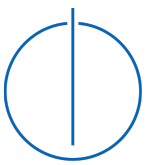
DEPARTMENT OF MATHEMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Seminar: Gaussian Processes for Machine Learning

Gaussian Processes for Regression

Author: Davit Papikyan
Lecturer: Prof. Dr. Michael Marc Wolf



Contents

1 Introduction

Supervised Learning is an approach to creating a model in Machine Learning that is trained on input data that has been labeled for particular output. The main classes of problems are regression and classification. Classification is related to data categorization in which the labels are discrete. In simpler words, classification is the grouping of related data points into classes. In case of regression, the labels are continuous. The aim of regression is to explain the relationships between a dependent variable and one or more independent variables.

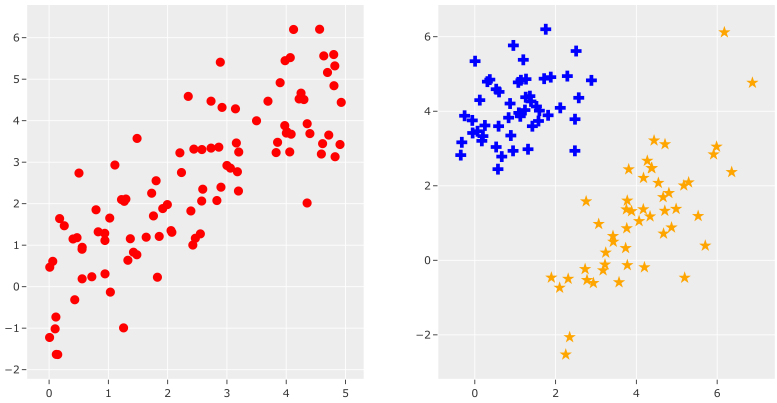


Figure 1: Regression (left) vs. classification (right) in terms of data given in Euclidean space

In this report we will analyze Gaussian Processes for solving the regression. Particularly, in section 2 we will analyze Bayesian interpretation of Linear Regression and will introduce the concept of kernel. Next, in section 3, we will introduce Gaussian Processes for regression and state how it is connected with Bayesian Linear Regression. In the end we will explore on a high level how hyperparameter tuning is done to achieve better results and will analyze the pros and cons of the model. [3]

2 Bayesian Linear Regression

Consider a training set with n observations, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) \mid i = 1, 2, \ldots, n\}$ where $\boldsymbol{x}$ is the input vector of dimension d and y is a scalar target value that we aim to learn

to predict. In a vectorized notation we have a data matrix (also called design matrix) $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ which stores input data points as columns and a target vector $\boldsymbol{y} \in \mathbb{R}^n$. The goal of Bayesian Linear Regression is to model the relationship between $\boldsymbol{X}$ and $\boldsymbol{y}$, that is, the conditional distribution of targets given inputs.

We assume that data is generated from the following process

$$y_i = f(\boldsymbol{x}_i) + \epsilon \text{ where } \epsilon \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2)\,.$$

where $\boldsymbol{w}$ is the weight vector we want to learn. Note that the variance of noise σ^2 is a hyperparameter (we will talk about it later). The second and main assumption is that f is linear, i.e. $f(\boldsymbol{x}_i) = \boldsymbol{x}_i^T \boldsymbol{w}$. In order to make predictions for unseen data we need to derive the posterior predictive distribution over a single unseen data point for which we will need Likelihood, Prior and Posterior distributions.

2.1 Likelihood

Likelihood is the probability density of data given the parameters. Because of the independence assumption, the joint density can be factorized as given below

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) &= \prod_{i=1}^{n} p(y_i \mid \boldsymbol{x}_i, \boldsymbol{w}) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2}{2\sigma^2} \right] \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left[-\frac{1}{2\sigma^2} \|\boldsymbol{y} - \boldsymbol{X}^T \boldsymbol{w}\|^2 \right] = \mathcal{N}(\boldsymbol{X}^T \boldsymbol{w}, \sigma^2 \boldsymbol{I})\,. \end{aligned}$$

2.2 Prior

Prior is the probability density of parameters expressing our beliefs about them before observing the data. We will assume that parameters are driven from 0-centered Gaussian with some covariance matrix $\boldsymbol{\Sigma}_w$, i.e.

