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, $P(X|I)$ is a normalization constant called the *marginal likelihood* or evidence, and $P(X|\Theta,I)$ and $P(\Theta|I)$ are the *likelihood* and *prior*, respectively. In other words, Eq. (1.4) states the probability of the parameters Θ given the knowledge of outcomes X.

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.

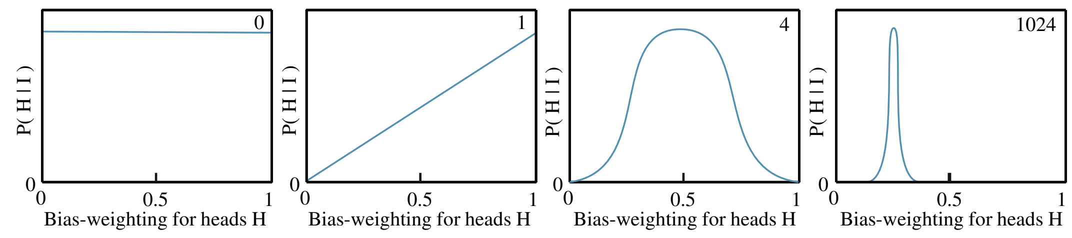


Figure 1.1: The posterior probability distribution of the bias-weighting of a coin for the uniform prior, $P(H|I)$. The first panel from the left is before the coin is tossed, the second panel is after 1 toss, the third is after 4 tosses, and the fourth is after 1024 tosses. The posterior converges towards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

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. 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

$$\frac{dP}{dX}\bigg|_{X_0} = 0, \qquad\qquad \frac{d^2P}{dX^2}\bigg|_{X_0} < 0, \tag{1.5}$$

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 $P(X|\mathcal{D}, I)$

$$L = L(X_0) + \frac{1}{2}\frac{d^2L}{dX^2}\bigg|_{X_0}(X - X_0)^2 + ..., \qquad L = \log_e \Big[P(X|\mathcal{D}, I)\Big] \tag{1.6}$$

The first term, $L(X_0)$, is just a constant. From Eq. (1.5) 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. (1.6).

The Gaussian Distribution

Taking the exponential of Eq. (1.6) and ignoring higher order terms, the posterior can then be approximated as

$$P(X|\mathcal{D}, I) \approx A \exp\left[\frac{1}{2}\frac{d^2L}{dX^2}\bigg|_{X_0}(X - X_0)^2\right], \tag{1.7}$$

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

Equation (1.7) is now in the shape of a *Gaussian distribution*, given by

$$P(X|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(X - \mu)^2}{2\sigma^2}\right], \tag{1.8}$$

where μ and σ^2 are the two parameters of the Gaussian distribution mentioned above. 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

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

variance for the approximated Gaussian distribution of $P(X|\mathcal{D},I)$ are then given by

$$m(X) = X_0,\, \mathbb{V}(X) = \Big(-\frac{d^2L}{dX^2}\Big)^{-1/2}. \tag{1.9}$$

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 Fig. 1.2 an example of a Gaussian distribution $\mathcal{N}(\mu,\sigma^2)$ is shown.

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 $\mathcal{N}(m(X),\mathbb{V}(X))$,

$$m(X) \;:\; \text{Mean of } X, \tag{1.10}$$
$$\mathbb{V}(X) \;:\; \text{Variance of } X. \tag{1.11}$$

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

$$\mathbb{E}[X] = \int XP(X|I)dX \;:\; \text{Expectation value of } X. \tag{1.12}$$

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

$$\mathbb{V}(X) = \mathbb{E}\big[(X-X_0)^2\big] = \int\int (X-X_0)^2 P(X|\mathcal{D},I)dX. \tag{1.13}$$

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

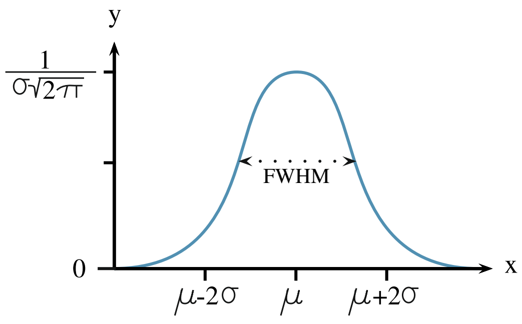


Figure 1.2: A Gaussian probability distribution. The maximum is at the mean value μ, with a full width at half maximum (FWHM) at around 2.35σ. Figure from [6].

