, it could for example be the mean of the training data.

matrix $K(X^*, X^*)$ now contains the covariance of all test points $\mathbf{x}_i^*$, calculated using the covariance function $k(\mathbf{x}_i^*, \mathbf{x}_j^*)$. Combined with an initial mean of zero [3] one obtains the *prior* distribution of predicted target values $\mathbf{f}^*$

$$\mathbf{f}^* \sim \mathcal{N}(\mathbf{0}, K(X^*, X^*)). \tag{1.38}$$

This distribution contains the prior assumptions about the function values $f(\mathbf{x})$, in that the smoothness of the function and the correlation between function values are encoded in the covariance matrix. This is the prior probability distribution discussed in Sec. 1.1.2, that will be modified by the data to provide the posterior probability distribution. The choice of kernel is therefore one of the most important steps in learning with GPs.

Noise-Free Model

Now consider the addition of training data to the prior distribution in Eq. 1.38, in the form of a noise-free set of n training points $\{\mathbf{x}_i, y_i\}$, so that $y = f(\mathbf{x})$. The input vectors $\mathbf{x}_i$ form the $n \times m$-matrix X, where the rows are the input vectors. The training targets y_i form the corresponding n-dimensional vector $\mathbf{f}$. The test points are still contained in the $n^* \times m$ matrix X^*. The goal is to predict an n^*-dimensional vector $\mathbf{f}^*$ containing the predictions of the function values at the points $\mathbf{x}_i^*$, which is conditioned on the known function values $\mathbf{f}$.

The joint distribution of training outputs, $\mathbf{f}$, and test outputs, $\mathbf{f}^*$, according to the prior in Eq. 1.38 is

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X,X^*) & K(X^*,X^*) \end{bmatrix}\right), \tag{1.39}$$

where, as before, $K(X_i, X_j')$ is the covariance matrix between the sets of points $\{\mathbf{x}_i\}$ and $\{\mathbf{x}_j'\}$ calculated using the covariance function $k(\mathbf{x}_i, \mathbf{x}_j')$. By conditioning the distribution of $\mathbf{f}^*$ on the observations $\mathbf{f}$, the posterior distribution over $\mathbf{f}^*$ is obtained[4] [5]

$$\mathbf{f}^* \big| X^*, X, \mathbf{f} \sim \mathcal{N}(K(X^*,X)K(X,X)^{-1}\mathbf{f}, \tag{1.40}$$
$$K(X^*,X^*) - K(X^*,X)K(X,X)^{-1}K(X,X^*)), \tag{1.41}$$

where $\mathbf{f}^*|X^*, X, \mathbf{f}$ means the target values $\mathbf{f}^*$ for the features X^*, given the known features X and targets $\mathbf{f}$. The prior in Eq. 1.38 has now been modified by the data X, $\mathbf{f}$ to give the posterior in Eq. 1.40, leaving two distributions from which function values can be drawn.

[4]For more details, see Appendix A.2 in [5].

Drawing Samples

To generate samples $\mathbf{f} \sim \mathcal{N}(\mathbf{m}, K)$ with mean $\mathbf{m}$ and covariance matrix K using a scalar Gaussian generator[5], one proceeds as follows: first the Cholesky decomposition — also known as the matrix square root — L of the covariance matrix is found using $K = LL^T$. L is a lower triangular matrix. A vector $\mathbf{u}$ is then generated by multiple calls to the scalar Gaussian generator $\mathbf{u} \sim \mathcal{N}(0, I)$. Then $\mathbf{f} = \mathbf{m} + L\mathbf{u}$ has the desired distribution with mean $\mathbf{m}$ and covariance $L\mathbb{E}[\mathbf{u}\mathbf{u}^T]L^T = LL^T = K$ [5].

Using the method described above, random functions have been drawn from the prior and posterior in Eq. 1.38 and Eq. 1.40, respectively, and shown in Fig. 1.6. As discussed, the samples drawn from the prior have mean equal to zero $m(\mathbf{x}) = \mathbf{0}$, and constant covariance, $K(X^*, X^*)$, meaning that if you drew enough functions the mean of all function values at every $\mathbf{x}$ would be zero. The prior is shown in the upper panel of Fig. 1.6, where the mean of the distribution is represented by the thick, black line, and the covariance is the light blue band around the mean. In the posterior distribution the mean values and covariances have been modified by the training data, represented by red dots in the lower panel of Fig. 1.6. In a point where there is training data the uncertainty is zero[6], and so all samples drawn from the posterior distribution must pass through this point. Far away from training points the covariance is large. The mean has also been modified to pass through the training points, as seen by the thick black line in the lower panel.