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_w)\,.$$

The choice of mean to be zero comes from the concept of Overfitting, a well-known problem in Machine Learning. To avoid that, one needs to lower the magnitude of weights of a model which is achieved by 0-centered assumption of prior.

2.3 Posterior

Posterior is the probability density of parameters given the data.

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \qquad p(\boldsymbol{w} \mid \boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w}) \times p(\boldsymbol{w})}{p(\boldsymbol{y} \mid \boldsymbol{X})}$$

The denominator of posterior, known as marginal likelihood, is independent of $\boldsymbol{w}$ and plays a role of normalization constant. It is given by the following formula

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int p(\boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{w})\, p(\boldsymbol{w})\, d\boldsymbol{w}\,.$$

Since the density of a normally distributed random variable $Z \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can up to a normalization constant be written as $\exp\left[-\frac{1}{2}(\boldsymbol{z}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right]$, then

$$\begin{aligned} p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) &\propto \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y}-\boldsymbol{X}^T\boldsymbol{w})^T(\boldsymbol{y}-\boldsymbol{X}^T\boldsymbol{w})\right] \exp\left[\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Sigma}_w\boldsymbol{w}\right] \\ &\propto \exp\Bigg[-\frac{1}{2}(\boldsymbol{w}-\widetilde{\boldsymbol{w}})^T \underbrace{\Big(\frac{1}{\sigma^2}\boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{\Sigma}_w^{-1}\Big)}_{\boldsymbol{A}} (\boldsymbol{w}-\widetilde{\boldsymbol{w}})\Bigg]\,, \end{aligned}$$

where $\widetilde{\boldsymbol{w}} = \frac{1}{\sigma^2}\boldsymbol{A}\boldsymbol{X}\boldsymbol{y}$ and $\boldsymbol{A} = \frac{1}{\sigma^2}\boldsymbol{X}\boldsymbol{X}^T + \boldsymbol{\Sigma}_w^{-1}$. Therefore

$$p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{N}(\widetilde{\boldsymbol{w}}, \boldsymbol{A}^{-1})\,.$$

2.4 Posterior predictive distribution

Now focus on the main problem – predicting a target y_* for an unseen data point $\boldsymbol{x}_*$. For so we need to compute the posterior predictive distribution $p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$:

$$\begin{aligned} p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) &= \int p(y_*, \boldsymbol{w} \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})\, d\boldsymbol{w} \\ &= \int p(y_* \mid \boldsymbol{x}_*, \boldsymbol{w})\, p(\boldsymbol{w} \mid \boldsymbol{X}, \boldsymbol{y})\, d\boldsymbol{w} \\ &= \mathcal{N}\left(\frac{1}{\sigma^2}\boldsymbol{x}_*{}^T\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{y},\ \boldsymbol{x}_*{}^T\boldsymbol{A}^{-1}\boldsymbol{x}_*\right). \end{aligned}$$

Using the mean of posterior predictive as a predicted target we can also estimate the *uncertainty* using the covariance.

2.5 Addressing nonlinear dependency

In case when the dependency between y and $\boldsymbol{x}$ is not linear, we can transform data points to some other space where the dependency between the response variable and new features will be linear. Let $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^k$ with $\boldsymbol{\Phi} := \Phi(\boldsymbol{X}) \in \mathbb{R}^{k \times n}$ be the aggregation of columns $\boldsymbol{\phi}(\boldsymbol{x})$ for all samples in training set. Our model will now become $f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$ where $\boldsymbol{w} \in \mathbb{R}^k$. The posterior predictive is given below

$$p(y_* \mid \boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\left(\frac{1}{\sigma^2}\boldsymbol{\phi}(\boldsymbol{x}_*)^T \boldsymbol{A}^{-1}\boldsymbol{\Phi}\boldsymbol{y},\, \boldsymbol{\phi}(\boldsymbol{x}_*)^T \boldsymbol{A}^{-1}\boldsymbol{\phi}(\boldsymbol{x}_*)\right)$$

where $\boldsymbol{A} = \frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_w^{-1}$.

