Predicting the Future and Quantifying the Uncertainty

Tanakij Jivacharoen
201684567

Supervised by Kurt Langfeld

Submitted in accordance with the requirements for the
module MATH5872M: Dissertation in Data Science and Analytics
as part of the degree of

Master of Science in Data Science and Analystics

The University of Leeds, School of Mathematics

September 2023

The candidate confirms that the work submitted is his/her own and that appropriate credit has been given where reference has been made to the work of others.

UNIVERSITY OF LEEDS

School of Mathematics

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

Academic integrity statement

I am aware that the University defines plagiarism as presenting someone else's work, in whole or in part, as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance.

I promise that in the attached submission I have not presented anyone else's work, in whole or in part, as my own and I have not colluded with others in the preparation of this work. Where I have taken advantage of the work of others, I have given full acknowledgement. I have not resubmitted my own work or part thereof without specific written permission to do so from the University staff concerned when any of this work has been or is being submitted for marks or credits even if in a different module or for a different qualification or completed prior to entry to the University. I have read and understood the University's published rules on plagiarism and also any more detailed rules specified at School or module level. I know that if I commit plagiarism I can be expelled from the University and that it is my responsibility to be aware of the University's regulations on plagiarism and their importance.

I re-confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes. I confirm that I have declared all mitigating circumstances that may be relevant to the assessment of this piece of work and that I wish to have taken into account. I am aware of the University's policy on mitigation and the School's procedures for the submission of statements and evidence of mitigation. I am aware of the penalties imposed for the late submission of coursework.

Name Tanakij Jivacharoen

Student ID 201684567

Acknowledgements

I would like to thank my supervisor, Professor Kurt Langfeld, the Head of the School of Mathematics at the University of Leeds. His guidance, expertise, and consistent support have been invaluable throughout the period of my research. His insightful feedback and advice have greatly contributed to the refinement of my work. I am truly fortunate to have had the opportunity to learn under his mentorship. In addition, I would like to extend my appreciation to the entire academic community at the University of Leeds for fostering an environment of learning and intellectual growth.

Abstract

This dissertation explores the application of Gaussian processes to analyse noisy time-series data, particularly focusing on the stock market data using the Alphabet Inc. (GOOG) stock data collected from Yahoo Finance website. With the ubiquitous presence of noisy time-series data in modern applications, the need to capture underlying trends and predict future trajectories as well as complete missing data is crucial. Gaussian processes, as a probabilistic and principled framework, offer an approach to achieve these goals, providing both interpolation and extrapolation capabilities along with associated uncertainties. It can also perform the task of filling in missing data. We explore various kernel functions in the Gaussian process framework to investigate their impacts on data analysis and prediction. By utilising different time windows, the study presents the detailed analysis of the behaviour of Gaussian processes and their ability to model stock market data. Key findings highlight the suitability of specific kernel choices for different tasks, such as completing missing data and predicting trends, while emphasising the importance of including linear kernel and accounting for noise. Through comprehensive analysis and experimentation, our dissertation contributes to a deeper understanding of Gaussian process applications in time-series analysis and prediction. We also provides insights into model behavior, paves the way for more robust and accurate predictive modeling, and facilitates decision-making in financial domains and beyond.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In modern applications, noisy time-series data plays a vital role and often lacks a definitive theoretical underpinning (curve). The primary endeavor is to encapsulate the data with an underlying time-dependent function (interpolation) and use this function for predictive insights into the data's future trajectory. While various techniques exist for extrapolating data, assessing the reliability of such extrapolation is important. This is where a Gaussian process, a stochastic framework, comes into play. This research aims to employ the Gaussian processes with diverse covariance functions (kernel functions) to fit observed data and yield predictions including confidence interval bands (uncertainties). Additionally, the study seeks to delve into the influence of specific covariance functions (kernel functions) on the attained outcomes.

In the present thesis, the Gaussian process is used to analyse financial stock market data, focusing on the Alphabet Inc. (GOOG) stock market data collected from Yahoo Finance, and to describe the data and make predictions about its future trajectory (which can then be analysed what happened to the stock market). The essential aspect of this research is determining confidence intervals for the predictions. Through the utilisation of data captured across different time periods (”crash scenarios”), the study aims to construct various model types and evaluate the validity of the confidence intervals. The questions addressed is the alignment of actual market data within the confidence intervals.

The study unfolds systematically, starting with a comprehensive literature review in Chapter 2, which explores the methodologies for analysing time-series data. Building upon this foundation, Chapter 3 explains the essential background knowledge concerning Gaussian processes. Chapter 4 integrates the knowledge into the domain of stock market data analysis. In this chapter, the methodology employed to conduct data interpolation and extrapolation using Gaussian processes, implemented with TensorFlow Probability is meticulously outlined. In Chapter 5, the results of the Gaussian process analysis of the stock market data are presented. This chapter not only showcases the obtained outcomes, but also entails a comprehensive analysis and discussion concerning our findings. Finally, chapter 6 concludes our results and offers insights into potential avenues for further studies.

Chapter 2

Literature Reviews

Rasmussen & Williams (2006) highlighted Gaussian processes as a principled and probabilistic approach to kernel machine learning, enabling interpretable predictions and robust model selection. These processes extended probability distributions to functions, providing convenient inference and learning, particularly suited for supervised learning in machine learning. The historical context revealed challenges with neural networks and the search for principled frameworks, leading to insights on Gaussian processes emerging in infinite-sized networks which was verified by Neal (1996), a graduate student at the University of Toronto. The work emphasised the convergence of statistical and machine learning goals, with Gaussian processes bridging the gap, connecting models such as Bayesian linear models and neural networks. Gaussian processes simplified model handling and interpretation, making them powerful for understanding and predicting complex data across various domains (Rasmussen & Williams, 2006).

Rasmussen & Williams (2006) and Li (2018) mentioned two fundamental strategies in supervised learning. The first approach involved assuming a predefined class of functions that the model aimed to learn by tuning its parameters accordingly. The second approach encompassed attributing a prior probability to the entire spectrum of potential functions, followed by selecting the function that optimised training data likelihood to derive the posterior distribution. The second method appeared complicated due to the fact that there are several possible functions available for probability assignment. They explained that this was where Gaussian processes emerged as a powerful solution, as Gaussian processes offered a tractable means to deal with the complexity of functions and probabilities in such scenarios.

In the work of D'Elia et al. (2023), they highlighted a limitation in traditional regression models such as L2 norm, which lacked the capacity to provide insights into the reliability of estimates, a crucial aspect for robust decision-making. They emphasised the importance of confidence intervals in enhancing the validity of findings. The authors pointed out that the multivariate normal distribution offerred a convenient choice for explicit calculations, making it adaptable to Gaussian processes. This concept could be categorised into unsupervised and supervised learning settings.