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. (1.6). In the case of several random variables X_i, a set of *simultaneous equations* must be solved to get the best estimate

$$\frac{dP}{dX_i}\Big|_{X_{0j}} = 0, \qquad \frac{d^2P}{dX_i^2}\Big|_{X_{0j}} < 0 \tag{1.14}$$

To simplify expressions consider the problem in two dimensions, so that $\{X_i\} = (X, Y)$. Analogously to Eq. (1.6), the Taylor expansion of $L = \log_e \Big[P(X, Y|I)\Big]$ is found

$$\begin{aligned} L =& L(X_0, Y_0) + \frac{1}{2}\Big[\frac{d^2L}{dX^2}\Big|_{X_0,Y_0}(X - X_0)^2 \\ &+ \frac{d^2L}{dY^2}\Big|_{X_0,Y_0}(Y - Y_0)^2 + 2\frac{d^2L}{dXdY}\Big|_{X_0,Y_0}(X - X_0)(Y - Y_0)\Big] + ... \end{aligned} \tag{1.15}$$

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. (1.15) in matrix form gives

$$Q = \begin{pmatrix} X - X_0 & Y - Y_0 \end{pmatrix} \begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} X - X_0 \\ Y - Y_0 \end{pmatrix}, \tag{1.16}$$

where the matrix elements are

$$A = \frac{\partial^2 L}{\partial X^2}\Big|_{X_0,Y_0}, \qquad B = \frac{\partial^2 L}{\partial Y^2}\Big|_{X_0,Y_0}, \qquad C = \frac{\partial^2 L}{\partial X \partial Y}\Big|_{X_0,Y_0}. \tag{1.17}$$

The expression for the variance of X for the distribution $P(X, Y|I)$ is very similar to Eq. (1.13), except for an additional integral over the random variable Y

$$\mathbb{V}(X) = \sigma_X^2 = \mathbb{E}\big[(X - X_0)^2\big] = \int\int (X - X_0)^2 P(X, Y|\mathcal{D}, I) dX dY. \qquad (1.18)$$

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

$$\sigma_{XY}^2 = \mathbb{E}\big[(X - X_0)(Y - Y_0)\big] = \int\int (X - X_0)(Y - Y_0) P(X, Y|\mathcal{D}, I) dX dY. \qquad (1.19)$$

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]

$$\text{cov}(X, Y) = \begin{pmatrix} \sigma_X^2 & \sigma_{XY}^2 \\ \sigma_{XY}^2 & \sigma_Y^2 \end{pmatrix} = - \begin{pmatrix} A & C \\ C & B \end{pmatrix}^{-1}. \qquad (1.20)$$

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 $\sigma_{X_i X_j}^2$, 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

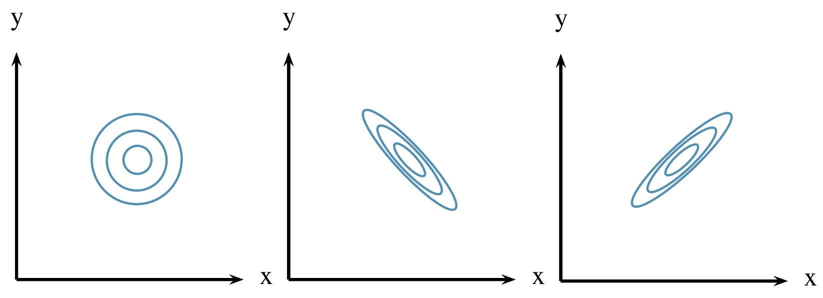


Figure 1.3: A schematic illustration of covariance and correlation. (a) The contours of a posterior pdf with zero covariance, where the inferred values of X and Y are uncorrelated. (b) The corresponding plot when the covariance is large and negative; (c) The case of positive correlation.

$\mathbf{x} \in \mathcal{X}$, $\mathbf{x}' \in \mathcal{X}$ 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

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{1.21}$$

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

$$k_{SE}(r) = \exp\Big(-\frac{r^2}{2\ell^2}\Big), \tag{1.22}$$

[2]Invariant to rigid rotations in input space.

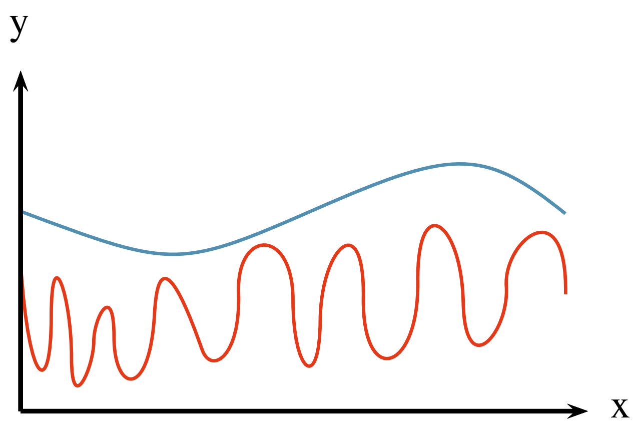


Figure 1.4: The effect of varying the length scale ℓ. A long length scale (blue) gives a smooth, slowly varying function, while a short length scale (red) gives a more staccato, quickly varying function.