Note that we need to compute the inverse of a $k \times k$ matrix which is not tractable for large values of k. To address this problem, we will try to transform both mean and variance of the posterior predictive distribution to escape from computation of $\boldsymbol{A}^{-1}$. Denote $\boldsymbol{K} := \boldsymbol{\Phi}^T \boldsymbol{\Sigma}_w \boldsymbol{\Phi}$ and $\boldsymbol{\phi}_* := \boldsymbol{\phi}(\boldsymbol{x}_*)$. Let's start with rewriting the mean.

$$\boldsymbol{A}\boldsymbol{\Sigma}_w\boldsymbol{\Phi} = \frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_w\boldsymbol{\Phi} + \boldsymbol{\Phi} = \frac{1}{\sigma^2}\boldsymbol{\Phi}(\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_w\boldsymbol{\Phi} + \sigma^2\boldsymbol{I}) = \frac{1}{\sigma^2}\boldsymbol{\Phi}(\boldsymbol{K} + \sigma^2\boldsymbol{I})\,.$$

Multiplying both sides by $\boldsymbol{A}^{-1}$ from left and $(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}$ from right we will get

$$\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1} = \frac{1}{\sigma^2}\boldsymbol{A}^{-1}\boldsymbol{\Phi}\,,$$

by using which we can rewrite the mean of posterior predictive distribution as

$$\frac{1}{\sigma^2}\boldsymbol{\phi}_*{}^T\boldsymbol{A}^{-1}\boldsymbol{\Phi}\boldsymbol{y} = \boldsymbol{\phi}_*{}^T\boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}\,.$$

Next we rewrite the variance using the matrix inversion lemma

$$(\boldsymbol{B} + \boldsymbol{U}\boldsymbol{C}\boldsymbol{V})^{-1} = \boldsymbol{B}^{-1} - \boldsymbol{B}^{-1}\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{B}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\boldsymbol{B}^{-1}\,.$$

By denoting $\boldsymbol{B} := \boldsymbol{\Sigma}_w^{-1}$, $\boldsymbol{U} := \boldsymbol{\Phi}$, $\boldsymbol{C} := \frac{1}{\sigma^2}\boldsymbol{I}$, $\boldsymbol{V} := \boldsymbol{\Phi}^T$ we will get

$$\boldsymbol{A}^{-1} = \left(\frac{1}{\sigma^2}\boldsymbol{\Phi}\boldsymbol{\Phi}^T + \boldsymbol{\Sigma}_w^{-1}\right)^{-1} = \boldsymbol{\Sigma}_w - \boldsymbol{\Sigma}_w\boldsymbol{\Phi}(\sigma^2\boldsymbol{I} + \underbrace{\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_w\boldsymbol{\Phi}}_{\boldsymbol{K}})^{-1}\boldsymbol{\Phi}^T\boldsymbol{\Sigma}_w\,.$$

After plugging the above equation for $\boldsymbol{A}^{-1}$ into the expression of variance we get

$$\boldsymbol{\phi_*}^T \boldsymbol{A}^{-1} \boldsymbol{\phi_*} = \boldsymbol{\phi_*}^T \boldsymbol{\Sigma_w} \boldsymbol{\phi_*}^T - \boldsymbol{\phi_*}^T \boldsymbol{\Sigma_w} \boldsymbol{\Phi} (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{\Sigma_w} \boldsymbol{\phi_*} .$$

Thus, the posterior predictive distribution can be rewritten as

$$p(y_* \mid \boldsymbol{x_*}, \boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}\Big(\boldsymbol{\phi_*}^T \boldsymbol{\Sigma_w} \boldsymbol{\Phi} (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}, \\ \boldsymbol{\phi_*}^T \boldsymbol{\Sigma_w} \boldsymbol{\phi_*}^T - \boldsymbol{\phi_*}^T \boldsymbol{\Sigma_w} \boldsymbol{\Phi} (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{\Sigma_w} \boldsymbol{\phi_*} \Big)$$

where we need to invert matrix of size $n \times n$ which is more convenient when $n < k$.

