Contents

Chapter 1

Gaussian Processes

In this chapter Gaussian process regression is introduced and explained. To begin with some concepts and expressions in Bayesian statistics are introduced. The following section introduces the mathematical framework needed, before some covariance functions are discussed. A few concepts in Bayesian model selection are used as a basis to quantify 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

As it turns out, statistics are not merely statistics. In fact, there is great disagreement as to what statistics should be. The main opponents in this fight can be divided into two groups: the *frequenstist* statisticians and the *Bayesian* statisticians. In order to understand why they can't get along, first consider a statement they *do* agree on

Statisticians use probability to describe uncertainty.

This seems reasonable. Where statisticians reach disagreement is at the definition of the *uncertain*. Defining the uncertain defines probability, and at a high level one can distinguish between two branches, namely *objective* and *subjective* probability. Consider an example in which a statistician throws a dice. Before throwing, he is uncertain about the outcome of the dice toss. This uncertainty related to the outcome is the *objective uncertainty*: 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 dice toss. Is the dice loaded? Is one of the edges sharper than the others? This uncertainty is the *subjective uncertainty*, as it may vary from person to person depending on how much information one has about the system. One of the main critiques 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 it is helpful to introduce a soundingly Bayesian framework, namely *Bayes' theorem*. 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. $P(X|I)$ means the probability of outcome X given the information I, and $P(X|Y, I)$ means 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 probability of the outcome X happening plus the probability of X not happening is equal to 1. This is rather untuitive, considering an event either takes place or it doesn't. 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 already happened. These expressions can easily be combined into Bayes' theorem, first formulated by 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. So what is Bayesian about Bayes' theorem? *Nothing*. It merely reformulates the rules of logical consistent reasoning statet by Richard Cox in 1946 [4]. Laplace was the one to make Bayes' theorem Bayesian, when he used the theorem to perform inference about distribution parameters. These are, for example, the mean and variance of a Gaussian distribution. The resulting expression is

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

where Θ are the possible probability distribution parameters, X are the possible outcomes, $P(X = x)$ is a normalization constant called the *marginal likelihood*, and $P(X = x|\Theta = \theta)$ and $P(\Theta = \theta)$ are the *likelihood* and *prior*, respectively, both of which will be revisited later. In other words, Eq. (1.4) states the probability of the parameters θ, given the knowledge of outcomes x.

A crucial parting of the Bayesians from the frequentists is at the introduction of the *prior*, which expresses a probability distribution on the *parameters* of the probability distribution. Fret not, this will be further explained.

1.1.2 Priors and Likelihood

The likelihood $P(X = x|\Theta = \theta)$ is simply the probability of the observations given the parameters. This is rather intuitive, the more interesting concept introduced is the prior. The prior expresses belief or assumption of the data, and has to be determined beforehand. By using Bayes' theorem in the version from Eq. (1.4), we end up with the measure $P(\Theta = \theta|X = x)$, also called the *posterior distribution*. This can be thought of as our prior belief, modified by how well this belief fits the data obtained,

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

Consider an example. The statistician from before has now grown tired of throwing dice, and starts to throw a coin. Before the throw he assumes the probability of getting heads or tails is equal, so he assumes a flat prior. After one throw he gets heads, and the posterior changes to a function with high probability for heads, and low for tails. Another heads makes the posterior even narrower around 1, but after several throws the distribution converges to a narrow peak around 0.25, as can be seen in Fig. (1.1), along with other examples of priors. This indicates an unfair coin. Note that the different priors arrive at the same posterior after a large amount of throws

1.1.3 Best Estimate and Reliability

Now that the posterior distribution has been defined, it would be useful with a way to quantify how well it actually fits the data. For that purpose the *best estimate* and *reliability* are introduced. The best estimate X_0 is the outcome with the highest probability. In other words, it is the maximum of the posterior

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

The second derivative must be negative in order to insure that it is, in fact, a maximum. Once the best estimate has been found, one should ask oneself how reliable this estimate is. This reliability is represented by the width of the distribution. A very narrow distribution has very low uncertainty, while a wide distribution has a higher probability for the values aroung the best estimate. The width is found using a Taylor expansion of the posterior, and taking the logarithm [1]

$$L = L(X_0) + \frac{1}{2}\frac{d^2}{dx^2}L\Big|_{X_0}(X - X_0)^2 + ..., \quad L = \log_e\Big[\text{prob}(x|\{data\}, I)\Big] \tag{1.6}$$