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

The *Matérn class of covariance functions* is given by

$$k_{Matérn}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big)^{\nu} K_{\nu}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big), \tag{1.23}$$

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

$$k_{\nu=p+\frac{1}{2}} = \exp\Big(-\frac{\sqrt{2\nu}r}{\ell}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu}r}{\ell}\Big)^{p-i}. \tag{1.24}$$

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

$$k_{\nu=3/2}(r) = \Big(1+\frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{1.25}$$

$$k_{\nu=5/2}(r) = \Big(1+\frac{\sqrt{5}r}{\ell}+\frac{5r^2}{3\ell^2}\Big)\exp\Big(-\frac{\sqrt{5}r}{\ell}\Big). \tag{1.26}$$

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

$$k(\mathbf{x}_i, \mathbf{x}_j)_{noise} = C\delta_{ij}, \tag{1.27}$$

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

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j))^T M(\mathbf{x}_i - \mathbf{x}_j) + \sigma_n^2 \delta_{ij}, \tag{1.28}$$

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

$$M_1 = \ell^{-2}\mathbb{I}, \qquad\qquad M_2 = \mathrm{diag}(\boldsymbol{\ell})^{-2}. \tag{1.29}$$

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

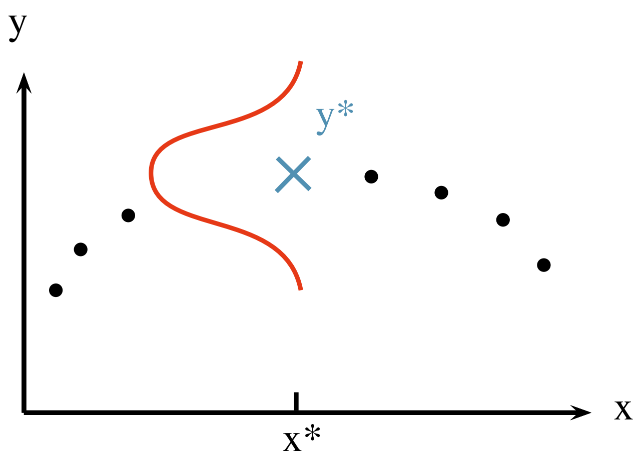


Figure 1.5: An illustration of a GP prediction of the target value y^* (blue cross), given the known set of points $\{x_i, y_i\}$ (black dots). The prediction is a Gaussian distribution in y with mean y^* and variance σ^2. The Gaussian distribution is drawn in red with y on the vertical axis, with uncertainty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

Before delving into the details of Gaussian process regression, it is wise to introduce some notation, as Gaussian process regression is notoriously confusing.

As discussed in Sec. 1.3, the Gaussian distribution is denoted $\mathcal{N}(\mu, \sigma^2)$, for a mean value μ and variance σ^2. The Gaussian 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*. For a multivariate distribution over f_i, $i = 1, ..., n$, the random variables form an n-dimensional vector $\mathbf{f}$. The mean values, μ_i, are then contained in the vector $\bar{\mathbf{f}}$, characterized by a bar. 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

$$\mathbf{f} \sim \mathcal{N}\Big(\bar{\mathbf{f}},\ \text{cov}(\mathbf{f}, \mathbf{f})\Big). \tag{1.30}$$

Most times there will be a set of features $\{\mathbf{x}_i\}$, and corresponding targets $\{y_i\}$. For n training points $\mathbf{x}_i$, where $\mathbf{x}_i$ is an m-dimensional vector, the features comprise the $n \times m$-matrix X. The targets comprise the corresponding n-dimensional vector $\mathbf{y}$. A central assumption in Gaussian processes is that the covariance between targets y_i and y_j is given by the covariance function k used on their features

$$\text{cov}(y_i, y_j) = k(\mathbf{x}_i, \mathbf{x}_j). \tag{1.31}$$

The distribution over target values $\mathbf{f}$ will therefore be written

$$\mathbf{f} \sim \mathcal{N}\Big(\ \bar{\mathbf{f}}, K(X, X)\ \Big), \tag{1.32}$$

where $K(X, X)$ is the covariance matrix containing the covariances of the features contained in X, decided by 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 set of function values $f(\mathbf{x}_i)$, and a 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. (1.32), $\mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}, K(X, X))$. 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 drawn from a Gaussian process is written as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{1.33}$$

Function Space View

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

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{1.34}$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \tag{1.35}$$

where $\mathbb{E}[a]$ is the expectation value of some quantity a. The mean $m(\mathbf{x})$ and covariance given by the covariance function $k(\mathbf{x}, \mathbf{x}')$ are now functions of the input vector $\mathbf{x}$. This notation 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 $\mathbf{x}_i^*$ gets a similar distribution.