In the following examples we have generated synthetic data points (in red) from $y = sin(x)$ true signal by perturbing function values with Gaussian noise. In the first example we don't transform data, i.e. we use identity basis function $\phi(x) = x$.

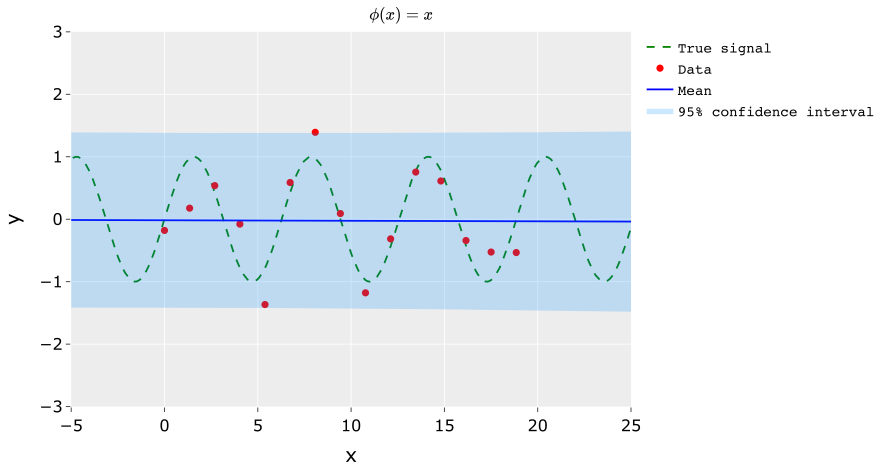


Figure 2: Bayesian Linear Regression applied on data generated from $sin(x)$ function perturbed by a negligible Gaussian noise

In contrast to the first example, below we have applied $sin(x)$ transformation onto original data space. One can easily see that transformed model fits the data better and approximates true signal with high accuracy (see Figure 3).

2.6 Kernel trick

From the expressions of mean and variance obtained for posterior predictive distribution, one can notice that feature vectors $\boldsymbol{\phi(x)}$ appear always with covariance matrix

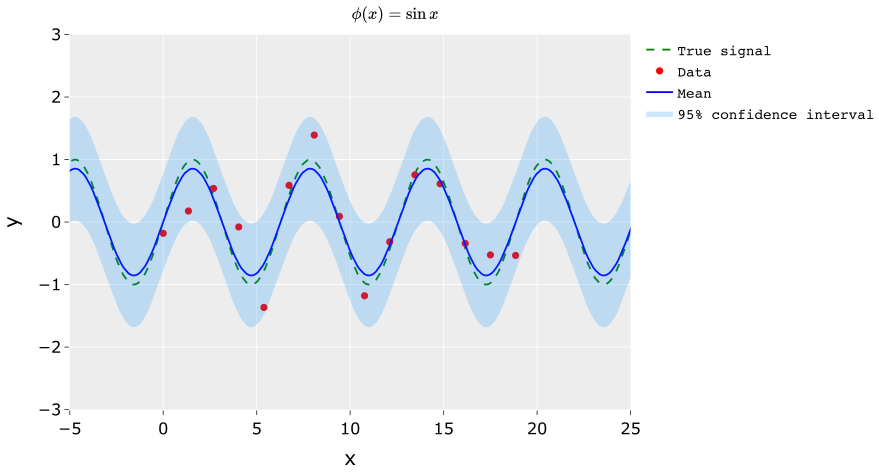


Figure 3: Bayesian Linear Regression applied on sine transformed data generated from $sin(x)$ function perturbed by a negligible standard Gaussian noise

of prior, i.e. in the forms of $\mathbf{\Phi}^T\mathbf{\Sigma}_w\mathbf{\Phi}$, $\boldsymbol{\phi}_*{}^T\mathbf{\Sigma}_w\mathbf{\Phi}$ or $\boldsymbol{\phi}_*{}^T\mathbf{\Sigma}_w\boldsymbol{\phi}_*$. An entry of each of these matrices has the form of $k(\boldsymbol{x}, \boldsymbol{x'}) := \boldsymbol{\phi}(\boldsymbol{x})^T\mathbf{\Sigma}_w\boldsymbol{\phi}(\boldsymbol{x'})$ which is known as *kernel function*. If we define $\boldsymbol{\psi}(\boldsymbol{x}) := \mathbf{\Sigma}_w^{1/2}\boldsymbol{\phi}(\boldsymbol{x})$ we can rewrite kernel as a simple dot product

