Brief Introduction to Gaussian Processes

Overview

There is something many statisticians, regardless of their expertise, would probably agree on: regression is an essential tool, which is of great importance in several disciplines. From learning the inverse dynamics of a robot arm, [11], to predicting the price of cryptocurrencies, e.g. Bitcoin, [8], it is clear that several subjects require the involvement of some form of regression.

Regression belongs in the field of Supervised Learning and it is a way to predict the relationship between a dependent variable, which is usually called the response variable and one or more independent variables, usually called covariates or features, [3]. Due to its popularity in several fields, including engineering, physical and chemical sciences, astronomy, economics, [6], it has been studied extensively by scientists. Additionally, it has been said that regression "may be the most widely used statistical technique", [6].

In this report, a more sophisticated model used in regression will be considered; the so called Gaussian Process. A Gaussian Process, as the name betrays is a stochastic process, such that every finite subset of those random variables is a multivariate Gaussian distribution, a formal definition and a description of its properties will be given in Section 2.

The main idea behind the Gaussian Process is that they define a distribution over functions. To achieve this, one combines the available data (likelihood) and any prior information over functions (prior distribution) and using the Bayes theorem, one obtains the posterior distribution. From this, we understand that the choice of prior is of great importance, as it provides information about certain function properties, such as, smoothness, periodicity, differentiability. Additionally, even the hyperparameters of the prior distribution can drastically change the behaviour of a Gaussian Process, but there exist ways to tune these hyperparameters.

As we shall see later, the most important function the user needs to specify is the covariance function of the prior distribution. The only requirement for the covariance function to be valid is that it needs to be positive semi–definite. There are several common function used in practice, and each comes with specific properties and apriori assumptions. It might also be wise to combine some of those covariance functions so as to describe more complex shapes.

Although Gaussian Processes is a powerful algorithm which can be used for regression, a major disadvantage of them is that they are computationally expensive. To put it into perspective for a training set size of N, the training Gaussian Process prediction scales in $\mathcal{O}(N^3)$, which puts a practical limit of $N \approx 1000$. There exist certain solution approaches but they are not in the scope of this report. The interested reader is referred to, Sparse Gaussian Processes with inducing variables [4] and Combination of local Gaussian Process expert models [2].

Introduction

As already mentioned in the overview, regression is an important tool used in many different fields. In this report we are going to take a look at one of the many models used for regression; Gaussian Process. Gaussian Process is a non–parametric model that generalises the Gaussian probability distribution, and it has been used in several applications, [11].

The ultimate goal is to make predictions on new inputs x^*, given some training data. The "challenge" is to move from the finite training dataset to a function f that will eventually make predictions for every new realisation. The way a Gaussian Process works is by introducing a prior probability for every possible function, which gives a higher probability to functions that we consider apriori to be more likely, e.g. because we believe the data might contain some sort of periodicity.

The structure of the report is as follows. In Section 2, we provide more details about the problem and formally introduce the Gaussian Process. In Section 3, we discuss how the marginalisation property allows us to use Gaussian Processes in practice. In Section 4, we give the link of a Gaussian Process Regression with the Bayesian Inference problem. In Section 5, we provide the mathematical foundation behind Gaussian Process predictions. In Section 6, we define some of the most common covariance functions. In Section 7, we give a common way of tuning the hyperparameters and a way to conduct model selection. Finally, in Section 8, we combine all previous knowledge and use Gaussian Processes with different kernels to perform regression in a simulated dataset.

Problem Setting

The simplest model for regression is the simple linear model where the output is a linear combination of the inputs. Albeit being simple and easily interpretable, it is not flexible as one requires the assumption that the relationship between input and output is linear. There are a couple of ways to extend the simple linear model but for this report we are going to focus on Gaussian Processes (GP).