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

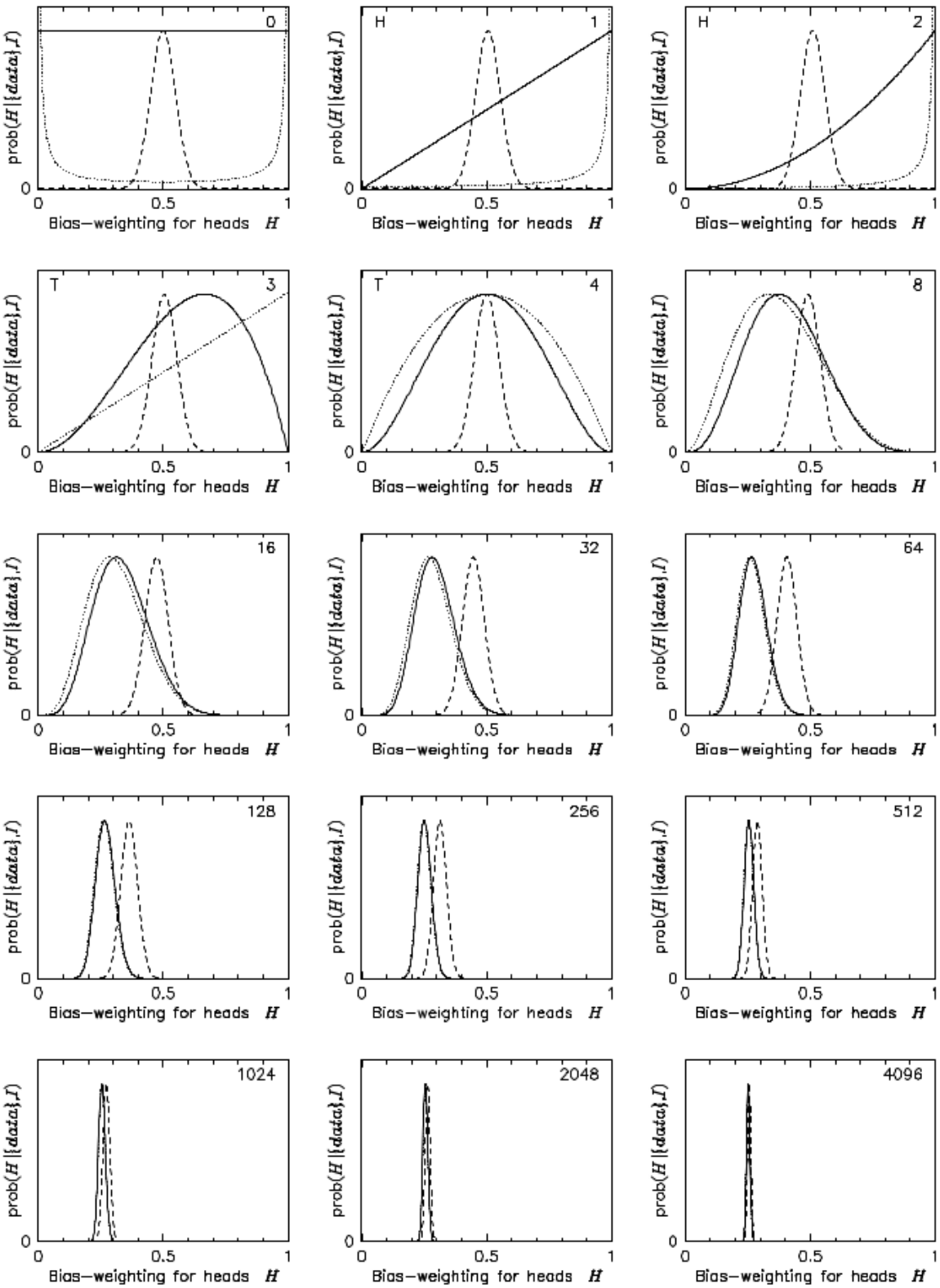


Figure 1.1: The effect of different priors, $P(H|I)$, on the posterior pdf for the bias-weighting of a coin. The solid line is a flat prior, the dashed line assumes a fair coin, and the dotted line is the Jeffreys prior. Figure from [4].

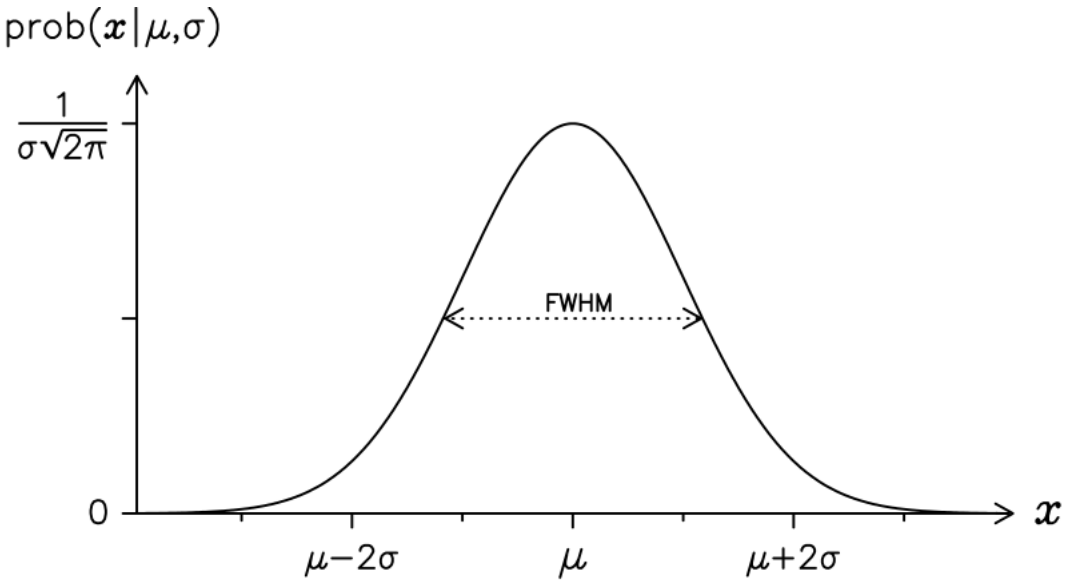


Figure 1.2: A Gaussian probability distribution. The maximum is at the mean value μ, with a full width at half maximum (FWHM) at around 2.35σ. Figure from [4].