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \boldsymbol{\psi}(\boldsymbol{x})^T\boldsymbol{\psi}(\boldsymbol{x'}),$$

where $\mathbf{\Sigma}_w^{1/2} = \boldsymbol{UD}^{1/2}\boldsymbol{U}^T$ for $\boldsymbol{UDU}^T$ being the SVD of $\mathbf{\Sigma}_w$. That is, we can think of a kernel value to be a measure of interaction (similarity) of data points. Kernel function provides us with an efficient (computationally cheaper) way to transform data without having to compute the coordinates in a new space, i.e. feature vectors $\boldsymbol{\phi}(\boldsymbol{x})$. This approach is known in the literature as **kernel trick**.
Being equipped with the preliminary knowledge we are now ready to discuss Gaussian Process Regression.

3 Gaussian Process Regression

Besides assuming that data is normally distributed with some 0-centered noise, we additionally assume that function f is a random variable drawn from a Gaussian Process conditioned on data.

3.1 Gaussian Process

Definition: Gaussian Process

Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a positive definite kernel and $m : \mathbb{R}^d \to \mathbb{R}$ be a finite real-valued function. Then a random function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be a Gaussian Process (GP) with mean m and covariance kernel k, denoted by $GP(m, k)$, if the following holds: for any finite set $X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) \subset \mathbb{R}^d$ of any size $n \in \mathbb{N}$, the random vector

$$f_X = (f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n))^T \in \mathbb{R}^n$$

follows a multivariate normal distribution $\mathcal{N}(m_X, k_{XX})$ with covariance $k_{XX} = [k(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j=1}^n$ and mean vector $m_X = [m(\boldsymbol{x}_1), m(\boldsymbol{x}_2), \ldots, m(\boldsymbol{x}_n)]^T$. It is written as

$$f \sim GP(m, k).$$

The key point of the definition is that the function values $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n)$ are jointly Gaussian for any finite n.

In order to proceed with regression, as stated in the definition of Gaussian Process, we need to define mean and kernel (covariance) functions

$$\begin{aligned} m(\boldsymbol{x}) &= \mathbb{E}_{f \sim GP(m,k)}[f(\boldsymbol{x})], \\ k(\boldsymbol{x}, \boldsymbol{x}') &= \mathbb{E}_{f \sim GP(m,k)}\left[\big(f(\boldsymbol{x}) - m(\boldsymbol{x})\big)\big(f(\boldsymbol{x}') - m(\boldsymbol{x}')\big)\right]. \end{aligned}$$

Let's take the mean to be a constant 0 function and define the kernel to be RBF (Radial Bases Function) with length scale parameter of 1

$$k(\boldsymbol{x}, \boldsymbol{x}') = \operatorname{cov}\big(f(\boldsymbol{x}), f(\boldsymbol{x}')\big) = \exp\left(-\frac{1}{2}||\boldsymbol{x} - \boldsymbol{x}'||_2^2\right).$$

Assuming to have a finite point set $\boldsymbol{X}_*$ we can compute the corresponding covariance matrix $K(\boldsymbol{X}_*, \boldsymbol{X}_*)$ by applying the kernel function to each pair of points. Next, let's draw some samples from the prior distribution $f \sim \mathcal{N}\big(\boldsymbol{0}, K(\boldsymbol{X}_*, \boldsymbol{X}_*)\big)$ (see Figure 4).