Our aim is to find a distribution over functions, say $p(f)$, that explains the data. There are two ways one can approach this setting through the Bayesian viewpoint. For the purposes of this report we will consider inference directly in function space. To do this, we define a Gaussian Process (GP), to use it to describe a distribution over functions. Formally, [11],

Definition 1. *A Gaussian process is a collection of random variables $f_1, f_2, \ldots$, any finite number of which is Gaussian distributed.*

Informally, one can think of a GP as being a generalisation of a multivariate Gaussian distribution to infinitely many variables. A Gaussian distribution is specified by a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\Sigma}$, while a GP is completely specified by a *mean function* $m(\cdot)$ and a *covariance function (kernel)* $k(\cdot, \cdot)$, defined as,

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

From now on we will write the Gaussian process as,

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

The mean function allows us to bias the model (which could make sense in application–specific settings) but for notational simplicity we usually take the mean function to be zero. Thus, the "agnostic" mean function, in the absence of data prior knowledge is, $m(\cdot) \equiv 0$, everywhere.

The covariance function is symmetric and positive semi–definite. It allows us to compute the covariances/correlations between (unknown) function values by looking at the corresponding inputs:

$$\mathrm{Cov}(f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)) = k(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Marginalisation Property with Gaussian Processes

As a preliminary stage, we turn our attention to the marginalisation property which will eventually help us with the computation of GPs.

We treat a function as an infinitely long vector of function values, $(f_1, f_2, \dots)$, and look at a distribution over function values $f_i = f(\boldsymbol{x}_i)$. Consider a finite number of N function values, say $\boldsymbol{f}$, and all other (infinitely many) function values, say $\tilde{\boldsymbol{f}}$. Informally, we have,

$$\begin{pmatrix} \boldsymbol{f} \\ \tilde{\boldsymbol{f}} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mu_{\boldsymbol{f}} \\ \mu_{\tilde{\boldsymbol{f}}} \end{pmatrix}, \quad \begin{pmatrix} \Sigma_{\boldsymbol{f},\boldsymbol{f}} & \Sigma_{\boldsymbol{f},\tilde{\boldsymbol{f}}} \\ \Sigma_{\tilde{\boldsymbol{f}},\boldsymbol{f}} & \Sigma_{\tilde{\boldsymbol{f}},\tilde{\boldsymbol{f}}} \end{pmatrix} \right], \tag{1}$$

where $\Sigma_{\tilde{\boldsymbol{f}},\tilde{\boldsymbol{f}}} \in \mathbb{R}^{m\times m}$ and $\Sigma_{\boldsymbol{f},\tilde{\boldsymbol{f}}} \in \mathbb{R}^{N\times m}, m \to \infty$. For completeness we state that, $\Sigma_{\boldsymbol{f},\boldsymbol{f}} = \mathrm{Cov}[f(\boldsymbol{x}_i), f(\boldsymbol{x}_j)] = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. However, the marginal remains finite,

$$p(\boldsymbol{f}) = \int p(\boldsymbol{f}, \tilde{\boldsymbol{f}}) d\tilde{\boldsymbol{f}}, \text{ so } \boldsymbol{f} \sim \mathcal{N}(\mu_{\boldsymbol{f}}, \Sigma_{\boldsymbol{f},\boldsymbol{f}}).$$

In practice, we would always have finite training, say $\boldsymbol{f}$, and test, say $\boldsymbol{f}^*$, inputs, thus by integrating out all other (unknown) values we get the finite marginal,

$$p(\boldsymbol{f}, \boldsymbol{f}^*) = \int p(\boldsymbol{f}, \boldsymbol{f}^*, \boldsymbol{f}_{\text{other}}) d\boldsymbol{f}_{\text{other}}$$

and

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{f}^* \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mu_{\boldsymbol{f}} \\ \mu_{\boldsymbol{f}^*} \end{pmatrix}, \quad \begin{pmatrix} \Sigma_{\boldsymbol{f},\boldsymbol{f}} & \Sigma_{\boldsymbol{f},\boldsymbol{f}^*} \\ \Sigma_{\boldsymbol{f}^*,\boldsymbol{f}} & \Sigma_{\boldsymbol{f}^*,\boldsymbol{f}^*} \end{pmatrix} \right], \tag{2}$$

From this, we understand that computing the joint distribution of an arbitrary number of training, and test inputs in a computer, boils down to manipulating finite–dimensional Gaussian distributions.