This gives a proximate posterior which is a **Gaussian distribution**, with mean and variance given by

$$\text{prob}(x|\mu,\sigma) \text{ where } \mu = X_0,\ \sigma = \Big(-\frac{d^2L}{dx^2}\Big)^{-1/2}. \tag{1.7}$$

The Gaussian distribution is symmetric with respect to the maximum at $x = \mu$, and has a full width at half maximum (FWHM) at around 2.35σ, as shown in Fig. (1.2).

1.1.4 Covariance

The final term needed before embarking on Gaussian processes is *covariance*. In most cases the expressions one has to solve are not as simple as Eq. (1.6), as the probability distribution will normally have several quantities of interest $\{X_i\}$. In that case, one has to solve a set of *simultaneous equations* in order to get the best estimate

$$\frac{dP}{dX_i}\Big|_{X_{0j}} = 0, \tag{1.8}$$

This section is discussed in more detail in [4]. In order to get simpler expressions, consider the problem in two dimensions, so that $\{X_i\} = (X, Y)$. The Taylor

expansion of L is then

$$\begin{aligned} L =& L(X_0, Y_0) + \frac{1}{2}\Big[\frac{d^2L}{dX^2}\Big|_{X_0,Y_0}(X - X_0)^2 \\ &+ \frac{d^2L}{dY^2}\Big|_{X_0,Y_0}(Y - Y_0)^2 + 2\frac{d^2L}{dXdY}\Big|_{X_0,Y_0}(X - X_0)(Y - Y_0)\Big] + ... \end{aligned} \tag{1.9}$$

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 part of 1.9 in matrix form gives

$$Q = \begin{pmatrix} X - X_0 & Y - Y_0 \end{pmatrix} \begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} X - X_0 \\ Y - Y_0 \end{pmatrix}, \tag{1.10}$$

where the matrix elements are

$$A = \frac{\partial^2 L}{\partial X^2}\Big|_{X_0,Y_0}, \qquad B = \frac{\partial^2 L}{\partial Y^2}\Big|_{X_0,Y_0}, \qquad C = \frac{\partial^2 L}{\partial X \partial Y}\Big|_{X_0,Y_0}. \tag{1.11}$$

On a seemingly unrelated note, we define the *variance* as the expectation value of the square of deviations from the mean. In the two-dimensional case this becomes [4]

$$\mathrm{Var}(X) = \sigma_x^2 = \big\langle (X - X_0)^2 \big\rangle = \int\int (X - X_0)^2 P(X, Y|\{\mathrm{data}\}, I) dX dY. \tag{1.12}$$

This quantity is called the variance σ_X^2, and its square root is the standard deviation σ_X. A similar expression can be found for Y, by switching X and Y. It is also possible to find the simultaneous deviations of the parameters X and Y, or the correlation between the inferred parameters. This is called the *covariance* σ_{XY}^2, and is in this case

$$\sigma_{XY}^2 = \langle (X - X_0)(Y - Y_0) \rangle = \int\int (X - X_0)(Y - Y_0) P(X, Y|\{\mathrm{data}\}, I) dX dY. \tag{1.13}$$

The covariance indicates how an over- or underestimation of one parameter affects the other parameter. If, for example, an overestimation of X leads to an overestimation of Y, the covariance is positive. If the overestimation of X has little or no effect on the estimation of Y, the covariance is negligible or zero $|\sigma_{XY}| \ll \sqrt{\sigma_X^2 \sigma_Y^2}$. These effects are illustrated in Fig. (1.3). It can be shown that [4]

$$\mathrm{cov} = \begin{pmatrix} \sigma_X^2 & \sigma_{XY}^2 \\ \sigma_{XY}^2 & \sigma_Y^2 \end{pmatrix} = -\begin{pmatrix} A & C \\ C & B \end{pmatrix}^{-1}. \tag{1.14}$$

This is called the *covariance matrix*.

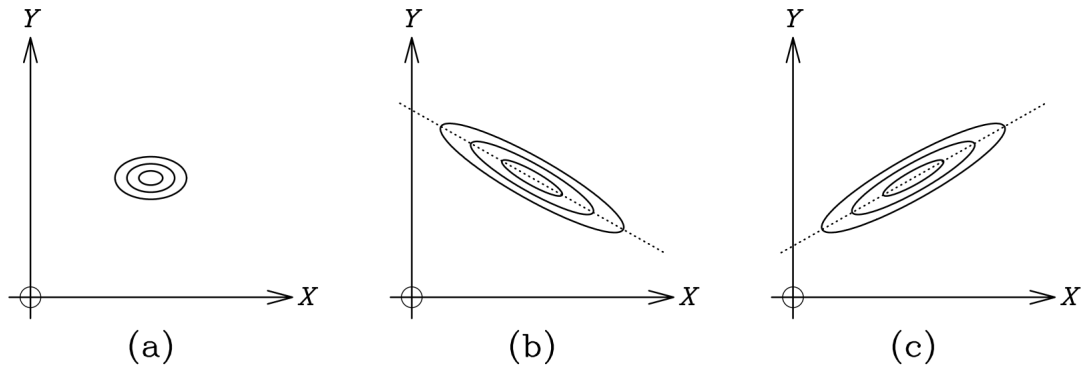


Figure 1.3: A schematic illustration of covariance and correlation. (a) The contours of a posterior pdf with zero covariance, where the inferred values of X and Y are uncorrelated. (b) The corresponding plot when the covariance is large and negative; $Y + mX =$ constant along the dotted line (where $m > 0$), emphasizing that only this sum of the two parameters can be inferred reliably. (c) The case of positive correlation, where we learn most about the difference YmX; this is constant along the dotted line. Figure from [4]