Duvenaud (2014) expanded upon this notion, reiterating that learning functions from data

encompassed prediction, extrapolation, and induction tasks. He introduced an automated method that constructed, visualised, and characterised a diverse array of models, beneficial for forecasting and uncovering structures in time-series domain. Those models, based on Gaussian processes, exhibited the capability to capture various statistical patterns such as periodicity, changepoints, additivity, and symmetries, enhancing their versatility for diverse applications.

There are diverse approaches in time-series prediction using Gaussian processes (GP). Gonzalves et al. (2019) provided a comprehensive overview of time-series prediction methods employing Gaussian processes, with particular attention to the GP-ARX model, a fusion of ARX (autoregressive) models within the Gaussian process framework. They explored hyperparameters estimation through maximum likelihood and discussed the persistence prediction, where the present value forecasts the next step. The authors underlined the importance of kernel choice and feature selection. They noted that the ARX model became equivalent to the GP-ARX model under linear kernel conditions. By combining exponential and linear kernels and kernel capability, their GP-ARX model showcased potential as a robust interpolator for financial applications (Gonzalves et al., 2019, pp.26-28).

Roelants (2018) provided a brief explanation of the fundamental concept of the multivariate normal distribution underlying Gaussian processes on his GitHub page. In 2019, he extended his contributions by publishing a Gaussian process implementation using TensorFlow Probability (Roelants, 2019). This implementation was also applied to fitting a Mauna Loa CO2 dataset using a combined kernel comprising squared exponential, local periodic, rational quadratic, and noise kernels (Roelants, 2019). During model training, He employed the Adam optimiser to minimise the negative log marginal likelihood.

TensorFlow (2022) has developed a valuable resource, TensorFlow Probability, which covers a comprehensive class for implementing Gaussian process regression. This class was used by Roelants (2019) in his work mentioned previously. This powerful toolkit offers the command `tfp.distributions.GaussianProcess`, enabling the implementation of the marginal distribution of a Gaussian process at discrete points (TensorFlow, 2023). Moreover, TensorFlow Probability facilitates Gaussian process regression through the command `tfp.distributions.GaussianProcessRegressionModel`, which efficiently derives the posterior predictive distribution within a conjugate Gaussian process regression model, thus providing a versatile framework for probabilistic modeling and predictions (TensorFlow, 2023).

In summary, Gaussian processes emerge as a principled approach to kernel machine learning, facilitating interpretable predictions and bridging statistical and machine learning goals (Rasmussen & Williams, 2006). Gaussian processes extend probability distributions to functions, simplifying model interpretation and handling (Rasmussen & Williams, 2006). Rasmussen & Williams (2006) and Li (2018) delved into unsupervised and supervised learning strategies, with Gaussian processes addressing challenges in assigning probabilities to functions and providing confidence intervals. Duvenaud (2014) introduced automated methods using Gaussian processes for time-series data and Gonzalves et al. (2019) focus on Gaussian pro-

cesses in time-series prediction, spotlighting the GP-ARX model. Roelants (2019) and Roelants (2019) showcases Gaussian processes in fitting data using TensorFlow Probability developed by TensorFlow (2022) for Gaussian process regression. These contributions emphasise Gaussian processes' significance in versatile applications, offering interpretability, probabilistic modeling, and robustness.

Making predictions would be meaningless without an uncertainty. Although the traditional machine learning could make great predictions, it often lacks the provision of confidence intervals or uncertainties. Incorporating confidence intervals is crucial for assessing prediction reliability. Therefore, the Gaussian process is a popular choice due to its capability to provide predictions along with associated uncertainties, making it a valuable tool for meaningful and robust decision-making.

Chapter 3

Gaussian Process

In the context of regression problems, the primary objective is to generate predictions for previously unseen input data based on provided training information. However, there are cases of missing data, and there might be a necessity to fill these gaps in the dataset. These tasks are inherently inductive, necessitating assumptions about the characteristics or behaviours of an underlying predictive function. According to Rasmussen & Williams (2006, p.2) and Li (2018, p.10), addressing this challenge involves two primary approaches. The first method is to restrict the type of functions we consider. Another method is to provide a prior probability to every possible function. The higher probabilities are given to functions we consider to be more likely. However, there is a problem in the second approach. Rasmussen & Williams (2006, p.2) pointed out that while there was an infinite set of possible functions, we wanted to compute with this set in finite time. They stated that the Gaussian process could solve this issue.

A Gaussian process extends the scope of the Gaussian probability distribution. It represents a stochastic process composed of random variables adhering to a Gaussian distribution, thereby characterising a system subjects to random variations over time (Rasmussen & Williams, 2006, p.2).

In regression, the "prior" signifies our initial assumptions about the functions we expect to encounter prior to observing any data. The selection of the prior holds significance as it establishes the attributes of functions subject to inference. Consider a (training) data set denoted as $D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)\}$, encompassing n observations. Our goal is to focus on functions that accurately (or closely) align with theses data points. This results in a reduction of uncertainty near the observations. The integration of the prior with the observed data yields the "posterior" distribution encompassing functions (Rasmussen & Williams, 2006, pp.3-4).

Gaussian processes represent distributions over functions $f(x)$, with their distribution defined through a mean function and a positively definite covariance function (Roelants, 2019). The task of learning in Gaussian processes specifically involves determining appropriate attributes for the covariance function. This provides us a data model, offering insights into features like smoothness and characteristic length-scale, which can be interpreted meaningfully

(Rasmussen & Williams, 2006, p.4).

3.1 Stochastic Processes

Stochastic processes commonly characterise systems that undergo random alterations over time. Because of the uncertainty in the systems, the processes are stochastic. Although the initial state is known, numerous pathways exist through which these processes can unfold (Roelants, 2019).

The stochastic process can result in diverse trajectories, often referred to as *realisations* of the process. Roelants (2019) mentioned that a stochastic process could be interpreted as a probabilistic distribution over functions. Nonetheless, the stochastic process's randomness would lead to variations in each realised function.

3.2 Gaussian Processes

Gaussian processes are distributions over functions $f(\mathbf{x})$. These distributions are defined by a mean function, denoted as $m(\mathbf{x})$, and a positive definite covariance function, denoted as $k(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}$ denotes the input vector and $(\mathbf{x}, \mathbf{x}')$ represents all conceivable pairs within the input domain (Roelants, 2019):

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')), \tag{3.1}$$

where the marginal distribution is a multivariate Gaussian distribution for any finite subset $X = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_n\}$ of the domain of $\mathbf{x}$ (Roelants, 2019):

$$f(X) \sim N(m(X),\ k(X, X)) \tag{3.2}$$

with mean vector $\mu = m(X)$ and covariance matrix $\Sigma = k(X, X)$.