Gaussian Process Regression as a Bayesian Inference Problem

In this section we give a formal justification of how we are using GPs for regression. A simple example of a GP can be obtained from the Bayesian linear regression model

$$f(\boldsymbol{x}) = \phi^{\intercal}(\boldsymbol{x})\boldsymbol{w}, \qquad y = f(\boldsymbol{x}) + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma_n^2),$$

where $\phi(\boldsymbol{x})$ maps a $D-$dimensional input vector $\boldsymbol{x}$ into an N dimensional feature space and w is a vector of weights (parameters) of the linear model. The projection of inputs into a higher dimensional space using the set of basis functions $\phi(\cdot)$ is done so as to provide more expressiveness.

Using the so called *kernel trick* we can turn our focus from the feature space to the corresponding covariance function/kernel. This is valid because for every positive definite covariance function $k(\cdot,\cdot)$, there exists a possible infinite expansion in terms of basis functions, [5]. We thus understand that for every Bayesian regression model, there exists a corresponding covariance function which can be used instead. This is often useful as it is more convenient to compute the kernel, $k(\cdot,\cdot)$, rather than the feature vectors, $\phi(\cdot)$.

Therefore, from now on we will use a covariance function $k(\cdot,\cdot)$ to model a Bayesian linear regression. More specifically we will,

- Use a prior over functions $p(f)$, with mean m and kernel k, $\boldsymbol{f} \sim \mathcal{GP}(m, k)$.
- Have a normal likelihood, $y|f, X \sim \mathcal{N}(f(X), \sigma_n^2)$.

The Bayes' theorem then, yields the following posterior over functions,

$$p(f|X,\boldsymbol{y}) = \frac{\overbrace{p(\boldsymbol{y}|f,X)}^{\text{likelihood}}\overbrace{p(f)}^{\text{prior}}}{\underbrace{p(y|X)}_{\text{marginal}}}.$$

Using Gaussian properties we get that $f(\cdot)|X,\boldsymbol{y} \sim \mathcal{GP}(m_{\text{posterior}}(\cdot), k_{\text{posterior}}(\cdot,\cdot))$, where,

$$\begin{aligned} m_{\text{posterior}}(\cdot) &= m(\cdot) + k(\cdot,X)(K+\sigma_n^2 I)^{-1}(y - m(X)). \\ k_{\text{posterior}}(\cdot,\cdot) &= k(\cdot,\cdot) - k(\cdot,X)(K+\sigma_n^2 I)^{-1}K(X,\cdot), \end{aligned}$$

where $K = k(X,X)$. As we shall see in the next section this posterior is going to be used for predictions. Additionally, the marginal can be computed analytically (using Gaussian properties), $p(\boldsymbol{y}|X) = \mathcal{N}(\boldsymbol{y}|m(X), K+\sigma_n^2 I)$ and it can be used for model selection and tuning of hyperparameters. We note again that this is equivalent to a Bayesian linear regression for some basis functions $\phi(\cdot)$ but is now written in terms of the covariance function $k(\cdot,\cdot)$.

Mathematical Foundation for Gaussian Process Predictions

Assume now that we have training data $X, \boldsymbol{y}$ and are interested in finding function values corresponding to test inputs X^*. Define $\boldsymbol{f} := (f(x_{\text{train}}^{(1)}, \ldots, f(x_{\text{train}}^{(n_{\text{train}})})$ and $\boldsymbol{f}^* := (f(x_{\text{test}}^{(1)}, \ldots, f(x_{\text{test}}^{(n_{\text{test}})})$. Then, since $f \sim \mathcal{GP}$, if follows that $\boldsymbol{f}, \boldsymbol{f}^*$ are jointly Gaussian distributed we get,

$$\begin{pmatrix} \boldsymbol{f} \\ \boldsymbol{f^*} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} m(X) \\ m(X^*) \end{pmatrix}, \quad \begin{pmatrix} K & k(X,X^*) \\ k(X^*,X) & k(X^*,X^*) \end{pmatrix}\right].$$