1.2 Gaussian Process Regression

Now that some basics of Bayesian statistics and Gaussian distributions have been covered, it's time to delve into the subject of Gaussian Processes. Gaussian processes are a supervised machine learning method, designed to solve regression and probabilistic classification problems. In this thesis the focus will be solely on regression problems. As experience dictates that Gaussian Processes can be somewhat hard to grasp, it will serve both the reader and the author well to begin on a conceptual level: *what do Gaussian Processes do?*

Consider a set of points $\{x_i\}, \{f(x_i)\}$, such as the blue dots shown in Fig. (1.4). The set of points is discrete, and so imagine that the value $f(x^*)$ for some x^* is unknown. Gaussian Processes (GP) finds a distribution *of function values* at this point x^*. This ditribution is Gaussian, and its mean is the value f^* predicted by the GP. The variance of the distribution σ^2 is the uncertainty of this value, as illustrated by the Gaussian distribution in the point x^* in Fig.

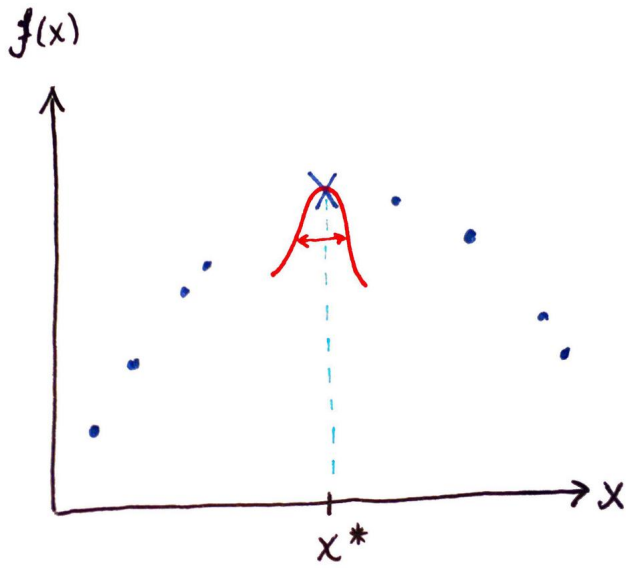


Figure 1.4: An illustration of a Gaussian Process prediction of the function value $f(x^*)$ (blue cross) given the known set of points $\{x_i, f(x_i)\}$ (blue dots). The prediction is a Gaussian distribution in $f(x^*)$ with mean $f(x^*)$ and variance σ^2.

(1.4). So Gaussian Processes provide a *distribution over function values*, meaning that for each x one can find a Gaussian distribution over $f(x)$ with a mean value $\mu \simeq f(x)$ and an uncertainty σ.

The mean value of the distribution in $f(x^*)$ is a linear combination of the known function values $\{f(x_i)\}$, weighted by the covariance between x_i and the test value x^*. The covariance is decided by the *kernel function*, which will be revisited.

The more general framework considers multidimensional $\mathbf{x}$, so that the set of known points, or *training points*, are a matrix $\{X_i\}$. Input vectors are in machine learning called *features*. The corresponding function values y_i are called *targets*. For a single point $\mathbf{x}$ the mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ are defined as

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})], \tag{1.15}$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \tag{1.16}$$

where $\mathbb{E}[y(\mathbf{x})]$ is the expectation value of some function $y(\mathbf{x})$. The mean and covariance define the Gaussian distribution for $f(\mathbf{x})$, which is denoted

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{1.17}$$

The covariance $k(\mathbf{x}_p, \mathbf{x}_q)$ is thus decided by a covariance function, or a *kernel*. The running example in this text with be the squared exponential (SE) kernel, given by

$$k(f(\mathbf{x}_p), f(\mathbf{x}_q)) = k(\mathbf{x}_p, \mathbf{x}_q) = \exp\Big(-\frac{1}{2}|\mathbf{x}_p - \mathbf{x}_q|^2\Big). \tag{1.18}$$

Note that the covariance of the function values $f(\mathbf{x}_p)$, $f(\mathbf{x}_q)$ only depends on the input parameters $\mathbf{x}_p$, $\mathbf{x}_q$.

Specifying the covariance function implies a distribution over functions [3]. This is because the allowed functions are now restricted to the set of functions that obey the correlation decided by $k(\mathbf{x}_p, \mathbf{x}_q)$. Using the kernel on a number of input points X^* gives the covariance matrix, and giving an inital mean of zero provides the distribution

$$f(x) \sim \mathcal{N}(0, K(X^*, X^*)). \tag{1.19}$$

This is the *prior* on the the function distribution. This distribution encodes the prior knowledge one has about the function values $f(x)$, and so the choice of kernel is one of the most important steps in learning with Gaussian Processes. This prior is modified by the training data to provide a posterior distribution. One can draw samples from both the prior and posterior, as shown in Fig. (1.5).