Given the mean and covariance, the Gaussian distribution for $f(\mathbf{x})$ is completely specified. The collection of Gaussian distributions that are now a function of the input vector $\mathbf{x}$, are the Gaussian processes, denoted $\mathcal{GP}$. In the same way that you can draw a random variable A from a distribution over A, $P(A|I)$, you can draw random functions from the distribution over functions $\mathcal{GP}$,

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{1.36}$$

The covariance between two points $k(\mathbf{x}_i, \mathbf{x}_j)$ is decided by the covariance function, as mentioned in Sec. 1.2. In this text the running example will be the squared exponential (SE) kernel with characteristic length scale ℓ, given by

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\frac{1}{2}|\mathbf{x}_i - \mathbf{x}_j|^2\Big). \tag{1.37}$$

Note that 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, or correlation, between two function values is a function of the input vectors. Consider again the illustration in Fig. 1.4. The variation in two 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

Specifying a mean $m(\mathbf{x})$ and a covariance function $k(\mathbf{x}, \mathbf{x}')$ specifies a distribution over functions, $\mathcal{GP}$ [5]. As metioned above, functions can be drawn from this distribution. All functions $f(\mathbf{x})$ that are drawn from this distribution must have a covariance decided by the covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. 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 sometimes be referred to as *drawing samples.*

Consider a set of n^* test points $\mathbf{x}_i^*$, where each point is m-dimensional. These can be combined into a $n^* \times m$-matrix X^*, where each row is a test point $\mathbf{x}_i^*$. Using the kernel on the matrix X^*, gives the covariance matrix, as discussed in Sec. 1.2. The covariance matrix $K(X^*, X^*)$ now contains the covariance of all

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

test points $\mathbf{x}_i^*$, found using the kernel $k(\mathbf{x}_i^*, \mathbf{x}_j^*)$. Combined with an initial mean of zero ^{3} one obtains the *prior* distribution

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

This distribution contains the prior assumptions about the function values $f(x)$, 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. So the choice of kernel is one of the most important steps in learning with GPs. In Fig. 1.6 samples are drawn from both the prior and posterior distribution.

Noise-Free Model

Consider a simple example of a noise-free set of n training points $\{\mathbf{x}_i, y_i\}$, so that $y = f(\mathbf{x})$. The input vectors $\mathbf{x}_i$ are m-dimensional and form a $n \times m$-matrix X, where the rows are the input vectors. The training targets y_i form a corresponding n-dimensional vector $\mathbf{f}$. For a set of n^* test points $\{\mathbf{x}_i\}$ the $n^* \times m$-matrix X^* is constructed, as discussed above. The Gaussian processes will predict an n^*-dimensional vector $\mathbf{f}^*$ containing the predictions of the function values at the points $\mathbf{x}_i^*$.

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

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

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, the posterior distribution over $\mathbf{f}^*$ is obtained[4] [5]

$$\begin{aligned} \mathbf{f}_* \big| X_*, X, \mathbf{f} \sim \mathcal{N}(&K(X_*, X)K(X,X)^{-1}\mathbf{f}, &(1.40)\\ &K(X_*, X_*) - K(X_*, X)K(X,X)^{-1}K(X, X_*)). &(1.41) \end{aligned}$$

Drawing Samples

To generate samples $\mathbf{f} \sim \mathcal{N}(\mathbf{m}, K)$ with an arbitrary mean $\mathbf{m}$ and covariance matrix K using a scalar Gaussian generator[5], one proceeds as follows: first the

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

[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`.

Cholesky decomposition, or matrix square root, L of K is found $K = LL^T$, where L is a lower triangular matrix. The a vector $\mathbf{u}$ is 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].

In the case of the prior, the samples are drawn from a distribution of functions with mean zero and constant variance, meaning that if you drew enough functions the mean over all function values at every $\mathbf{x}$ would be zero. For the posterior, the mean values and uncertainties have been modified by the training data. In a point where there is training data the uncertainty is zero [6], and so all samples drawn from this distribution must pass through this point. Far away from training points the variance is large.