Since $\boldsymbol{f}$ is unobserved, we have,

$$\begin{pmatrix} \boldsymbol{y} \\ \boldsymbol{f^*} \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} m(X) \\ m(X^*) \end{pmatrix}, \quad \begin{pmatrix} K+\sigma_n^2 I & k(X,X^*) \\ k(X^*,X) & k(X^*,X^*) \end{pmatrix}\right].$$

The posterior predictive distribution $p(f^*|X,\boldsymbol{y},X^*)$ at test inputs X^* is obtained by Gaussian conditioning,

$$\begin{aligned} f^*|X,\boldsymbol{y},X^* &\sim \mathcal{N}(\mathbb{E}[f^*|X,\boldsymbol{y},X^*], \mathbb{V}[f^*|X,\boldsymbol{y},X^*]) \\ \mathbb{E}[f^*|X,\boldsymbol{y},X^*] &= m_{\text{posterior}}(X^*) = m(X^*) + k(X^*,X)(K+\sigma_n^2 I)^{-1}(\boldsymbol{y} - m(X)) \qquad (3) \\ \mathbb{V}[f^*|X,\boldsymbol{y},X^*] &= k_{\text{posterior}}(X^*,X^*)) = \underbrace{k(X^*,X^*)}_{\text{prior variance}} - \underbrace{k(X^*,X)(K+\sigma_n^2 I)^{-1}K(X,X^*)}_{\geq 0}. \qquad (4) \end{aligned}$$

We can notice few things from Equations 3, 4. First, there is no need to explicitly calculate the feature vectors as the calculations only involve the covariance function k. Also, the mean prediction in Equation 3 is a linear combination of the observations $\boldsymbol{y}$. Additionally, the variance in Equation 4 does not depend on the observed targets, only on the inputs. Further, we can see that it is the difference of two terms, the prior variance minus a positive term, which shows the amount of information the observations gives us about the function.

Covariance Function

A Gaussian process requires the specification of a kernel/covariance function k. This function needs to be symmetric and positive semi–definite and encodes high–level structural assumptions about the latent function, e.g. smoothness, differentiability and periodicity. In addition, it provides a way to define similarity between data points. We are going to introduce a few covariance functions. To simplify the notation we define $r = |\boldsymbol{x} - \boldsymbol{x}'|$.

Squared Exponential Covariance Function

One of the most commonly used covariance functions in practice is the *squared exponential* or *radial basis function (rbf)* , which has the form, [10],

$$k_{\text{SE}}(r) = \sigma_f^2 \exp\left(-\frac{r^2}{2l^2}\right),$$

where l is the *length–scale* parameter, indicating how far we need to move in the input space before the function value changes significantly, and σ_f^2 determines the average distance of the function away from its mean. This covariance function is infinitely differentiable, which means that the GP with this covariance function is smooth. Due to this assumption being unrealistic in practice, [9], the Matérn class has been recommended.

Matérn Class Covariance Function

The Matérn class of covariance function has the form, [11],

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

with positive parameters ν and l, where l is as before a smoothness parameter.

It is noteworthy to see that if $\nu \to \infty$ we obtain the Squared Exponential covariance function. Additionally, for this class of functions, the process needs to be k–times differentiable if and only if $\nu > k$. The most common values in Machine Learning are $\nu = 3/2$ and $\nu = 5/2$.

Polynomial Covariance Function

The polynomial covariance function has the form, [11],

$$k(\boldsymbol{x}, \boldsymbol{x}') = (\sigma_0^2 + \boldsymbol{x}^\intercal \Sigma_p \boldsymbol{x}')^p,$$

where Σ_p is a general covariance matrix of the components of $\boldsymbol{x}$ and p is a positive–integer. Kernels of this class have been effectively used in high–dimensional classification problems, where the input data are binary or greyscale normalized to $[-1, 1]$ on each dimension, [11].