Although the multivariate Gaussian encompasses a finite set of Gaussian jointly distributed, the Gaussian process transcends this constraint. Its mean and covariance function are determined by a functional definition. Each input to the functions (mean and covariance functions) is a variable that is correlated with other variables within the input domain, as defined by the covariance function. Given that functions can possess an infinite input domain, the Gaussian process can be interpreted as an infinite dimensional Gaussian random variable (Roelants, 2019).

3.3 Regression

Regression is one of the supervised learning, focusing on forecasting continuous values. An illustrative scenario includes financial contexts, where endeavors involve foretelling stock market prices or currency exchange rates (Rasmussen & Williams, 2006, p.7). In this section, we explain Gaussian process approaches for regression tasks.

Multiple perspectives exist for comprehending Gaussian process regression models. One perspective is treating a Gaussian process as the *weight-space view*. Another angle is to conceptualise a Gaussian process as defining a distribution across functions, with inference occurring in the space of functions (also known as the *function-space view*) (Rasmussen & Williams, 2006, p.7).

3.3.1 Weight-space View

The extensively studied and frequently employed simple linear regression model represents the output as a linear combination of the inputs. In this section, our exploration commences with a Bayesian approach to the linear model. Subsequently, we introduce a straightforward augmentation to this model category by projecting the inputs into a feature space of higher dimensions, followed by applying the linear model within that space. Our focus resides in drawing inferences about the relationship between inputs and targets (the conditional distribution of the targets given the inputs). However, our concern does not encompass the modelling of the input distribution (Rasmussen & Williams, 2006, pp.7-8).

The Standard Linear Model

We will examine the Bayesian analysis of the standard linear regression model accompanied by Gaussian noise (Rasmussen & Williams, 2006, p.8):

$$f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\mathbf{w}, \qquad y = f(\mathbf{x}) + \epsilon, \tag{3.3}$$

where $\mathbf{x}$ represents the input vector, $\mathbf{w}$ is a vector of weights (parameters) of the linear model, f denotes the function value, and y is the observed target value. Note that we have made an assumption that the observed values y deviate from the function values $f(\mathbf{x})$ due to the additional noise. Furthermore, we will continue to assume that the noise adheres to an independent, identically distributed (iid) Gaussian distribution, characterised by a zero mean and a variance of σ_n^2 (Rasmussen & Williams, 2006, p.8):

$$\epsilon \sim N(0,\ \sigma_n^2). \tag{3.4}$$

The conjunction of this noise assumption and the model inherently leads to the *likelihood* – the probability distribution of the observed data given the parameters. This likelihood is decomposed over instances within the training set due to the assumption of independence, resulting in

(Rasmussen & Williams, 2006, pp.8-9):

$$\begin{aligned} p(\mathbf{y} \mid X, \mathbf{w}) &= \prod_{i=1}^{n} p(y_i \mid \mathbf{x}_i, \mathbf{w}) \\ &= \prod_{i=1}^{n} \left[\frac{1}{\sqrt{2\pi}\sigma_n} e^{-\frac{(y_i - \mathbf{x}_i^{\mathrm{T}}\mathbf{w})^2}{2\sigma_n^2}} \right] \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} e^{-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^{\mathrm{T}}\mathbf{w}|^2} \\ &\sim N(X^{\mathrm{T}}\mathbf{w},\ \sigma_n^2 I), \end{aligned} \tag{3.5}$$

where $|\mathbf{z}|$ is the Euclidean magnitude of vector $\mathbf{z}$. In the Bayesian framework, the necessity arises to establish a prior encompassing the parameters, encapsulating our pre-observation presumptions about these parameters. We introduce a Gaussian prior with a mean of zero and a covariance matrix Σ_p assigned to the weights. In the Bayesian linear model, the basis for inference relies on the posterior distribution regarding the weights, estimated through the application of Bayes' rule (Rasmussen & Williams, 2006, p.9):

$$\begin{aligned} \text{posterior} &= \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \\ p(\mathbf{w} \mid \mathbf{y}, X) &= \frac{p(\mathbf{y} \mid X, \mathbf{w}) p(\mathbf{w})}{p(\mathbf{y} \mid X)}, \end{aligned} \tag{3.6}$$

where the marginal likelihood (normalising constant) is integrated over the weights and given by

$$p(\mathbf{y} \mid X) = \int p(\mathbf{y} \mid X, \mathbf{w}) p(\mathbf{w})\, d\mathbf{w}. \tag{3.7}$$

The posterior merges the likelihood and the prior, encapsulating all our parameter-related knowledge. Demonstrably, we can prove that the shape of the posterior distribution is a Gaussian form (Rasmussen & Williams, 2006, p.9):

$$p(\mathbf{w} \mid X, \mathbf{y}) \sim N\left(\frac{1}{\sigma_n^2} A^{-1} X\mathbf{y},\ A^{-1} \right), \tag{3.8}$$

where $A = \sigma_n^{-2} X X^{\mathrm{T}} + \Sigma_p^{-1}$. Note that in this particular model (and for any Gaussian posterior), the mean of the posterior distribution $p(\mathbf{w} \mid \mathbf{y}, X)$ aligns with its mode. This maximum point is also referred to as the maximum a posterior (MAP) estimate of $\mathbf{w}$ (Rasmussen & Williams, 2006, p.9).

Rasmussen & Williams (2006, p.11) described that when making predictions for a test case, we compute an average across various potential parameter values, considering their respective posterior probabilities as weights. As a result, the predictive distribution for $f_* = f(\mathbf{x}_*)$ at $\mathbf{x}_*$ is determined by taking the average output of all conceivable linear models according to the

Gaussian posterior:

$$\begin{aligned} p(f_* \mid \mathbf{x}_*, X, \mathbf{y}) &= \int p(f_* \mid \mathbf{x}_*, \mathbf{w}) p(\mathbf{w} \mid X, \mathbf{y})\, d\mathbf{w} \\ &= N\left(\frac{1}{\sigma_n^2}\mathbf{x}_*^{\mathrm{T}} A^{-1} X\mathbf{y},\ \mathbf{x}_*^{\mathrm{T}} A^{-1}\mathbf{x}_*\right). \end{aligned} \tag{3.9}$$

Rasmussen & Williams (2006, p.11) also emphasised that the predictive distribution is a Gaussian shape. The mean of this distribution is derived by multiplying the posterior mean of the weights from Eq.(3.8) by the test input, aligning with the expectations arising from symmetry analysis. The predictive variance represents a quadratic formulation of the test input using the posterior covariance matrix. They also mentioned that this observation illustrates that predictive uncertainties increase with the magnitude of the test input, as expected for a linear model.