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.* The covariance can then be expressed as

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

which gives for the prior distribution

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

The conditioned distribution is then

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

where

$$\begin{aligned} m(\mathbf{f}^*) &= K(X^*,X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1}\mathbf{y}, \\ \text{cov}(\mathbf{f}^*) &= K(X^*,X^*) - K(X^*,X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1} K(X,X^*) \end{aligned} \tag{1.45}$$

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 $\mathbf{k}(\mathbf{x}^*) = \mathbf{k}^*$. Using this compact notation the GP prediction for a single test point $\mathbf{x}^*$ is

$$m(f^*) = \mathbf{k}^{*T}(K + \sigma_n^2 \mathbb{I})^{-1}\mathbf{y}, \tag{1.46}$$

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

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

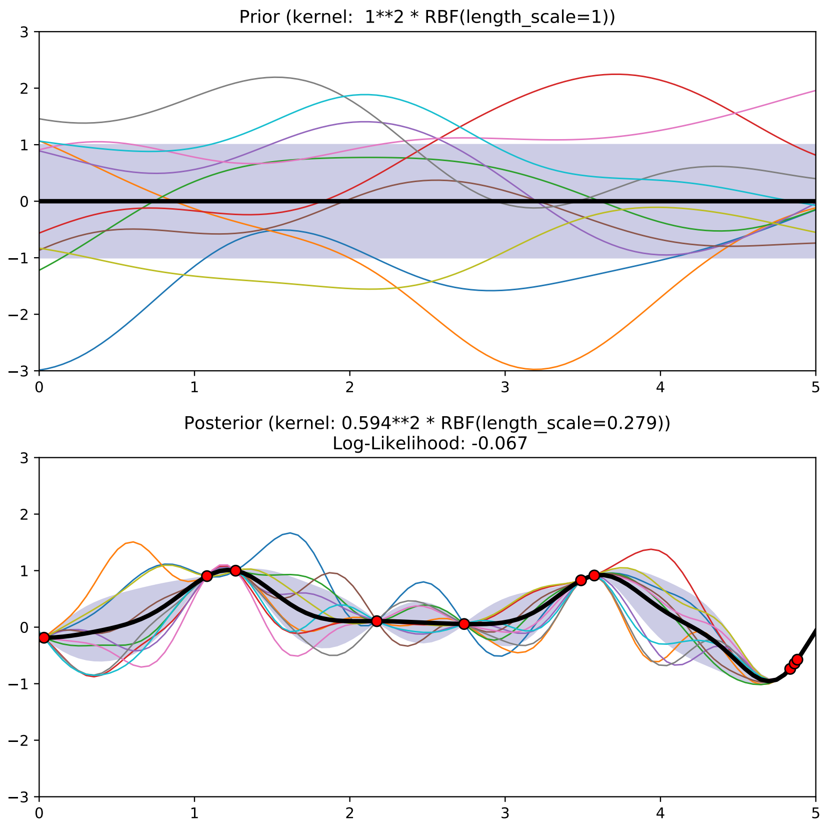


Figure 1.6: Drawing functions from the prior (top) and posterior (bottom) distributions. The thick, black line represents the mean of the distribution, while the shaded, blue area is the variance. The multiple colored lines are functions drawn randomly from the prior and posterior distributions, whose 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.