Periodic Covariance Function

The periodic covariance function has the form, [10],

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{2\sin^2\left(\pi\frac{x-x'}{p}\right)}{l^2}\right),$$

where p is the period and l is the lengthscale parameter and σ_f^2 is the same as before Functions of this class are used when we want to model repetition (controlled by p).

Synthesising New Covariance Functions

Previously, we introduced a few well–known kernels used commonly in practice. There are situations though where one would want to combine existing covariance functions to create new ones. For example, if one wants to model a smooth trend with a seasonal variation, one could combine a squared exponential and a periodic covariance function.

Assume k, k_1, k_2 are valid kernels, and $g(\boldsymbol{x}) = a(\boldsymbol{x})f(\boldsymbol{x})$, where $a(\boldsymbol{x})$ is a deterministic (possibly non–linear) function. Then,

- The sum of the two kernels, $k_1 + k_2$, is a valid kernel.
- The product of the two kernels, $k_1 k_2$, is a valid kernel.

- Choose $a(\boldsymbol{x}) = k^{-1/2}(\boldsymbol{x}, \boldsymbol{x})$, then, $\tilde{k}(\boldsymbol{x}, \boldsymbol{x}') = \frac{k(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{k(\boldsymbol{x}, \boldsymbol{x})}\sqrt{k(\boldsymbol{x}', \boldsymbol{x}')}}$ is a valid kernel which ensures $\tilde{k}(\boldsymbol{x}, \boldsymbol{x}) = 1, \forall x$.

The above sections on covariance functions provide a list, which is definitely not exhaustive. The interested reader could find more at [10, 11]

Choosing the Hyper–Parameters and Model Selection in a Gaussian Process

We have already mentioned that a GP consists of a set of hyper–parameters. These include, the parameters of the mean function (which is usually 0), the parameters of the covariance function and the likelihood parameters.

The goal is to find a way to choose the hyper–parameters in a way that would make probabilistic sense. The idea is to use probability distributions to represent possibility of hyper–parameters while incorporating information from the data. Bayes' theorem is again going to be used here, by placing a prior $p(\theta)$ on the hyper–parameters and afterwards using the likelihood of the parameters (which indicates how well does θ predict the data we observed). Thus we get the following posterior over hyper–parameters,

$$p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) = \frac{p(\boldsymbol{\theta})p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})}, \quad \text{where} \quad p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \int p(\boldsymbol{y}|f, \boldsymbol{X})p(f|\boldsymbol{X}, \boldsymbol{\theta})df.$$

We assume a uniform prior $p(\boldsymbol{\theta})$, thus we choose hyper–parameters $\boldsymbol{\theta}^*$ such that,

$$\boldsymbol{\theta}^* \in \arg\max_{\boldsymbol{\theta}} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}).$$

Thus, our goal becomes to maximise the (log) marginal likelihood. As we have already seen we can write the marginal likelihood as a Normal $\mathcal{N}(0, K + \sigma_n^2 I)$ (assuming $m(\cdot) \equiv 0$) and we thus get,

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^\intercal K_y^{-1}\boldsymbol{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi,$$

where $K_y = K + \sigma_n^2 I$. Note that we have explicitly written the marginal conditioned on the hyper-parameters $\boldsymbol{\theta}$.

To maximise the log marginal likelihood, we find the partial derivative of it with respect to the hyperparameters,

$$\begin{aligned}\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})}{\partial \theta_i} &= \frac{1}{2}\boldsymbol{y}^\intercal K_y^{-1}\boldsymbol{y} - \frac{1}{2}\mathrm{tr}\left(K_y^{-1}\frac{\partial K_y}{\partial \theta_i}\right) \\ &= \frac{1}{2}\mathrm{tr}\left((\alpha\alpha^\intercal - K_y^{-1})\frac{\partial K_y}{\partial \theta_i}\right), \text{ where } \alpha = K_y^{-1}\boldsymbol{y}. \end{aligned} \tag{5}$$

The problem with maximising the marginal likelihood is that it is non–convex and it might suffer from multiple local optima. However, empirically it seems that local maxima do not complicate things, [11]. The problem arises when there is not enough data points, since it becomes harder for the model to reject other possibilities. However, when we have higher data sizes, it is common for one local optima to be orders of magnitude more probable than other ones.