3.3.2 Projection of Inputs into Feature Space

In the preceding subsection, we examined the Bayesian linear model, which encounters constraints in terms of its expressiveness. To tackle this challenge, a straightforward approach involves projecting the inputs into a higher-dimensional space using a set of basis functions. Subsequently, the linear model is employed within this transformed space rather than directly on the original inputs. Implementing this notion raises the query of how to select the appropriate basis functions. This is where the Gaussian process framework comes into play, enabling us to address this inquiry (Rasmussen & Williams, 2006, p.11).

3.3.3 Function-space View

Another approach is considering inference directly within the realm of function space. We employ a Gaussian process to characterise a distribution over functions. Formally, *a Gaussian process consists of an assemblage of random variables, where any finite subset of them collectively adheres to a joint Gaussian distribution* (Rasmussen & Williams, 2006, p.13).

In general, the complete specification of a Gaussian process hinges on its mean function and covariance function. The mean function , $m(\mathbf{x})$, and the covariance function , $k(\mathbf{x}, \mathbf{x}')$, are defined for a real process, $f(\mathbf{x})$, such that (Rasmussen & Williams, 2006, p.13)

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x}], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \tag{3.10}$$

and the Gaussian process can be expressed as

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}),\ k(\mathbf{x}, \mathbf{x}')). \tag{3.11}$$

Typically, for notational convenience, the mean function is assumed to be zero, even though this is not always the case (Rasmussen & Williams, 2006, p.13).

The definition of the Gaussian process inherently implies a *consistency* demand, sometimes referred to as the *marginalisation property*. This property implies that analysing a larger set of variables does not alter the distribution of the smaller subset. It is important to note that the consistency requirement is automatically met when the covariance function specifies the covariance matrix entries (Rasmussen & Williams, 2006, p.13).

Rasmussen & Williams (2006, p.14) noted that an illustration of a Gaussian process can arise from our Bayesian linear regression model, where $f(\mathbf{x}) = \phi(\mathbf{x})^{\mathrm{T}}\mathbf{w}$ with the prior distribution over parameters $\mathbf{w} \sim N(0,\ \Sigma_p)$. This yields the following expressions for the mean and covariance:

$$\begin{aligned} \mathbb{E}[f(\mathbf{x})] &= \phi(\mathbf{x})^{\mathrm{T}}\mathrm{E}[\mathbf{w}] = 0, \\ \mathbb{E}[f(\mathbf{x})f(\mathbf{x}')] &= \phi(\mathbf{x})^{\mathrm{T}}\mathbb{E}[\mathbf{w}\mathbf{w}^{\mathrm{T}}]\phi(\mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\Sigma_p\phi(\mathbf{x}'). \end{aligned} \tag{3.12}$$

Consequently, $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian, with a mean of zero and covariance determined by $\phi(\mathbf{x})^{\mathrm{T}}\Sigma_p\phi(\mathbf{x}')$. In fact, the function values $f(\mathbf{x}_1), f(\mathbf{x}_2), f(\mathbf{x}_3), \ldots, f(\mathbf{x}_n)$ associated with any number of input points n follow a Gaussian distribution. However, if $N < n$, this Gaussian distribution becomes singular due to the joint covariance matrix having a rank of N (Rasmussen & Williams, 2006, p.14).

It can be proved that the squared exponential covariance function

$$\mathrm{cov}(f(\mathbf{x}_p), f(\mathbf{x}_q)) = k(\mathbf{x}_p, \mathbf{x}_q) = e^{-\frac{1}{2}|\mathbf{x}_p - \mathbf{x}_q|^2} \tag{3.13}$$

aligns with a Bayesian linear regression model encompassing an infinite number of basis functions (Rasmussen & Williams, 2006, ch.4). In fact, the specification of the covariance function suggests the formulation of a distribution over functions (Rasmussen & Williams, 2006, p.14).

Prediction with Noise-free Observations

Our primary focus typically is not on generating random functions from the prior; instead, our aim is to integrate the insights derived from the training data into the function. To begin, we will examine the uncomplicated scenario where the observations are devoid of noise, meaning we possess knowledge about $\{(\mathbf{x}_i, f_i) \mid i = 1, \ldots, n\}$. The joint distribution of the training targets, $\mathbf{f}$, and the test targets, $\mathbf{f}_*$, in accordance with the prior, is as follows (Rasmussen & Williams, 2006, p.15):

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim N\left(0,\ \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{3.14}$$

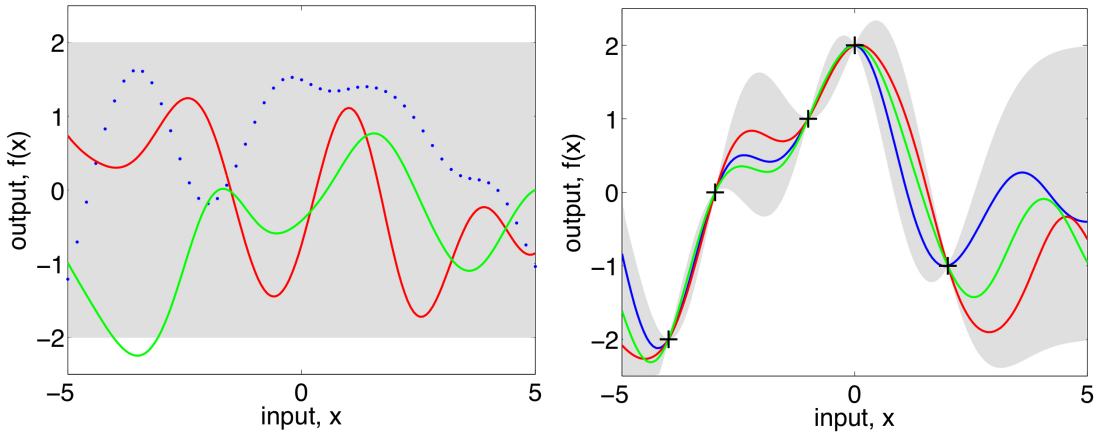
In scenarios involving n training points and n_* test points, the matrix $K(X, X_*)$ represents the covariances computed for all combinations of training and test points. The same applies to the entries $K(X, X)$, $K(X_*, X_*)$, and $K(X_*, X)$. To attain the posterior distribution over functions, it is essential to limit this joint prior distribution exclusively to functions that align with the observations. In probabilistic terms, this procedure is remarkably straightforward, involving the act of conditioning the joint Gaussian prior distribution on the observed data points

(Rasmussen & Williams, 2006, pp.15-16), yielding

$$\mathbf{f}_* \mid X_*, X, \mathbf{f} \sim N\left(K(X_*, X)K(X, X)^{-1}\mathbf{f},\ K(X_*, X_*) - K(X_*, X)K(X, X)^{-1}K(X, X_*)\right). \tag{3.15}$$

Rasmussen & Williams (2006, p.16) emphasised that samples of function values $\mathbf{f}_*$ that are related to test inputs X_* can be drawn from the joint posterior distribution by computing the mean and covariance matrix as specified in Eq.(3.15), and subsequently generating the samples.