3.2 Posterior predictive distribution

Recall the standard assumptions $y = f(\boldsymbol{x}) + \epsilon$ where noise is i.i.d. $\mathcal{N}(0, \sigma^2)$. This implies that

$$\begin{aligned} \operatorname{cov}(y_i, y_j) = \operatorname{cov}\big(f(\boldsymbol{x}_i) + \epsilon_i, f(\boldsymbol{x}_j) + \epsilon_j\big) = \\ \operatorname{cov}\big(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)\big) + \operatorname{cov}\big(f(\boldsymbol{x}_i), \epsilon_j\big) + \\ \operatorname{cov}\big(\epsilon_i, f(\boldsymbol{x}_j)\big) + \operatorname{cov}\big(\epsilon_i, \epsilon_j\big) = \\ k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sigma^2 \delta_{ij} \end{aligned}$$

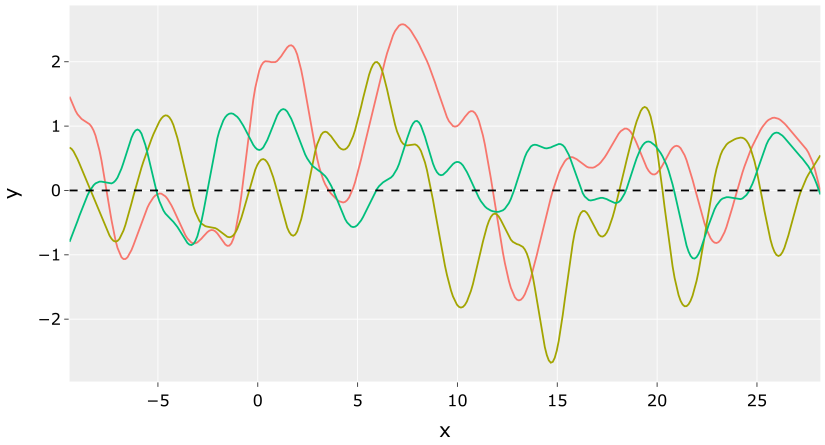


Figure 4: "Samples" drawn from prior distribution

or in a vectorized notation

$$\operatorname{cov}(\boldsymbol{y}) = k(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}\,.$$

According to the definition of Gaussian processes, the joint distribution of observed and test (unseen) values is again Gaussian

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I} & K(\boldsymbol{X}, \boldsymbol{X}_*) \\ K(\boldsymbol{X}_*, \boldsymbol{X}) & K(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix}\right).$$

Using the property that Gaussian distribution is closed under conditioning, one can write that

$$\boldsymbol{f}_* \,|\, \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_* \sim \mathcal{N}\big(\bar{\boldsymbol{f}}_*, \operatorname{cov}(\boldsymbol{f}_*)\big)$$

where

$$\begin{aligned} \bar{\boldsymbol{f}}_* &:= \mathbb{E}[\boldsymbol{f}_* \,|\, \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{X}_*] = K(\boldsymbol{X}_*, \boldsymbol{X})\big(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}\big)^{-1} \boldsymbol{y}\,, \\ \operatorname{cov}(\boldsymbol{f}_*) &= K(\boldsymbol{X}_*, \boldsymbol{X}_*) - K(\boldsymbol{X}_*, \boldsymbol{X}_*)\big(K(\boldsymbol{X}, \boldsymbol{X}) + \sigma^2 \boldsymbol{I}\big)^{-1} K(\boldsymbol{X}, \boldsymbol{X}_*)\,. \end{aligned}$$

Let's rewrite the result that we got for a single test point in order to compare it with Bayesian Linear Regression case. Denote $\boldsymbol{K} := K(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}_* := K(\boldsymbol{X}, \boldsymbol{X}_*)$, $\boldsymbol{k}_* := \boldsymbol{k}(\boldsymbol{x}_*) = [k(\boldsymbol{x}_i, \boldsymbol{x}_*)]_{i=1}^n$ the covariance vector of a test point $\boldsymbol{x}_*$ and n training points. Then,

$$f_* \,|\, \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{x}_* \sim \mathcal{N}\big(\bar{f}_*, \mathbb{V}[f_*]\big)$$

where

$$\bar{f}_* = \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}]\right)^{-1} \boldsymbol{y}\,,$$
$$\mathbb{V}[f_*] = \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_*^T \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_*\,.$$

The prediction of a Gaussian Process Regression is the value computed by the mean function of its posterior predictive distribution, i.e. by $\bar{f}_*$. $\mathbb{V}[f_*]$, as in the case of Bayesian Linear Regression, provides us with an estimate of uncertainty of predictions.