Gaussian Noise Model

Noise-free observations are rare. In most cases targets will contain some noise $y = f(\mathbf{x}) + \varepsilon$, where the noise ε is assumed to follow a Gaussian distribution $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. This is the *Gaussian noise model.* As discussed in Sec. 1.2 the covariance of a function with Gaussian noise can be expressed as

$$\text{cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \delta_{ij}. \tag{1.42}$$

Training targets are now contained in the n-dimensional vector $\mathbf{y}$, while training features are contained in the $n \times m$-matrix X, test features in the $n^* \times m$-matrix X^* and predicted targets in the n^*-dimensional vector $\mathbf{f}^*$, as before. With the addition of the noise the prior distribution becomes

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 \mathbb{1} & K(X, X^*) \\ K(X, X^*) & K(X^*, X^*) \end{bmatrix}\right). \tag{1.43}$$

[5]A scalar Gaussian generator generates random numbers from a Gaussian distributions, and can be found in most programming enviroments, such as the `random` environment in `Python`.

[6]Assuming there is no noise in the data.

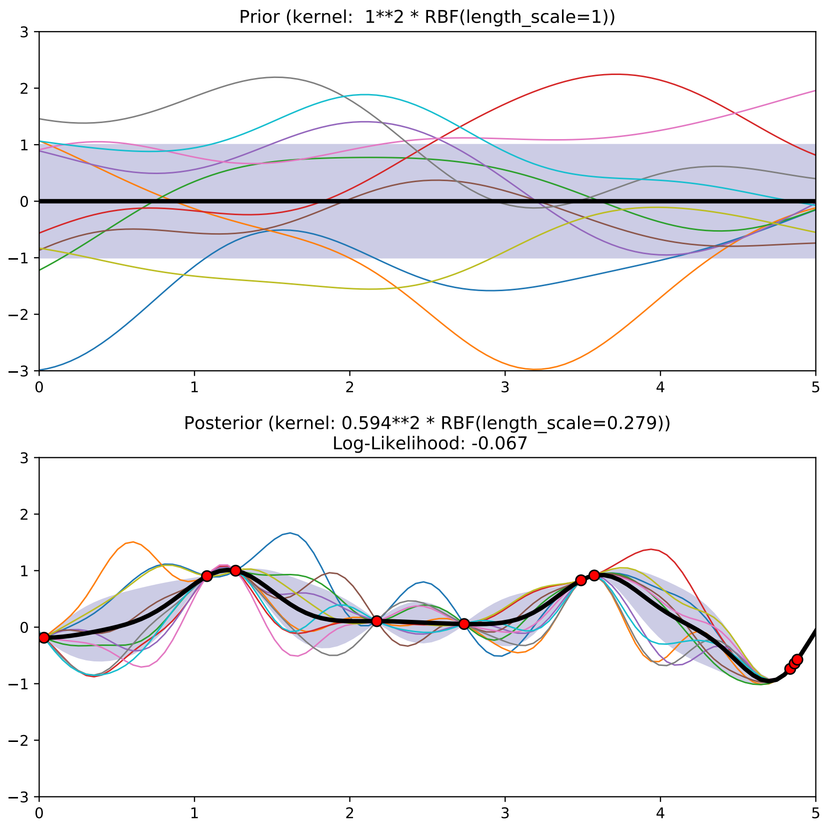


Figure 1.6: Drawing functions from the prior (top) and posterior (bottom) distributions. The thick, black line represents the mean of the distribution, while the shaded, blue area is the variance. The multiple colored lines are functions drawn randomly from the prior and posterior distributions, whose correlation are dictated by the covariance function. The prior has mean 0 and covariance given by the squared exponential function. The posterior has been modified by training points (red dots), giving rise to zero uncertainty at the points where training data exists, and an altered mean value for the distribution. The kernel of the prior distribution has hyperparameters $\sigma_f = 1$ and $\ell = 1$, while for the posterior they are $\sigma_f = 0.594$ and $\ell = 0.279$. Figure generated using scikit-learn.

The conditioned distribution is then

$$\mathbf{f}^* \big| X^*, X, \mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}^*, \mathrm{cov}(\mathbf{f}^*)) \tag{1.44}$$

where

$$m(\mathbf{f}^*) = K(X^*, X)[K(X,X) + \sigma_n^2 \mathbb{1}]^{-1}\mathbf{y}, \tag{1.45}$$
$$\mathrm{cov}(\mathbf{f}^*) = K(X^*, X^*) - K(X^*, X)[K(X,X) + \sigma_n^2 \mathbb{1}]^{-1} K(X, X^*). \tag{1.46}$$

Equations (1.45)-(1.46) are the key predictive equations for Gaussian process regression. The calculation requires the inverting the $n \times n$-matrix $[K(X,X) + \sigma_n^2 \mathbb{1}]$, which for large n becomes computationally unviable. This is discussed further in Sec. **??**, where Distributed Gaussian processes are proposed as a way of scaling Gaussian processes to larger training sets.