To apply GP, consider a simple example of a noise-free set of training points $\{x_i, f(x_i)\}$. The joint distribution of training outputs, $\mathbf{f}$, and test outputs, $\mathbf{f}^*$, according to the prior is

$$\begin{bmatrix}\mathbf{f}\\ \mathbf{f}^*\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}K(X,X) & K(X,X^*)\\ K(X,X^*) & K(X^*,X^*)\end{bmatrix}\right) \tag{1.20}$$

For n training points and n^* test points, $K(X,X)$ is the $n \times n$ matrix containing the covariance of training points, $K(X,X^*)$ is the $n \times n^*$ matrix of covariance between the test and training points, and $K(X^*,X^*)$ is the $n^* \times n^*$ matrix containing the covariance of test points. By **conditioning** the distribution of $\mathbf{f}^*$ on the observations, one finds [3] [2]

$$\mathbf{f}_*\big|X_*, X, \mathbf{f} \sim \mathcal{N}(K(X_*,X)K(X,X)^{-1}\mathbf{f}, \tag{1.21}$$

$$K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)). \tag{1.22}$$

Now one can draw samples from this distribution, as illustrated in Fig. (1.5).

1.2.1 Gaussian Noise Model

In the real world, noise-free observations are very hard to come by. In most cases, the signal will contain some noise $y = f(\mathbf{x}) + \varepsilon$, where the noise is assumed to

[2] For more details, see Appendix A.2 in [3].

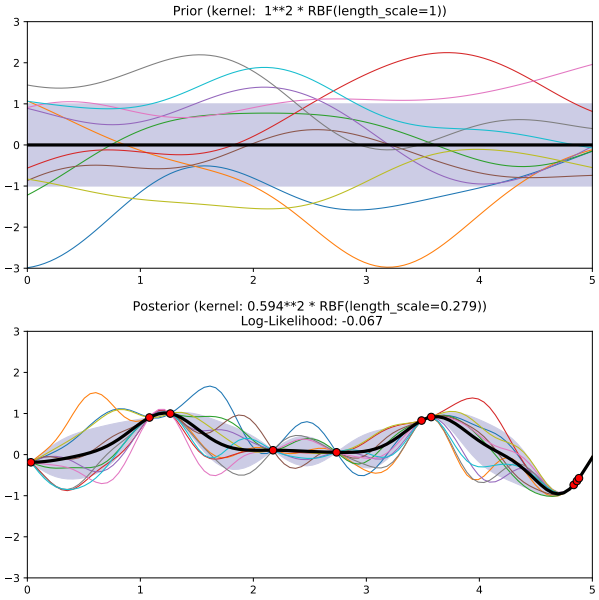


Figure 1.5: Drawing functions from the prior (top) and posterior (bottom) distributions. The prior has mean 0 and covariance given by the squared exponential function. The posterior has been modified by training points (red dots), where the uncertainty is now zero, and the mean value is changed. Figure generated using scikit-learn.

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

$$\text{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \qquad \text{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 \mathbb{I} \tag{1.23}$$

The prior distribution is now

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 \mathbb{I} & K(X, X_*) \\ K(X, X_*) & K(X_*, X_*) \end{bmatrix} \right), \tag{1.24}$$

which gives for the conditioned distribution

$$\mathbf{f}_* \big| X_*, X, \mathbf{f} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)), \text{ where} \tag{1.25}$$

$$\bar{\mathbf{f}}_* = K(X_*, X)[K(X, X) + \sigma_n^2 \mathbb{I}]^{-1} \mathbf{y}, \tag{1.26}$$

$$\text{cov}(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 \mathbb{I}]^{-1} K(X, X_*) \tag{1.27}$$

In order to tidy up the expression somewhat, define the matrix $K \equiv K(X, X)$ and the matrix $K_* = K(X, X_*)$. In the case of a single test point $\mathbf{x}_*$ write K_* as a vector $\mathbf{k}(\mathbf{x}_*) = \mathbf{k}_*$. Using this compact notation the GP prediction for a single test point $\mathbf{x}^*$ is

$$\bar{f}_* = \mathbf{k}_*^T (K + \sigma_n^2 \mathbb{I})^{-1} \mathbf{y}, \tag{1.28}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^T (K + \sigma_n^2 \mathbb{I})^{-1} \mathbf{k}_*. \tag{1.29}$$

The algorithm used in this project is implemented by `scikit-learn` [2], and is shown in Algorithm (1). The algorithm uses the Cholesky decomposition of the covariance matrix.

Data: X (inputs), $\mathbf{y}$ (targets), k (covariance function/kernel), σ_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 [3].

1.3 Covariance Functions

Covariance functions and kernels have been mentioned a few times already, and it is time to further investigate them. A 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* (invariant to rigid rotations in input space). 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.

A function which maps two arguments $\mathbf{x} \in \mathcal{X}$, $\mathbf{x}' \in \mathcal{X}$ into $\mathbb{R}$ is generally called a *kernel* k. Covariance functions are symmetric kernels, meaning that $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$. 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

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{1.30}$$

There are some restrictions on the covariance matrix, namely that is has to be *positive semidefinite* (PSD). This means that the $n \times n$ matrix K satisfies $Q(\mathbf{v}) = \mathbf{v}^T K \mathbf{v} \geq 0$ for all $\mathbf{v} \in \mathbb{R}^n$. A kernel k is PSD if

$$\int k(\mathbf{x}, \mathbf{x}') f(\mathbf{x}) f(\mathbf{x}') d\mu(\mathbf{x}) d\mu(\mathbf{x}') \geq 0, \tag{1.31}$$

for all $f \in L_2(\mathcal{X}, \mu)$.

1.3.1 The Radial Basis Function (RBF)

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

$$k_{SE}(r) = \exp\Big(-\frac{r^2}{2\ell^2}\Big), \tag{1.32}$$

where ℓ is the *characteristic length scale*. The length scale determines how smooth the function is, for example, 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 Fig. (**SETT INN ET EKSEMPEL HER**). The SE is infinitely differentiable, and so is very smooth.