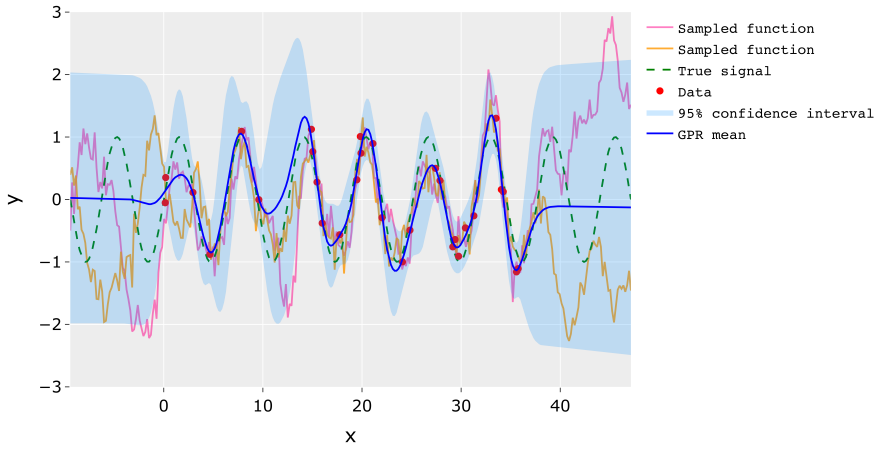


Figure 5: Gaussian Process Regression example

Interestingly, the model can also be used for solving a well known problem in numerical mathematics - interpolation. To do so, we only need to assume that the generated data is noise-free, i.e. $\epsilon = 0$ and do the corresponding changes in posterior predictive distribution (see Figure 6).

3.3 Equivalence to Bayesian Linear Regression

Recall that the prediction function of Bayesian Linear Regression is provided by its mean which has the following form

$$\boldsymbol{\phi_*}^T \boldsymbol{\Sigma}_w \boldsymbol{\Phi} (\boldsymbol{K} + \sigma^2 \boldsymbol{I})^{-1} \boldsymbol{y}\,.$$

If we define the kernel to be $k(\boldsymbol{x}_i, \boldsymbol{x}_j) := \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\Sigma}_w \boldsymbol{\phi}(\boldsymbol{x}_j)$, then the prediction function of Gaussian Process regression $\boldsymbol{k}(\boldsymbol{x}_*)^T \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y}$ will become equivalent to that of Bayesian Linear Regression.

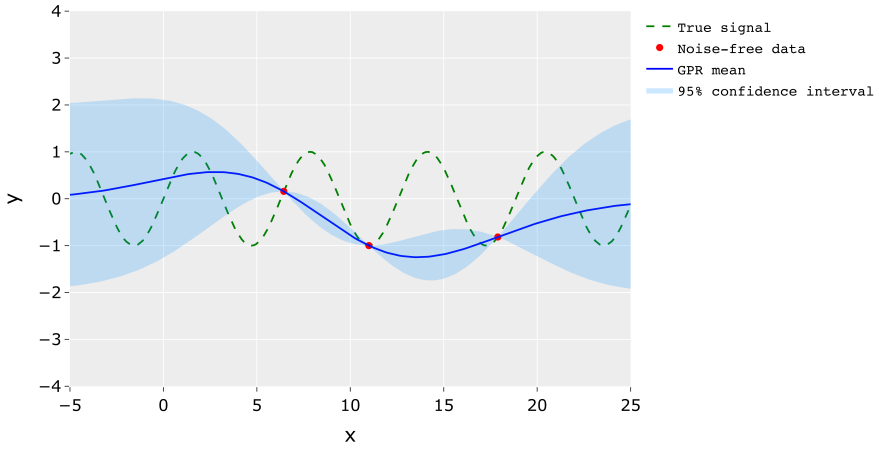


Figure 6: Gaussian Process Interpolation example

3.4 Hyperparameter optimization

In general, the performance of Gaussian Process Regression is highly dependent on hyperparameters - parameters that affect the model's behavior but are not learnt during the training process. An example of a hyperparameter is length scale l in RBF kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2^2}{2\,l^2}\right).$$