Assume now we have a finite set of models M_i each one specifying a certain mean function m_i and kernel k_i. If our goal is to find the best model, the most common way is to compare the marginal likelihood (assuming a uniform prior on the set of candidate models), [11]. Other possibilities include Cross Validation, [11] and Bayesian Information Criterion, [1], the analysis of both is not in the scope of this report.

Example

We are now ready to implement a Gaussian Process in an example and see the effects of different kernels. For the purposes of this example we generate a dataset of $n = 80$ data points with a smooth linear trend and periodicity which can be seen in Figure 1. We use $n = 60$ as the training set and $n = 20$ as the test set.

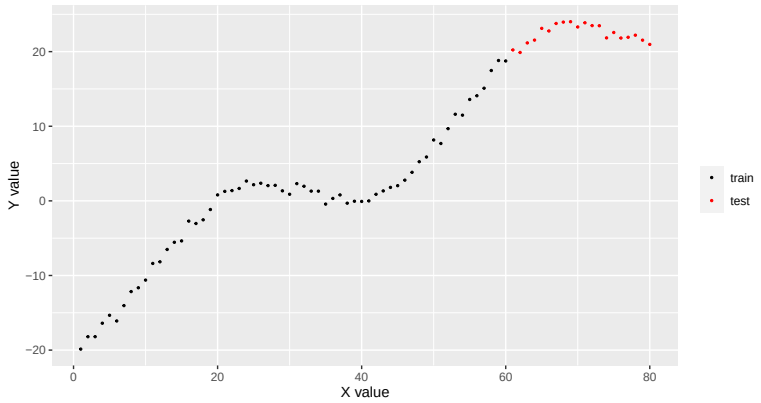

Figure 1: Generated dataset of size $n = 80$. The black dots $n = 60$ correspond to the training set, while the red dots $n = 20$ correspond to the test set.

Since we assume apriori that the data have some sort of periodicity we are going to use a periodic covariance function. Additionally, we will use a squared exponential covariance function as it one of the most common used in practice. Finally, we will use a product of the two functions which intuitively should provide the best result as the data have a smooth trend and a seasonal variation.

The kernel hyperparameters, σ_f, σ_n (noise variance), l and the period p are going to be tuned by maximising the marginal likelihood. To do this we are going to implement Gradient Ascent (GA), which we present now, [7].

GA is a first-order iterative optimization algorithm for finding a local maximum of a differentiable function $F(x)$. We start with an initial guess x_0 for a local maximum of F (in this context the marginal likelihood) and then find the sequence $x_0, x_1, \ldots$ such that,

$$x_{n+1} = x_n + \gamma_n \nabla F(x_n), n \geq 0,$$

where $\nabla F(x_n)$ is given by Equation 5 and γ_n is the learning rate and is usually kept constant, although it can be allowed to vary.

The learning rates used, as well as the final values of the hyper parameters and the log marginal likelihoods for the three kernels are provided in Table 1. We used 10^{-5} as the convergence threshold and 1000 as the maximum iterations allowed. Additionally, we ran the algorithm for different starting values 20 times, so as to be more confident in our results. Furthermore, to guarantee that our result is not far from the global optimum, we used a grid search and we conclude that we are confident that our results correspond to the global maximum.

	Learning Rate	σ_f	l_{rbf}	l_{per}	p	σ_{n}	log-marg-likelihood
Periodic	0.01	2.23	-	10.14	32.05	9.71	-215.8
rbf	0.01	4.26	13.27	-	-	1.01	-106.7
rbf x periodic	0.01	2.38	15.10	19.99	33.12	1.15	-99.4

Table 1: Learning rates for GA and final tuned hyperparameters for the three kernels.