They also suggested that the process could be conceptualised as generating functions from the prior and discarding those that do not agree with the observations as shown in Figure 3.1.



(a) Prior (b) Posterior

Figure 3.1: (a) randomly drawn functions from a Gaussian process prior, (b) three samples from the posterior (prior conditioned on five noise-free observations (Rasmussen & Williams, 2006, p.15).

Figure 3.1(a) illustrates three randomly drawn functions from a Gaussian process prior. The dots represent the generated y values, while the other two functions are approximately represented as lines by connecting numerous evaluated points. Figure 3.1(b) illustrates the outcomes of such evaluations (posterior distribution), based on the five marked data points denoted by + symbols. The shaded region is the pointwise mean along with two times the standard deviation for each input value. This interval corresponds to a 95% confidence interval region (Rasmussen & Williams, 2006, p.15).

Prediction with Noisy Observations

In many real-world modeling scenarios, it is common not to directly possess function values, but rather their noisy counterparts expressed as $y = f(\mathbf{x}) + \epsilon$ where ϵ represents additive independent identically distributed Gaussian noise with a variance of σ_n^2. Hence, the prior for the noisy observations can be written as (Rasmussen & Williams, 2006, p.16)

$$\text{cov}(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2\delta_{pq} \qquad \text{or} \qquad \text{cov}(\mathbf{y}) = K(X, X) + \sigma_n^2 I, \tag{3.16}$$

where δ_{pq} is the Kronecker delta defined as

$$\delta_{pq} = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{otherwise.} \end{cases} \tag{3.17}$$

Based on the assumption of noise independence, a diagonal matrix is introduced, which differs from the noise-free scenario presented in Eq.(3.13). By introducing the noise term into Eq.(3.14), the joint distribution of the observed target values and the function values at the test locations under the prior becomes (Rasmussen & Williams, 2006, p.16)

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K(X,X) + \sigma_n^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right). \tag{3.18}$$

Obtaining the conditional distribution associated with Eq.(3.15), we reach the fundamental predictive equations for Gaussian process regression (Rasmussen & Williams, 2006, p.16):

$$\mathbf{f}_* \mid X, \mathbf{y}, X_* \sim N\left(\bar{\mathbf{f}}_*,\ \text{cov}(\mathbf{f}_*)\right), \tag{3.19}$$

where

$$\bar{\mathbf{f}}_* = \mathbb{E}[\mathbf{f}_* \mid X, \mathbf{y}, X_*] = K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y}, \tag{3.20}$$

$$\text{cov}(\mathbf{f}_*) = K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2 I]^{-1}K(X,X_*). \tag{3.21}$$

We can calculate the corresponding covariance function as $k(\mathbf{x}_p, \mathbf{x}_q) = \phi(\mathbf{x}_p)^{\mathrm{T}}\Sigma_p\phi(\mathbf{x}_q)$, where $\phi(\mathbf{x})$ is the function that maps an input vector $\mathbf{x}$ from a D-dimensional space into an N-dimensional feature space (Rasmussen & Williams, 2006, p.11), for any set of basis functions. In contrast, there is a potentially infinite expansion expressed in terms of basis functions for each positive definite covariance function k (Rasmussen & Williams, 2006, p.17).

Rasmussen & Williams (2006, p.17) noted the formulas including $K(X,X)$, $K(X,X_*)$, and $K(X_*,X)$ might appear complex, thus they introduced a concise notation by setting $K = K(X,X)$ and $K_* = K(X,X_*)$. In the scenario where a single test point $\mathbf{x}_*$ is involved, they denoted the vector of covariances between the test point and the n training points as $\mathbf{k}(\mathbf{x}_*) = \mathbf{k}_*$. With this compacted notation, the expressions in Eq.(3.20) and Eq.(3.21) are simplified to

$$\bar{f}_* = \mathbf{k}_*^{\mathrm{T}}(K + \sigma_n^2 I)^{-1}\mathbf{y}, \tag{3.22}$$

$$\mathbb{V}[f_*] = k(\mathbf{x}_*,\mathbf{x}_*) - \mathbf{k}_*^{\mathrm{T}}(K + \sigma_n^2 I)^{-1}\mathbf{k}_*. \tag{3.23}$$

Note that the mean predictive distribution defined by Eq.(3.22) is a linear combination of the observed values $\mathbf{y}$. This is occasionally termed a *linear predictor* (Rasmussen & Williams, 2006, p.17). Another perspective on this equation is to interpret it as a linear combination of n kernel functions, each centered on a training point (Rasmussen & Williams, 2006, p.17). This

can be expressed by

$$\bar{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x}_*) \tag{3.24}$$

where $\boldsymbol{\alpha} = (K + \sigma_n^2 I)^{-1}\mathbf{y}$. Although the Gaussian process can be expressed using a potentially infinite number of basis functions, the mean prediction for $f(\mathbf{x}_*)$ can be represented as Eq.(3.24). This can be attributed to the *representer theorem*. The reason behind this result can be grasped intuitively: Despite the Gaussian process describing a joint Gaussian distribution for all y variables, each corresponding to a point in the index set X, the $(n+1)$-dimensional distribution formed by the n training points and the test point is only be considered for making predictions at $\mathbf{x}_*$. It becomes evident that conditioning this $(n+1)$-dimensional distribution on the observations leads us to the desired outcome, since Gaussian distributions can be marginalised by focusing on relevant blocks of the joint covariance matrix (Rasmussen & Williams, 2006, pp.17-18).

In addition, Rasmussen & Williams (2006, p.18) claimed that the variance in Eq.(3.21) is independent of the observed targets, relying only on the inputs. This behaviour is the nature of the Gaussian distribution. The variance consists of two components: the initial term $K(X_*, X_*)$ corresponds to the prior covariance, and subtracted a positive term that represents the information of the observations on the function. Calculating the predictive distribution of test targets $\mathbf{y}_*$ can be straightforwardly achieved by adding $\sigma_n^2 I$ to the variance, $\text{cov}(\mathbf{f}_*)$.

Indeed, Eq.(3.20) and Eq.(3.21) are the mean function and covariance function of the Gaussian posterior process, respectively, according to the Gaussian process definition. The posterior and posterior covariance are shown in Figure 3.2. Note that the covariance is substantial for nearby points, decreasing to zero at the training points where variance is vanish since it is a noise-free process, and then turning negative beyond. This pattern arises due to the fact that if the smooth function lies below the mean on one side of the data point, it tends to rise above the mean on the opposite side, leading to a change in the sign of the covariance at those data points. In contrast, it is important to highlight that the prior covariance follows a Gaussian shape and remains non-negative throughout (Rasmussen & Williams, 2006, p.18).