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

Eqs. (1.46)-(1.47) form the basis for GP prediction in `scikit-learn` [4]. The algorithm is shown in Algorithm (1), and uses the Cholesky decomposition of the covariance matrix.

```
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 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 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.

1.4.1 Bayesian Model Selection

A model has a set of model structures $\mathcal{H}_i$, hyperparameters $\boldsymbol{\theta}$ and parameters $\mathbf{w}$. Feature selection is done at all levels in a hierarchical way, by finding the posterior over *parameters*, the posterior over *hyperparameters* and the posterior for the *model.* Here only the posterior over parameters is considered [5],

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)}, \tag{1.48}$$

as it gives rise to the *marginal likelihood* $p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)$, given by

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w} \quad \text{(marginal likelihood)}. \tag{1.49}$$

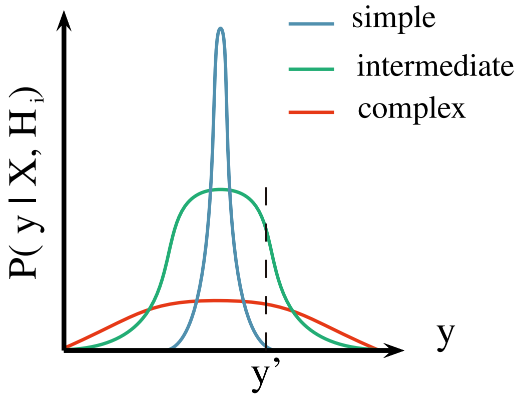


Figure 1.7: The marginal likelihood is the probability of the data, given the model. The number of data points n and inputs X are fixed. The horizontal axis represents an idealized set of all possible vectors of targets $\mathbf{y}$. Since the marginal likelihood is a probability distribution it must normalize to unity. For a particular set $\mathbf{y}'$, indicated by the dashed line, the intermediate model is preferred to the simple and complex ones.

Because of the complexity of integrals involved in model selection, it is common to maximize the marginal likelihood with respect to hyperparameters $\boldsymbol{\theta}$. This maximization is what distinguishes Bayesian model selection from other model selection schemes, as it incorporates a trade-off between model complexity and model fit. This means that a complex model will allow for several different kinds of models, but each of them will get a low probability. Meanwhile, simple models will only have a few possibilities, but each of these will have a large probability, see Fig. (1.7).

1.4.2 Log Marginal Likelihood

Gaussian process regression models with Gaussian noise have the wonderful trait of analytically tractable integrals for the marginal likelihood. The exact expression for the log marginal likelihood [7] can be shown to be [5]

$$\log p(\mathbf{y}|X,\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi. \tag{1.50}$$

Each of the terms has an interpretation: $-\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y}$ is the only term involving the data, and is therefore the data-fit; $-\frac{1}{2}\log|K_y|$ is the complexity penalty depending only on the covariance function and the inputs; and $-\frac{n}{2}\log 2\pi$ is a normalization term. The marginal likelihood is conditioned on the hyperparameters of the covariance function $\boldsymbol{\theta}$, and the optimal parameters are found by

[7]The logarithm is used as the marginal likelihood varies rapidly.

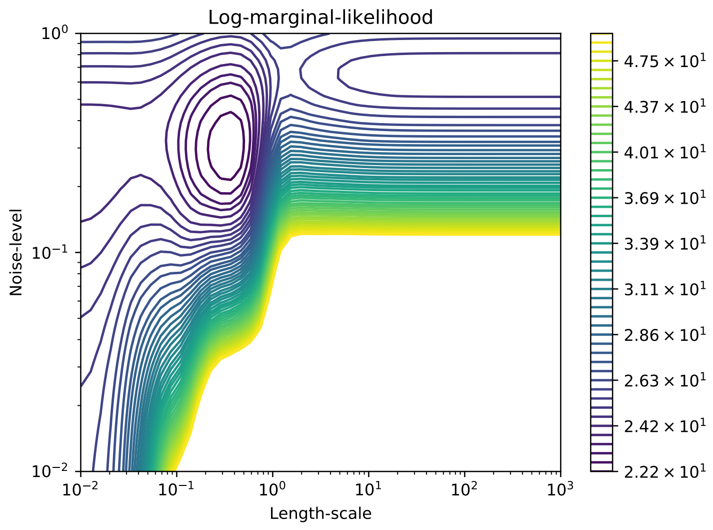


Figure 1.8: A contour plot of the log marginal likelihood with two local optima. The rightmost optima favours a short length scale and low noise, while the leftmost favors a high noise level and therefore several large length scales. Plot generated using scikit-learn.

maximizing. This requires the partial derivatives of the log marginal likelihood (LML)

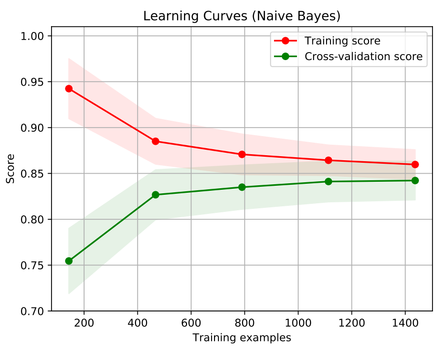
$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1}\mathbf{y} - \frac{1}{2}\text{tr}(K^{-1}\frac{\partial K}{\partial \theta_j}). \tag{1.51}$$

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.

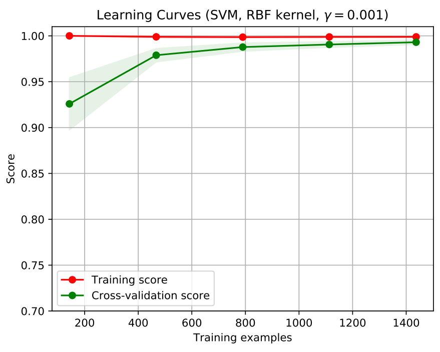
The LML can have several local optima, as seen in Fig. (1.8). These correspond to different interpretations of the data. The rightmost optima in Fig. (1.8) for example, favors a small length scale and smaller noise level. This means that it considers little of the data to be noise. The rightmost optimum has a higher noise level, and allows for several large length scales, as it considers most of the data to be noise. Features with very large length scales are considered superfluous, as the function value depends little on them. Because of this type of complication, it might be wise to restart the optimizer a few times during learning.

1.4.3 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



(a) Underfitting.



(b) Well fit.

Figure 1.9: Learning curves for two different estimators.

to train the model, and a single fold to validate it. This is repeated k times. Cross-validation requires a loss function, such as the mean relative deviance or the R^2 score. The latter 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 thesis.

Cross-validation can be used to plot learning curves, which is a tool to find out whether the model benefits from adding more data. The learning curve plots the training score and validation score 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.