For the sake of tidying up the expression define the matrix $K \equiv K(X,X)$ and the matrix $K^* \equiv K(X, X^*)$. In the case of a single test point $\mathbf{x}^*$ the matrix K^* is written as a vector $\mathbf{k}(X, \mathbf{x}^*) = \mathbf{k}^*$ to denote the covariances between the n training points and the test point $\mathbf{x}^*$. Using this compact notation the GP prediction of $f^* = f(\mathbf{x}^*)$ for a single test point $\mathbf{x}^*$ is

$$m(f^*) = \mathbf{k}^{*T}(K + \sigma_n^2 \mathbb{1})^{-1}\mathbf{y}, \tag{1.47}$$
$$\mathbb{V}[f^*] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^{*T}(K + \sigma_n^2 \mathbb{1})^{-1}\mathbf{k}^*. \tag{1.48}$$

Note that, as mentioned in the beginning of Sec. 1.3, the predicted mean value $\bar{f}^*$ can be viewed as a linear combination of y_i of the form $\alpha \mathbf{y}$, where $\alpha = \mathbf{k}^{*T}(K + \sigma_n^2 \mathbb{1})^{-1}$. α then only contains the covariance between features.

Eqs. (1.47)-(1.48) form the basis for GP prediction in `scikit-learn` [4]. The algorithm for Gaussian process regression on a single test point $\mathbf{x}^*$, with training data X, $\mathbf{y}$ is outlined in Algorithm 1. The algorithm uses the Cholesky decomposition, L, of the covariance matrix to find the weights $\boldsymbol{\alpha}$ used to calculate the predictive mean f^*. The variance $\mathbb{V}[\mathbf{x}^*]$ is calculated using L and the covariance of the test point with the training points, $\mathbf{k}^* = k(X, \mathbf{x}^*)$.

1.4 Model Selection

The choice of kernel and hyperparameters is important for the quality of the GP prediction. Model selection means finding the kernel and corresponding hyperparameters that best fit the data. This is here referred to as *training* in machine learning. In this section Bayesian model selection is quickly overviewed, and the log marginal likelihood and cross validation are considered.

```
Data: $X$ (inputs), $\mathbf{y}$ (targets), $k$ (covariance function/kernel), $\sigma_n^2$ (noise
      level), $\mathbf{x}_*$ (test input).
L = Cholesky decomposition $(K + \sigma_n^2 I)$ ;
$\boldsymbol{\alpha} = (L^T)^{-1}(L^{-1}\mathbf{y})$ ;
$\bar{f}_* = \mathbf{k}_*^T \boldsymbol{\alpha}$ ;
$\mathbf{v} = L^{-1}\mathbf{k}_*$ ;
$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^T\mathbf{v}$ ;
$\log p(\mathbf{y}|X) = -\frac{1}{2}\mathbf{y}^T\boldsymbol{\alpha} - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi$ ;
Result: $f_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y}|X)$ (log marginal likelihood).
```

Algorithm 1: Algorithm 2.1 from [5].

1.4.1 Log Marginal Likelihood

The *marginal likelihood* can be used to find the optimal hyperparameters of covariance functions. Gaussian process regression models with Gaussian noise have the wonderful trait of analytically tractable integrals for the marginal likelihood, $P(\mathbf{y}|X, \boldsymbol{\theta})$. Note that the marginal likelihood here looks somewhat different than in Eq. (1.4) — $\boldsymbol{\theta}$ are the hyperparameters of the covariance functions, $\mathbf{y}$ are the predicted outputs and X are the training inputs. The parameters in Eq. (1.4), denoted Θ, are now the training outputs, $\mathbf{f}$, and so the marginal likelihood is the integral of the likelihood times the prior over the outputs $\mathbf{f}$. The marginal likelihood is then given by

$$P(\mathbf{y}|X, \boldsymbol{\theta}) = \int P(\mathbf{y}|\mathbf{f}, X, \boldsymbol{\theta}) P(\mathbf{f}|X, \boldsymbol{\theta}) d\mathbf{f}, \tag{1.49}$$