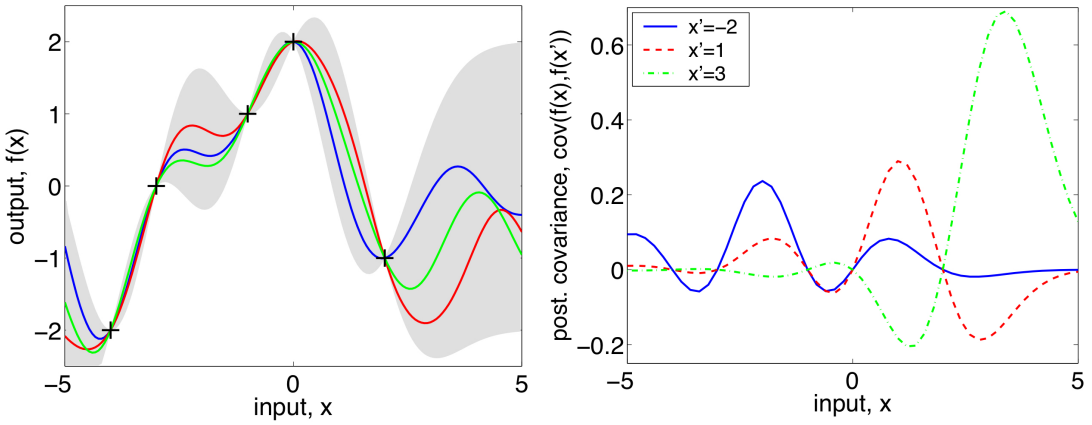
The concept of the *marginal likelihood* (or *evidence*), $p(\mathbf{y}|X)$, which is defined as the product of the integral of the likelihood and prior distribution:

$$p(\mathbf{y} \mid X) = \int p(\mathbf{y} \mid \mathbf{f}, X) p(\mathbf{f} \mid X)\, d\mathbf{f}, \tag{3.25}$$

refers to the marginalisation over function values, $\mathbf{f}$. In the Gaussian process model, the prior distribution is Gaussian, expressed as $\mathbf{f} \mid X \sim N(0, K)$, where K represents the covariance matrix. Hence, we can write (Rasmussen & Williams, 2006, pp.18-19)

$$\log p(\mathbf{f} \mid X) = -\frac{1}{2}\mathbf{f}^{\mathrm{T}} K^{-1}\mathbf{f} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi. \tag{3.26}$$

Furthermore, Rasmussen & Williams (2006, p.19) showed that the likelihood follows a fac-



(a) Posterior

(b) Posterior covariance

Figure 3.2: (a) three random functions sampled from the posterior distribution, (b) posterior covariance variation for $f(\mathbf{x})$ and $f(\mathbf{x}')$ for the same data at three different $\mathbf{x}'$ values (Rasmussen & Williams, 2006, p.18).

torised Gaussian distribution, $\mathbf{y} \mid \mathbf{f} \sim N(\mathbf{f}, \sigma_n^2 I)$, which enables us to carry out the integration, resulting in the log marginal likelihood:

$$\log p(\mathbf{y} \mid X) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}}(K + \sigma_n^2 I)^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2 I| - \frac{n}{2}\log 2\pi. \tag{3.27}$$

Note that this result can also be directly derived by recognizing that $\mathbf{y}$ follows a Gaussian distribution, $\mathbf{y} \sim N(0, K + \sigma_n^2 I)$ (Rasmussen & Williams, 2006, p.19).

In practice, the implementation of Gaussian process regression is depicted in Algorithm 1, outlined by Rasmussen & Williams (2006, p.19). Their algorithm employs Cholesky decomposition instead of matrix inversion due to its efficiency and numerical stability. The algorithm returns the predictive mean and variance for noise-free test data. However, to calculate the predictive distribution for noisy test data y_*, the noise variance σ_n^2 could be simply added to the predictive variance of f_*.

3.3.4 Decision Theory for Regression

So far, we have explored the methodology to calculate predictive distributions for the outputs y_* that correspond to new test inputs $\mathbf{x}_*$. These predictive distributions adhere to a Gaussian distribution with mean and variance defined by Eq.(3.22) and Eq.(3.23). However, in practical scenarios, we frequently encounter situations that demand decision-making, such as when we require a precise prediction that optimises certain criteria. To address this, we necessitate a loss function, denoted as $L(y_{\text{true}}, y_{\text{guess}})$, which stipulates the penalty (or loss) obtained by estimating the values y_{guess} when the actual value is y_{true}. This loss function could, for instance, be defined as the absolute deviation between the estimated and actual values (Rasmussen & Williams, 2006, pp.21-22).

Algorithm 1 Gaussian process regression (Rasmussen & Williams, 2006, p.19)

procedure PREDICTIONS AND LOG MARGINAL LIKELIHOOD FOR GAUSSIAN PROCESS REGRESSION
  input: X (inputs), $\mathbf{y}$ (targets), k (covariance function), σ_n^2 (noise level), $\mathbf{x}_*$ (test input)
  $L := \text{cholesky}(K + \sigma_n^2 I)$
  $\boldsymbol{\alpha} := L^{\mathrm{T}} \backslash (L \backslash \mathbf{y})$ predictive mean Eq.(3.22)
  $\bar{f}_* := \mathbf{k}_*^{\mathrm{T}} \boldsymbol{\alpha}$ predictive mean Eq.(3.22)
  $\mathbf{v} := L \backslash \mathbf{k}_*$ predictive variance Eq.(3.23)
  $\mathbb{V}[f_*] := k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{v}^{\mathrm{T}}\mathbf{v}$ predictive variance Eq.(3.23)
  $\log p(\mathbf{y} \mid X) := -\frac{1}{2}\mathbf{y}^{\mathrm{T}}\boldsymbol{\alpha} - \Sigma_i \log L_{ii} - \frac{n}{2}\log 2\pi$ Eq.(3.27)
  return: $\bar{f}_*$ (mean), $\mathbb{V}[f_*]$ (variance), $\log p(\mathbf{y} \mid X)$ (log marginal likelihood)
end procedure

Observe that our computation of the predictive distribution was carried out independently of the loss function. In non-Bayesian frameworks, models are usually trained by minimising the loss. On the other hand, in the Bayesian learning, a distinction exists between the likelihood function, that is used for training together with the prior, and the loss function. The role of the likelihood function is to explain how the noisy measurements are postulated to diverge from the noise-free underlying function. Conversely, the loss function encapsulates the results of opting for a specific choice when provided the actual state. It is noteworthy that the likelihood and the loss function are not necessarily required to share any common characteristics (Rasmussen & Williams, 2006, p.22).