Examples of learning curves are shown in Fig. (1.9) for Naive Bayes and SVM estimators [8]. 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.

1.4.4 Relative Deviance

In this project the main loss function used for comparing predictions is the relative deviance. For true values y_i and values predicted by the estimator $\hat{y}_i$ this is given

[8]Methods in Machine Learning.

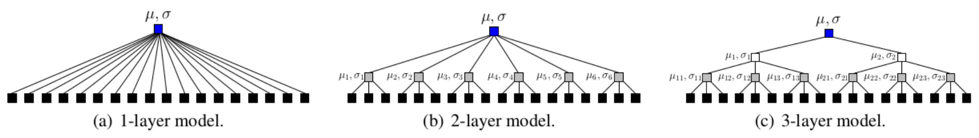


Figure 1.10: From [2].

by

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

The relative deviance is used because of the large span of the target values, ranging from about 10^{-30} to 10^9. The data is therefore divided into decades, meaning one set contains $\sigma \in [10^i, 10^{i+1}]$. 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.

1.5 Distributed Gaussian Processes

Limitations of Gaussian Processes

The biggest weakness of Gaussian processes is that they scale poorly with the size of the data set n. The training and predicting scale as $\mathcal{O}(n^3)$ and $\mathcal{O}(n^2)$, respectively, giving GP a practical limit of $\mathcal{O}(10^4)$.

In [2] a way of scaling GPs to large data sets is proposed, in the form of a robust Bayesian Comittee Machine (rBCM). This method is based on the product-of-experts and Bayesian Comittee Machine, and has the advantage of providing an uncertainty for the prediction.

1.5.1 Product-of-Experts

Product-of-expert (PoE) models are a way of parallelising large computations. They combine several independent computations on subsets of the total data, called 'experts'. In the case of distributed Gaussian processes each expert performs GP on a subset of the training data, and the predictions on a common test set are combined.

Consider the training data set $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$, which is partitioned into M subsets $\mathcal{D}^{(k)} = \{\mathbf{X}^{(k)}, \mathbf{y}^{(k)}\}$, $k = 1, ..., M$. Each GP expert does learning on its training data set $\mathcal{D}^{(k)}$, then predictions are combined at the parent node. This node could also be considered an expert for a PoE with several layers, see Fig. (1.10).

1.5.2 Algorithm

The marginal likelihood factorizes into the product of M individual terms because of the independence assumption [2]. The LML is then

$$\log p(\mathbf{y}^{(k)}|\mathbf{X}^{(k)}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{(k)}(\mathbf{K}_\psi^{(k)} + \sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y}^{(k)} - \frac{1}{2}\log|\mathbf{K}_\psi^{(k)} + \sigma_\varepsilon^2\mathbf{I}|, \tag{1.54}$$

where $a^{(k)}$ is the quantity corresponding to the kth expert. Computing the LML now entails inverting the $n_k \times n_k$ matrix $(\mathbf{K}_\psi^{(k)} + \sigma_\varepsilon^2\mathbf{I})$, which requires time $\mathcal{O}(n_k^3)$ and memory consumption $\mathcal{O}(n_k^2 + n_k D)$ for $\mathbf{x} \in \mathbb{R}^D$. For $n_k \ll N$, this reduces the computation time and memory use considerably, and allows for parallel computing.

Several methods for prediction are discussed in [2], but here only the robust Bayesian Comittee Machine is introduced. The PoE predicts a function value f^* at a corresponding test input $\mathbf{x}^*$ according to the predictive distribution

$$p(f^*|\mathbf{x}^*, \mathcal{D}) = \frac{\prod_{k=1}^M p_k^{\beta_k}(f^*|\mathbf{x}^*, \mathcal{D}^{(k)})}{p^{-1+\sum_k \beta_k}(f^*|\mathbf{x}^*)}. \tag{1.55}$$

This prediction scheme allows for much flexibility, as it can vary the importance of an expert. The combined predictive mean and variance are

$$\mu_*^{rbcm} = (\sigma_*^{rbcm})^2 \sum_k \beta_k \sigma_k^{-2}(\mathbf{x}_*)\mu_k(\mathbf{x}_*), \tag{1.56}$$

$$(\sigma_*^{rbcm})^{-2} = \sum_{k=1}^M \beta_k \sigma_k^{-2}(\mathbf{x}_*) + \big(1 - \sum_{k=1}^M \beta_k\big)\sigma_{**}^{-2}, \tag{1.57}$$

where the parameters β_k control the importance of the individual experts, but also the how strong the influence of the prior is. In the article, these are chosen according to the predictive power of each expert at $\mathbf{x}^*$. More specifically, β_k is the change in differential entropy between the prior $p(f^*|\mathbf{x}^*)$ and the posterior $p(f^*|\mathbf{x}^*, \mathcal{D}^{(k)})$, which can be calculated as

$$\beta_k = \frac{1}{2}(\log \sigma_{**}^2 - \log \sigma_k^2(\mathbf{x}^*)), \tag{1.58}$$

where σ_{**}^2 is the prior variance, and $\sigma_k^2(\mathbf{x}^*)$ is the predictive variance of the kth expert.

1.5.3 Implementing the Algorithm

The mean and variance in Eq. (1.56)-(1.57) were implemented in `Python` using the `scikit-learn` library's existing framework for regular Gaussian processes. The algorithm was parallelised, so that each expert can learn and predict in parallel, before being combined to the final prediction. Pseudocode for the implementation is found in Alg. (2).

For parallelisation the `scikit-learn` function `Parallel` from `joblib` was used, which runs Python functions as pipeline jobs. It uses the Python function `multiprocessing` as a backend. An example of usage with 3 parallel jobs is