where $\mathbf{f}$ are the training outputs, $P(\mathbf{f}|X, \boldsymbol{\theta})$ is the prior and $P(\mathbf{y}|\mathbf{f}, X, \boldsymbol{\theta})$ is the likelihood. Under the Gaussian process model the prior is a Gaussian, $\mathbf{y} \sim \mathcal{N}(\mathbf{0}, K + \sigma_n^2 \mathbb{1})$, so the logarithm of the marginal likelihood in Eq. (1.49) gives the *log marginal likelihood* (LML) [5]

$$\log P(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T (K + \sigma_n^2 \mathbb{1})^{-1}\mathbf{y} - \frac{1}{2}\log\left|K + \sigma_n^2 \mathbb{1}\right| - \frac{n}{2}\log 2\pi, \tag{1.50}$$

where n is the number of training points. Each of the terms has an interpretation: $-\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2 \mathbb{1})^{-1}\mathbf{y}$ is the only term involving the data, and is therefore the data-fit; $-\frac{1}{2}\log\left|K + \sigma_n^2 \mathbb{1}\right|$ is the complexity penalty depending only on the covariance function and the inputs; and $-\frac{n}{2}\log 2\pi$ is a normalization term.

As mentioned, the goal is to use the marginal likelihood to determine the optimal hyperparameters of the covariance function, $\boldsymbol{\theta}$. As the LML in Eq. (1.49) is really the probability of the test outputs, $\mathbf{y}$, maximizing the LML will give the best estimate, as discussed in Sec. 1.1.3. The optimal hyperparameters are there-

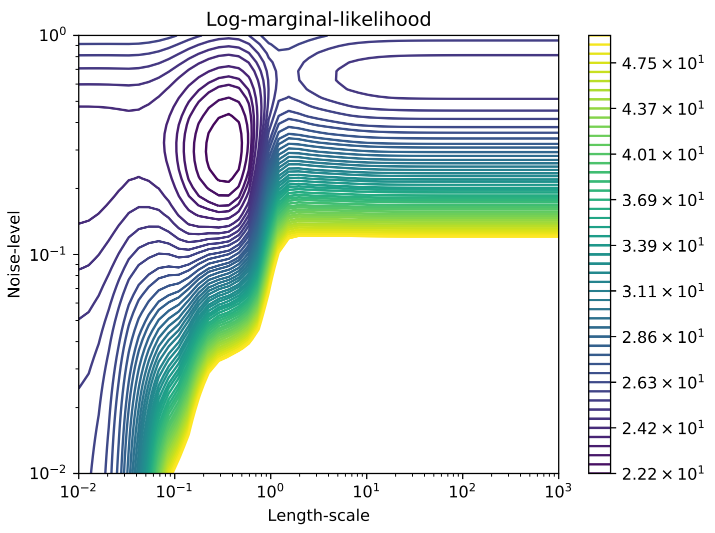


Figure 1.7: A contour plot of the log marginal likelihood with two local optima for a Gaussian process with kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-(\mathbf{x}_i - \mathbf{x}_j)^2/\ell^2) + \sigma_n^2 \delta_{ij}$. The rightmost optima favours a short length scale and low noise, with $\sigma_f = 0.64$, $\ell = 0.365$ and $\sigma_n^2 = 0.29$, while the leftmost favors a high noise level and therefore several large length scales, with $\sigma_f = 0.00316$, $\ell = 109$ and $\sigma_n^2 = 0.6$. The optimum to the right has LML -21.8 and the optimum to the right has LML -23.87. Plots were generated using scikit-learn.

fore found by maximizing the LML, which requires finding the partial derivatives

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{\partial K}{\partial \theta_j}). \tag{1.51}$$

Using partial derivatives, or the gradients, to find the optimal hyperparameters is called a *gradient based optimizer.* Computing the inverse of a matrix, K^{-1}, is computationally complex, and for n training points goes as $\mathcal{O}(n^3)$. Once this is done, however, finding the partial derivatives only requires complexity $\mathcal{O}(n^2)$, and so gradient based optimizers are advantageous.

The LML can have several local optima, as seen in the contour plot of the log marginal likelihood in Fig. 1.7. These correspond to different interpretations of the data. The leftmost optimum in Fig. 1.7, for example, favors a small length scale and smaller noise level. This means that little of the data is considered to be noise. The rightmost optimum has a higher noise level, and allows for a range of length scales, as most of the data is considered to be noise. Features with very large length scales are considered superfluous, as the function value depends little on them. To avoid ending up in a local optima, it can be wise to restart the optimizer a few times during learning.