The SE is implemented in `scikit-learn` under the name radial basis function, 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))
```

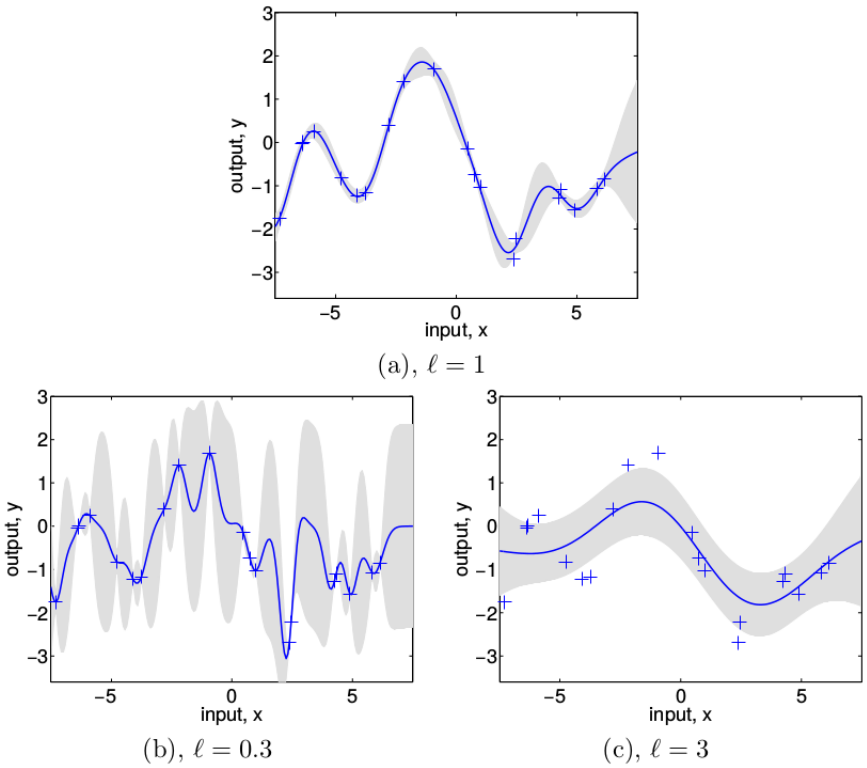


Figure 1.6: The effect of varying the length scale ℓ, where the blue + are data points, the blue line is the mean prediction, and the gray area is the uncertainty. Setting $\ell = 0.3$ gives a function that varies too quickly, and $\ell = 3$ is so slowly varying that it interprets much of the signal as noise. Choosing $\ell = 1$ gives the best fit to the values in this case. Figure from [3].

1.3.2 The Matérn Kernel

The *Matérn class of covariance functions* is given by

$$k_{Matérn}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big)^{\nu} K_{\nu}\Big(\frac{\sqrt{2\nu}r}{\ell}\Big), \tag{1.33}$$

where $\nu, \ell > 0$, and K_ν is a modified Bessel function. For $\nu \to \infty$ this becomes the SE. In the case of ν being half integer, $\nu = p + \frac{1}{2}$, the covariance function is simply the product of an exponential and a polynomial

$$k_{\nu=p+\frac{1}{2}} = \exp\Big(-\frac{\sqrt{2\nu}r}{\ell}\Big)\frac{\Gamma(p+1)}{\Gamma(2p+1)}\sum_{i=0}^{p}\frac{(p+i)!}{i!(p-i)!}\Big(\frac{\sqrt{8\nu}r}{\ell}\Big)^{p-i}. \tag{1.34}$$

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

$$k_{\nu=3/2}(r) = \Big(1 + \frac{\sqrt{3}r}{\ell}\Big)\exp\Big(-\frac{\sqrt{3}r}{\ell}\Big), \tag{1.35}$$

$$k_{\nu=5/2}(r) = \Big(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\Big)\exp\Big(-\frac{\sqrt{5}r}{\ell}\Big). \tag{1.36}$$

The Matérn kernel is considered more appropriate for physical processes [3], 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)
```

Other Kernels

There are other kernels, but they are not introduced 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 [3].

Hyperparameters

Each kernel has a vector of hyperparameters, e.g. $\boldsymbol{\theta} = (\{M\}, \sigma_f^2, \sigma_n^2)$ for the radial basis function (RBF)

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q))^T M(\mathbf{x}_p - \mathbf{x}_q) + \sigma_n^2\delta_{pq}. \tag{1.37}$$

The matrix M can have several forms, for example

$$M_1 = \ell^{-2}\mathbb{I}, M_2 = \text{diag}(\boldsymbol{\ell})^{-2}. \tag{1.38}$$

Choosing $\boldsymbol{\ell}$ to be a vector in stead of a scalar can in many cases be useful, especially if the vector of features contain 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 feature vector $\mathbf{x}$.

1.4 Model Selection

It becomes apparent that the choice of kernel and hyperparameters is important for the quality of the GP prediction. In this section Bayesian model selection is quickly overviewed, and the log marginal likelihood and cross validation are considered for their ability to optimize the GP.