```
>>> from joblib import Parallel, delayed
>>> from math import sqrt
>>> Parallel(n_jobs=3)(delayed(sqrt)(i**2) for i in range(10))
[0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0]
```

where `delayed` is a simple trick to be able to create a tuple with a function-call syntax.

```
Data: $N_{experts}$ (number of experts), $X$ (inputs), $\mathbf{y}$ (targets), $k$ (initial
      kernel), $\sigma_n^2$ (noise level), $\mathbf{x}^*$ (test input)
Split training data into $N$ subsets: $X_k, \mathbf{y}_k$;
for each expert do
    Fit GP to training data $X_k, \mathbf{y}_k$ ;
    Predict $\mu_*, \sigma_*^2$ for $\mathbf{x}^*$ using GP ;
    $\sigma_{**}^2 = k(x^*, x^*)$ ;
end
for each expert do
    $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$ ;
    $(\sigma_*^{rbcm})^{-2} += \beta\sigma^{-2} + \big(\frac{1}{n_{experts}} - \beta\big)\sigma_{**}^{-2}$
end
for each expert do
    $\mu_*^{rbcm} += (\sigma_*^{rbcm})^2 \beta \sigma_*^{-2} \mu_*$
end
Result: Approximative distribution of $f_* = f(\mathbf{x}_*)$ with mean $\mu_*^{rbcm}$ and
        variance $(\sigma_*^{rbcm})^2$.
```

Algorithm 2: Pseudocode for using rBCM on a single test point $\mathbf{x}_*$. For the fit and prediction of each expert GP Algorithm (1) is used.

1.5.4 Benchmark

The benchmark function for parallelised distributed Gaussian processes is

$$f(x_1, x_2) = 4x_1x_2,$$

where the vectors $\mathbf{x} = (x_1, x_2)$ were drawn from a random normal distribution using the `numpy` function `random.randn`. Gaussian processes implemented by

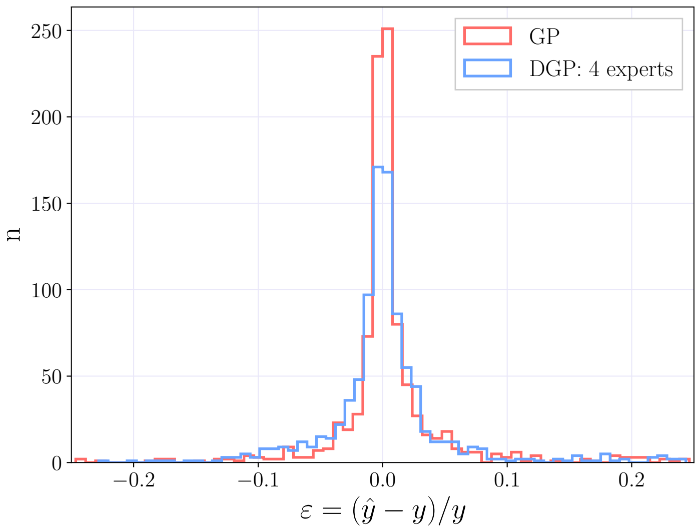


Figure 1.11: Histogram of the relative deviance between true value y and predicted value $\hat{y}$ for Gaussian process regression (GP) and Distributed gaussian process regression (DGP) for the function $f(x_1, x_2) = 4x_1x_2$.

`scikit-learn` in the function `GaussianProcessRegressor` were compared to distributed Gaussian processes with 4 experts. 2000 training points and 1000 test points were used, and the resulting times for the GP and DGP were

$$\text{Gaussian processes time: } 154.12 \text{ s} \tag{1.59}$$

$$\text{Distributed Gaussian processes time: } 5.61 \text{ s} \tag{1.60}$$

Histograms of the relative deviances for Gaussian processes (GP) and Distributed Gaussian processes (DGP) are found in Fig. (1.11).

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.