1.4.2 Cross Validation

Cross validation is a means of monitoring the performance of a model. In k-fold validation this is done by dividing the data into k subsets and using $k-1$ folds to train the model, and a single fold to validate it. This is repeated k times, one for each possible validation set. Cross-validation requires a scoring function, such as the R^2 score. The R^2-score is given by

$$R^2 = 1 - \frac{\sum_{i=0}^{N-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{N-1}(y_i - \bar{y})^2}, \tag{1.52}$$

where $\hat{y}_i$ is the predicted value of the ith sample, y_i is the true value and $\bar{y} = \frac{1}{N}\sum_{i=0}^{N-1} y_i$ for N samples. This is the score used for cross validation in this project.

Cross-validation can be used to plot *learning curves*, which are used to find out whether the estimator benefits from adding more data. The learning curve plots the *training score* and *validation score*. The training score is the R^2-score for the prediction of the training points, *i.e.* the training points are also used as test points. The validation score is the R^2 score for the prediction of validation points, which are not a part of the training data. The test- and validation scores are used to find out if the model is *overfitting* or *underfitting*. *Overfitting* means that the model is a perfect fit to the training data, but predicts poorly for test data because it is not general. *Underfitting* occurs when the model is not able to capture the underlying structure of the data. Ideally, the training score should always be 1, and the validation score should approach 1 with the addition of data.

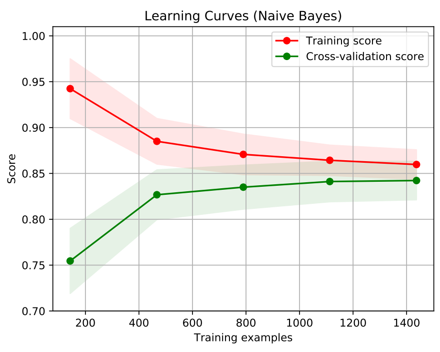
Examples of learning curves are shown in Fig. 1.8 as a function of training points, for two machine learning techniques called Naive Bayes and Support Vector Machine for an example problem. In **(a)** both the training score and cross-validation score tend to a value below 1, which indicates underfitting. This model will not benefit from more data. The example in **(b)** shows a training score of approximately 1, and a cross validation score that converges towards 1. This model could benefit from more data.

1.4.3 Relative Deviance

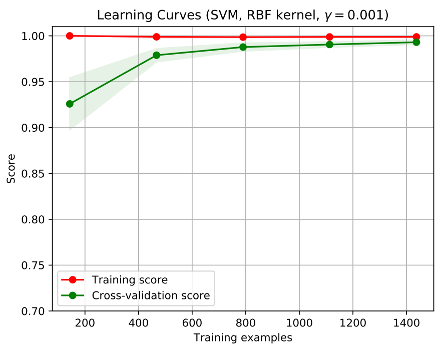
In this project predictions are compared using the *relative deviance.* For true values y_i and values predicted by the estimator $\hat{y}_i$ the relative deviance is given by

$$\varepsilon_i = \frac{y_i - \hat{y}_i}{y_i}. \tag{1.53}$$

In this project the target values have a very wide span, ranging from about 10^{-30} fb to 10^{9} fb. The data is therefore divided into decades, meaning one set contains



(a) Underfitting.



(b) Good fit.

Figure 1.8: Learning curves for two different estimators. The training scores are shown as red lines, with uncertainty bands in light red. The validation scores are shown as green lines, with uncertainty bands in light green. The estimator in **(a)** is underfitting, as both the training and validation tend to a value less than one. The estimator in **(b)** is a good fit, and could benefit from more data.

$\sigma \in [10^i, 10^{i+1}]$, where σ is the cross section. Then a distribution over the relative deviances within each decade is found, with a mean value and variance. These are plotted as a function of i, and denoted

$$m(\varepsilon_i) = m\left(\frac{y_i - \hat{y}_i}{y_i}\right), \tag{1.54}$$

$$\sigma(\varepsilon_i) = \mathbb{V}\left(\frac{y_i - \hat{y}_i}{y_i}\right). \tag{1.55}$$

Bibliography

[1] Mr. Bayes and Mr Price. An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfrs. *Philosophical Transactions (1683-1775)*, pages 370–418, 1763.

[2] Marc Peter Deisenroth and Jun Wei Ng. Distributed gaussian processes. *arXiv preprint arXiv:1502.02843*, 2015.

[3] Pierre Simon Laplace. *Théorie analytique des probabilités*. Courcier, 1820.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

[6] Devinderjit Sivia and John Skilling. *Data analysis: a Bayesian tutorial*. OUP Oxford, 2006.