Our objective is to attain the point prediction y_{guess} that results in the minimal loss. However, this proves challenging when we lack the knowledge of y_{true}. Alternatively, Rasmussen & Williams (2006, p.22) suggested that we embark on the path of minimising the expected loss by taking an average with respect to the perspective of our model regarding what the actual truth could entail:

$$\widetilde{R}_L(y_{\text{guess}} \mid \mathbf{x}_*) = \int L(y_*, y_{\text{guess}}) p(y_* \mid \mathbf{x}_*, D)\, dy_*. \tag{3.28}$$

Therefore, our optimal estimate (best guess), which aims to minimise the expected loss, is given by

$$y_{\text{optimal}} \mid \mathbf{x}_* = \underset{y_{\text{guess}}}{\operatorname{arcmin}} \widetilde{R}_L(y_{\text{guess}} \mid \mathbf{x}_*). \tag{3.29}$$

Typically, the value of y_{guess} that effectively reduces the risk associated with the loss function $|y_{\text{guess}} - y_*|$ is the median of $p(y_* \mid \mathbf{x}_*, D)$, whereas it corresponds to the mean of the distribution for the squared loss $(y_{\text{guess}} - y_*)^2$. In cases where the predictive distribution is Gaussian, the mean and median are the same. This holds true for any symmetric loss function and symmetric predictive distribution. Nevertheless, in numerous real-world scenarios, the loss functions could be asymmetrical, and point predictions could be directly calculated using Eq.(3.28) and Eq.(3.29) instead (Rasmussen & Williams, 2006, p.22).

3.4 Covariance Functions

The significance of the covariance function (or kernel) in a Gaussian process predictor lies in its role of encoding our assumptions about the target function being learned. In supervised learning, the concept of *similarity* between data points is fundamental, where the proximity of input values $\mathbf{x}$ often implies analogous target values y. This assumption indicates that neighboring training points hold valuable insights for predicting outcomes at nearby test points. Within the Gaussian process framework, the covariance function takes on the task of quantifying this notion of closeness or similarity. When the kernel is combined with the mean function, the kernel fully characterises a Gaussian process (Rasmussen & Williams, 2006, p.79).

For a kernel function to be valid, the resulting kernel matrix $\Sigma = k(X, X)$, have to be positive definite. This entails that the matrix is symmetric and invertible (Roelants, 2019).

A stationary covariance function, dependent on the difference between $\mathbf{x}$ and $\mathbf{x}'$, remains unchanged (invariant) when input space is translated. This property holds for covariance functions like the squared exponential covariance function from Eq.(3.13). Moreover, an isotropic covariance function depends only on $|\mathbf{x} - \mathbf{x}'|$, making it invariant to all rigid transformations. The squared exponential covariance function is also isotropic. In this case, when k depends solely on $r = |\mathbf{x} - \mathbf{x}'|$, they are referred to as *radial basis functions* (RBFs) (Rasmussen & Williams, 2006, pp.79-80).

3.4.1 Examples of Covariance Functions

In this section, we will explore some covariance functions (kernels) in more detail.

White Noise Kernel

A white noise kernel is independent and identically distributed (iid) noise into the Gaussian process distribution:

$$k_{\text{Noise}}(X, X) = \sigma^2 I, \tag{3.30}$$

where σ^2 is the variance of the noise and I is an $n \times n$ identity matrix. Eq.(3.30) gives a covariance matrix primarily composed of zeros, with the diagonal containing variances of individual random variables. The absence of covariances between samples is due to the uncorrelated nature of the noise (Roelants, 2019).

Linear Kernel

A linear kernel function is characterised by its form, which follows a specific mathematical structure (Li, 2018, p.25) and (Duvenaud, 2014):

$$k_{\text{Linear}}(\mathbf{x}, \mathbf{x}') = \frac{\mathbf{x} \cdot \mathbf{x}'}{l} \qquad \text{or} \qquad k(\mathbf{x}, \mathbf{x}') = \sigma_1^2 + \sigma_2^2(\mathbf{x} - \mathbf{h})(\mathbf{x}' - \mathbf{h}), \tag{3.31}$$

where l is a parameter which defines the *characteristic length-scale* and $\mathbf{h}$ is an offset that determines the x-coordinate at which all the lines in the posterior intersect. This results in the function having zero variance at that particular point, unless noise is introduced. σ_1^2 and σ_2^2 are variances. The value of the constant variance σ_1^2 determines the extent to which the function's height will deviate from 0 at the origin. This might seem slightly confusing, as it does not directly specify the value, but rather introduces a prior probability for it. Essentially, it represents the addition of an uncertain offset to the model. Furthermore, the linear kernel can also be derived from Bayesian linear regression (Duvenaud, 2014).

If a covariance function is solely dependent on $\mathbf{x}$ and $\mathbf{x}'$ through $\mathbf{x} \cdot \mathbf{x}'$, it is referred to as a *dot product* covariance function. An example is $k(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}' + \sigma_0^2$, which can be derived from linear regression by assigning $N(0, 1)$ priors to the coefficients of x_d, where $d = 1, ..., D$, and an $N(0, \sigma_0^2)$ prior to the bias (or constant function) 1, as shown in Eq.(3.12) (Rasmussen & Williams, 2006, p.80).

Priors derived from a Gaussian process using the linear kernel will yield linear functions. While the linear kernel's expressive capacity is limited, when combined with non-linear kernels, it can effectively capture both overall trends and local fluctuations in the target function (Li, 2018, p.25).

Squared Exponential Kernel

A squared exponential kernel (often referred to as the exponentiated quadratic kernel, Gaussian kernel, or radial basis function kernel) is a widely used kernel in Gaussian process modeling (Roelants, 2019). It is defined as

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2l^2}}, \tag{3.32}$$

where σ^2 is the overall variance (or σ is referred to as amplitude) and l is the characteristic length-scale. The covariance function is infinitely differentiable, indicating that the associated Gaussian process possesses mean square derivatives of all orders. As a result, it is very smooth (Rasmussen & Williams, 2006, p.83).

Rational Quadratic Kernel

A rational quadratic kernel is given by (Roelants, 2019)

$$k_{\text{RQ}}(\mathbf{x}, \mathbf{x}') = \sigma^2 \left(1 + \frac{|\mathbf{x} - \mathbf{x}'|^2}{2\alpha l^2}\right)^{-\alpha}, \tag{3.33}$$

where σ^2 is the overall variance (or σ is referred to as amplitude), l is the length-scale, and α is a *scale mixture*. The rational quadratic kernel leads to a relatively smooth prior on functions drawn from the Gaussian process. The rational quadratic kernel with α and $l > 0$ can be imagined as a scale mixture (an infinite sum) of squared exponential kernels with varying length-

scales (Rasmussen & Williams, 2006, p.87), where α determines the weighting between these length-scales. As α approaches infinity, the rational quadratic kernel approaches the squared exponential kernel (Duvenaud, 2014).