Figure 2 shows the fitted model for the three kernels. From it we can see what we expected. The periodic kernel fails to pick up on the upwards trend and thus has the worst log marginal likelihood. The other two kernels seem to fit the training data quite well with the rbf x periodic having the highest log marginal likelihood, which indicates it is a better model. Finally the rbf model seems to perform worse on the test, which could be attributed to the fact that it does not use the extra information about the

period. We also note that on the test set, without data both models start to revert back to 0, which is the prior mean.

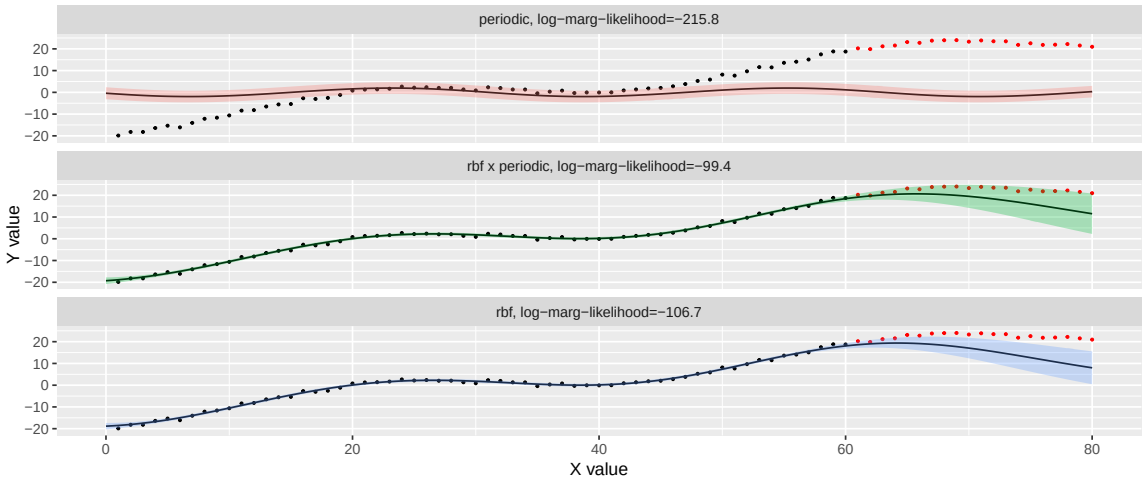

Figure 2: Plots showing the fitted models for each kernel used, with 95% confidence intervals. The black dots correspond to the training set, while the red dots correspond to the test set.

Conclusion

In this short report we presented the Gaussian Process model and explained how it can be used for regression. Additionally, we presented several commonly used covariance functions and used 3 of them in an example with a generated dataset.

References

[1] Scott Chen, Ponani Gopalakrishnan, et al. "Speaker, environment and channel change detection and clustering via the bayesian information criterion". In: *Proc. DARPA broadcast news transcription and understanding workshop*. Vol. 8. Virginia, USA. 1998, pp. 127–132.

[2] Marc Deisenroth and Jun Wei Ng. "Distributed gaussian processes". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 1481–1490.

[3] Julian J Faraway. *Linear models with R*. Chapman and Hall/CRC, 2004.

[4] James Hensman, Nicolo Fusi, and Neil D Lawrence. "Gaussian processes for big data". In: *arXiv preprint arXiv:1309.6835* (2013).

[5] Ha Quang Minh, Partha Niyogi, and Yuan Yao. "Mercer's theorem, feature maps, and smoothing". In: *International Conference on Computational Learning Theory*. Springer. 2006, pp. 154–168.

[6] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[7] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[8] Devavrat Shah and Kang Zhang. "Bayesian regression and Bitcoin". In: *2014 52nd annual Allerton conference on communication, control, and computing (Allerton)*. IEEE. 2014, pp. 409–414.

[9] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[10] *The Kernel Cookbook:* URL: `https://www.cs.toronto.edu/~duvenaud/cookbook/`.

[11] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.