1.4.1 Bayesian Model Selection

Feature selection at several levels: posterior over *parameters*, posterior over *hyperparameters* and posterior for the *model*,

$$p(\mathbf{w}|\mathbf{y}, X, \boldsymbol{\theta}, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)}{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)} \tag{1.39}$$

$$p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i) = \int p(\mathbf{y}|X, \mathbf{w}, \mathcal{H}_i)p(\mathbf{w}|\boldsymbol{\theta}, \mathcal{H}_i)d\mathbf{w} \quad \text{(marginal likelihood)} \tag{1.40}$$

$$p(\boldsymbol{\theta}|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \boldsymbol{\theta}, \mathcal{H}_i)p(\boldsymbol{\theta}|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)} \tag{1.41}$$

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)} \tag{1.42}$$

1.4.2 Log Marginal Likelihood

For Gaussian Processes with Gaussian we can find the exact expression for the marginal likelihood,

$$\log p(\mathbf{y}|X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi. \tag{1.43}$$

The optimal parameters are found by maximizing the marginal likelihood

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|X, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^T K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\mathrm{tr}(K^{-1}\frac{\partial K}{\partial\theta_j}). \tag{1.44}$$

This is what SciKitLearn uses, but can have **multiple local maxima**. Plug in Fig.

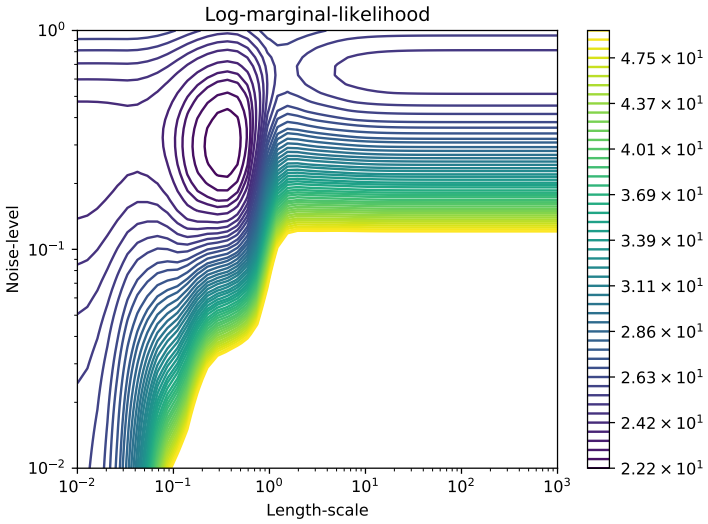


Figure 1.7: Used SciKitLearn. Several local maxima.

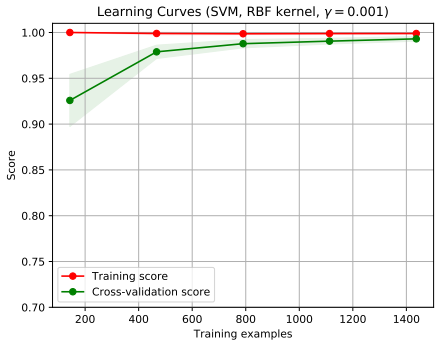


Figure 1.8: From scikit-learn.

1.4.3 Cross Validation

Divide into k-subsets and use validation and test set. Requires a loss function, e.g. R^2. Cross validate using Scikit-Learn, get a validation plot. Over training, over fitting, under-fitting.

1.4.4 Loss functions

Mean Relative Deviance

The mean relative deviance is used to quantify the quality of predictions

$$\varepsilon = \frac{y_{true} - y_{GP}}{y_{true}} \tag{1.45}$$

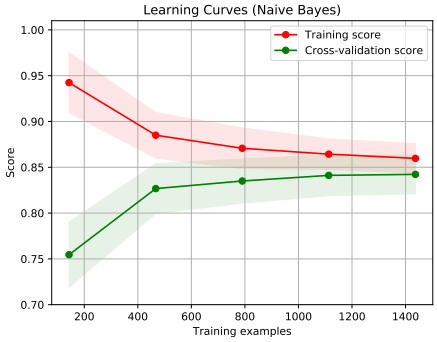
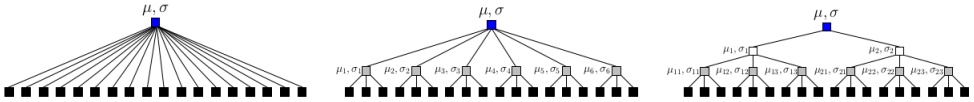


Figure 1.9: From scikit-learn.



(a) 1-layer model. (b) 2-layer model. (c) 3-layer model.

Figure 1.10: From [1].

R-Factor

1.5 Distributed Gaussian Processes

1.5.1 Limitations of Gaussian Processes

Problem because of $(K+\sigma_n^2\mathbb{I})^{-1}$, means inverting an $n \times n$-matrix. Training and predicting limits of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$. Limit $= \mathcal{O}(10^4)$. Some solutions exist, but **no prediction of variance is given with p-o-e.**

1.5.2 Product-of-Experts

Divide data between experts. "The assumption of independent GP experts leads to a block-diagonal approximation of the kernel matrix, which (i) allows for efficient training and predicting (ii) can be computed efficiently (time and memory) by parallelisation" [1].