Below we can see the effect of different values of length scale on prior functions sampled with constant 0 mean

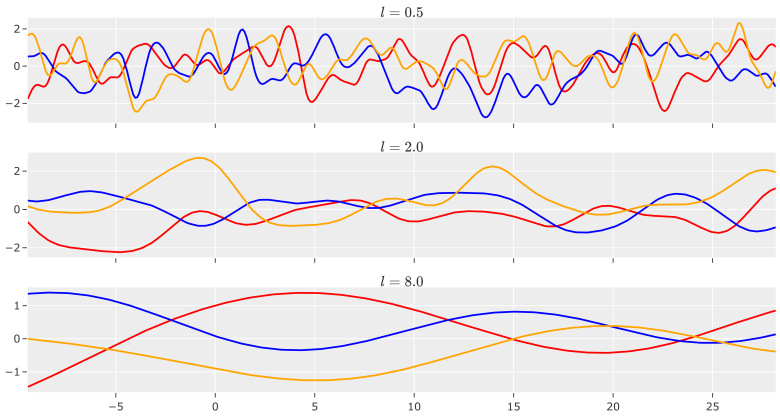


Figure 7: Fluctuations of prior functions increase with the increase of length scale

Hyperparameters can be optimized by maximizing so-called marginal likelihood (equivalently marginal log-likelihood)

$$p(\boldsymbol{y} \mid \boldsymbol{X}) = \int \overbrace{p(\boldsymbol{y} \mid \boldsymbol{f}, \boldsymbol{X})}^{\text{likelihood}} \overbrace{p(\boldsymbol{f} \mid \boldsymbol{X})}^{\text{prior}} d\boldsymbol{f} .$$

Because prior and likelihood above are Gaussian, $\boldsymbol{f} \mid \boldsymbol{X} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K})$ and $\boldsymbol{y} \mid \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{f}, \sigma^2 \boldsymbol{I})$, then the final marginal log-likelihood can be given by

$$\log p(\boldsymbol{y} \mid \boldsymbol{X}) = -\frac{1}{2} \boldsymbol{y}^T \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} - \frac{1}{2} \log |\boldsymbol{K} + \sigma^2 \boldsymbol{I}| - \frac{n}{2} \log 2\pi .$$

Denoting by Θ the set of all hyperparameters, then finding the optimal hyperparameters is equivalent to the following optimization problem

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \log p(\boldsymbol{y} \mid \boldsymbol{X}, \Theta) ,$$

which can be solved with any gradient-based optimization algorithm.

3.5 Pros and Cons

Gaussian Process Regression is flexible model since it can take any valid kernel function to define a prior distribution, i.e. the prediction function's shape. It is interpretable mostly because it allows to estimate model's uncertainty about any prediction (or region of domain). Another advantage is that these models can directly be optimized given the hyperparameters, as the computation of posterior predictive is pure Linear Algebra stuff.

Because the mean of the posterior predictive can be represented as a linear combination of targets

$$\bar{f}(\boldsymbol{x}_*) = \underbrace{\boldsymbol{k}(\boldsymbol{x}_*)^T \left(\boldsymbol{K} + \sigma^2 \boldsymbol{I}\right)^{-1}}_{\boldsymbol{\alpha}} \boldsymbol{y} = \textstyle\sum_{i=1}^{n} \alpha_i \boldsymbol{y}_i ,$$

then having outliers in the data can drive the mean function (and hence the predictions). Based on this we can state that Gaussian Process Regression is prone to outliers unless the hyperparameters and kernel are cleverly chosen. Moreover, since the mean and covariance of posterior predictive involve matrix/vector products and inversions, computing them when having high-dimensional data can be computationally expensive.

Bibliography

[1] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur. *Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences*. 2018. arXiv: 1807.02582 [stat.ML].

[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." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[3] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006, pp. I–XVIII, 1–248. ISBN: 026218253X.