Independence assumption

$$p(\mathbf{y}|\mathbf{X},\boldsymbol{\theta}) \approx \prod_{k=1}^{M} p_k(\mathbf{y}^{(k)}|\mathbf{X}^{(k)},\boldsymbol{\theta}) \tag{1.46}$$

$$\log p(\mathbf{y}^{(k)}|\mathbf{X}^{(k)},\boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{(k)}(\mathbf{K}_\psi^{(k)}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y}^{(k)} - \frac{1}{2}\log|\mathbf{K}_\psi^{(k)}+\sigma_\varepsilon^2\mathbf{I}| \tag{1.47}$$

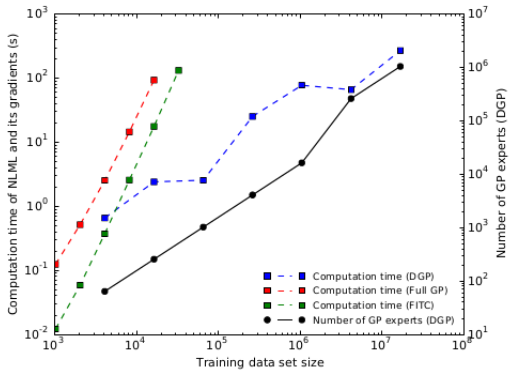


Figure 1.11: From [1].

1.5.3 Algorithm

$$\mu_*^{rbcm} = (\sigma_*^{rbcm})^2 \sum_k \beta_k \sigma_k^{-2}(\mathbf{x}_*)\mu_k(\mathbf{x}_*), \tag{1.48}$$

$$(\sigma_*^{rbcm})^{-2} = \sum_{k=1}^{M} \beta_k \sigma_k^{-2}(\mathbf{x}_*) + \big(1 - \sum_{k=1}^{M} \beta_k\big)\sigma_{**}^{-2}. \tag{1.49}$$

The posterior distribution for the test point $\mathbf{x}_*$ is given by a Gaussian with mean and variance

$$\mu(\mathbf{x}_*) = \mathbf{k}_*^T(\mathbf{K} + \sigma_\epsilon^2 \mathbb{I})^{-1}\mathbf{y}, \tag{1.50}$$

$$\sigma^2(\mathbf{x}_*) = k_{**} - \mathbf{k}_*^T(\mathbf{K} + \sigma_\epsilon^2 \mathbb{I})^{-1}\mathbf{k}_*. \tag{1.51}$$

Plot of time from the article

1.5.4 Implementing the Algorithm

Parallelizing

1.5.5 Benchmark

Data: $N_{experts}$ (number of experts), X (inputs), $\mathbf{y}$ (targets), k (covariance function/kernel), σ_n^2 (noise level), $\mathbf{x}^*$ (test input), $\mathbf{y}^*$ (test target)
X_{train}, X_{test}, y_{train}, y_{test} = train-test-split (X, y) (scikit-learn) ;
$y = \log_{10}(y)$;
$n = \frac{\text{Number of data points}}{N_{experts}}$;
$subsets = array\_split(X_{train}, n)$;
$\mu_{rbcm} = []$, $\sigma_{rbcm} = []$ (empty lists to be filled later);
for *each expert* **do**
  $gp_{temporary} = GaussianProcessRegressor.fit(X_{expert}, y_{expert})$;
  for *each* y^* *in* $\boldsymbol{y}^*$ **do**
    $\mu_*, \sigma_*^2 = gp_{temporary}.predict(x^*)$;
    $\sigma_{**}^2 = k(x^*, x^*)$;
    (fill inn the values) ;
    $\boldsymbol{\mu}[\text{expert}][x^*] = \mu_*^2$ (mean value from this expert);
    $\boldsymbol{\sigma}^2[\text{expert}][x^*] = \sigma_*^2$ (variance from this expert);
    $\boldsymbol{\sigma}_{**}^2[\text{expert}][x^*] = \sigma_{**}^2$ (variance from initial kernel)
  end
end
for *each expert* **do**
  for *each* y_* *in* $\boldsymbol{y}_*$ **do**
    $\mu_* = \boldsymbol{\mu}[\text{expert}][x_*]$ (retrieve relevant values);
    $\sigma_*^2 = \boldsymbol{\sigma}^2[\text{expert}][x^*]$;
    $\sigma_{**}^2 = \boldsymbol{\sigma}_{**}^2[\text{expert}][x^*]$;
    $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$;
    $(\sigma_*^{rbcm})^{-2}[y_*] += \beta\sigma^{-2} + \big(\frac{1}{n_{experts}} - \beta\big)\sigma_{**}^{-2}$
  end
end
for *each expert* **do**
  for *each* y_* *in* $\boldsymbol{y}_*$ **do**
    $\mu_* = \boldsymbol{\mu}[\text{expert}][x_*]$ (retrieve relevant values);
    $\sigma_*^2 = \boldsymbol{\sigma}^2[\text{expert}][x^*]$;
    $\sigma_{**}^2 = \boldsymbol{\sigma}_{**}^2[\text{expert}][x^*]$;
    $\beta = \frac{1}{2}(\log(\sigma_{**}^2) - \log(\sigma_*^2))$;
    $\mu_*^{rbcm}[y_*] += (\sigma_*^{rbcm})^2\beta\sigma_*^{-2}\mu_*$
  end
end
$\epsilon = \frac{10^{\mu_{rbcm}} - 10^{y_{test}}}{10^{y_{test}}}$ (relative error);
Result: Approximative distribution of $f_* = f(\mathbf{x}_*)$ with mean μ_*^{rbcm} and variance $(\sigma_*^{rbcm})^2$.

Algorithm 2: Algorithm for using rBCM on a single test point $\mathbf{x}_*$. The $GaussianProcessRegressor.fit()$-function is a function in scikit-learn, that uses Algorithm (**??**).

Bibliography

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

[2] 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.

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

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