Periodic Kernel

The periodic kernel serves as an interesting illustration of a non-stationary covariance function. It employs a non-linear mapping (or wrapping) $\mathbf{u}(\mathbf{x})$ of the input $\mathbf{x}$, followed by the application of a stationary covariance function in the u-space. This particular kernel involves the one-dimensional input variable x being mapped to the two-dimensional space $\mathbf{u}(x) = (\cos x, \sin x)$ (Rasmussen & Williams, 2006, p.92). If the squared exponential kernel is utilised in the $\mathbf{u}$-space, we obtain the periodic kernel as (Roelants, 2019)

$$k_{\text{Periodic}}(\mathbf{x}, \mathbf{x}') = \sigma^2 e^{-\frac{2}{l^2}\sin^2\left(\frac{\pi|\mathbf{x}-\mathbf{x}'|}{p}\right)}, \tag{3.34}$$

where p is a period. The periodic covariance function is employed to effectively model functions with periodic behaviour (Roelants, 2019).

3.4.2 Creating New Kernels from Old

A (new) combined kernel could be made by summing and/or multiplying existing kernels.

Combining Kernels by Summation

Kernels can be combined through summation, which involves element-wise addition of their corresponding covariance matrices. This implies that the covariances of the combined kernels will be low if both individual covariances are low (Roelants, 2019). It can be proven that the summation of two kernels results in another kernel (see Appendix B.1).

Combining Kernels by Multiplication

Kernels or covariance functions can be combined by element-wise multiplication, resulting in covariances that are high only when both kernels have high covariances (Roelants, 2019). It can also be proven that the multiplication of two kernels results in another kernel (see Appendix B.2).

Table 3.1, created by Rasmussen & Williams (2006, p.94), provides a summary of the covariance functions that have been explored, along with several other widely used kernel functions.

Covariance Function	Expression
White noise	$\sigma_n^2 I$
Constant	σ_0^2
Linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x'_d$
Squared exponential	$e^{-\frac{\lvert \mathbf{x}-\mathbf{x}' \rvert^2}{2l^2}}$
Matérn class	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{l} \lvert \mathbf{x}-\mathbf{x}' \rvert \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{l} \lvert \mathbf{x}-\mathbf{x}' \rvert \right)$
Exponential	$e^{-\frac{\lvert \mathbf{x}-\mathbf{x}' \rvert}{l}}$
γ-exponential	$e^{-\left(\frac{\lvert \mathbf{x}-\mathbf{x}' \rvert}{l} \right)^{\gamma}}$
Rational quadratic	$\left(1 + \frac{\lvert \mathbf{x}-\mathbf{x}' \rvert^2}{2\alpha l^2} \right)^{-\alpha}$

Table 3.1: Summary of common kernel functions and their expressions (Rasmussen & Williams, 2006, p.94).

3.5 Model Selection

In previous sections, we have explored Gaussian process regression using predefined fixed covariance functions. Yet, real-world scenarios often pose challenges in precisely specifying all aspects of the covariance function. While certain traits like stationarity can be straightforwardly identified, other properties, including hyperparameters such as length-scales, often remain vaguely known. To transform Gaussian processes into effective practical tools, it becomes crucial to develop techniques that tackle the problem of model selection (Rasmussen & Williams, 2006, p.105).

The concept of model selection encompasses both discrete choices and the tuning of continuous hyperparameters associated with covariance functions. This process aids in enhancing the model's predictions and provides valuable insights to users regarding the data's characteristics (Rasmussen & Williams, 2006, p.106).

Various families of covariance functions offer numerous possibilities. Each family generally involves free *hyperparameters* that need to be determined. The process of selecting a covariance function for a specific application entails adjusting hyperparameters within a family and comparing across various families. These challenges are addressed by similar methods, eliminating the need to differentiate between them. Thus, the term "model selection" encompasses both aspects. The choice of covariance function and its associated parameters will be referred as the training of a Gaussian process (Rasmussen & Williams, 2006, p.106).

In essence, the need is to compare different methods, which could vary in parameters or covariance function shape, or even compare a Gaussian process model to other models. While numerous approaches exist in the literature, three overarching principles generally address model selection comprehensively (Rasmussen & Williams, 2006, p.108): (1) calculate the probability of the model given the data, (2) evaluate the generalisation error and (3) bound the generalisation error. The *generalisation error* refers to the average error on new, unseen test examples from the same distribution as the training data. It is important to note that the training error often is not a

reliable indicator of generalisation performance, as a model might over-fit by fitting noise in the training set, resulting in low training error but poor generalisation performance (Rasmussen & Williams, 2006, p.108).

In the next subsection, the Bayesian view on model selection will be explained. This will include calculating of the probability of the model given the data using the marginal likelihood. Then cross-validation, that evaluates the generalisation performance, will be described. These two approaches will be applied to Gaussian process models in the subsequent part of this section (Rasmussen & Williams, 2006, p.108).

3.5.1 Bayesian Model Selection

A hierarchical model specification is widely used. The lower level encompasses the parameters $\mathbf{w}$, followed by the middle level with hyperparameters $\boldsymbol{\theta}$ that control the parameters' distribution at the lower level. Finally, at the highest level, there might be a discrete set of potential model structures, denoted as H_i, being considered (Rasmussen & Williams, 2006, p.108).

The process of Bayesian inference can be initially described in a systematic manner, followed by an intuitive discussion. Inference operates in a step-by-step manner, employing the principles of probability theory. At the foundational level, the posterior over the parameters is determined by applying Bayes' rule (Rasmussen & Williams, 2006, pp.108-109):

$$p(\mathbf{w} \mid \mathbf{y}, X, \boldsymbol{\theta}, H_i) = \frac{p(\mathbf{y} \mid X, \mathbf{w}, H_i)p(\mathbf{w} \mid \theta, H_i)}{p(\mathbf{y} \mid X, \boldsymbol{\theta}, H_i)}, \tag{3.35}$$

where $p(\mathbf{y} \mid X, \mathbf{w}, H_i)$ is the *likelihood* and $p(\mathbf{w} \mid \boldsymbol{\theta}, H_i)$ is the *prior* over parameters. The prior encapsulates our understanding (as a probability distribution) of the parameters before observing data. When there is limited prior knowledge, a wide prior distribution is chosen. The posterior combines prior and data information through the likelihood. The normalising constant in the denominator of Eq.(3.35), $p(\mathbf{y} \mid X, \boldsymbol{\theta}, H_i)$, is called the *marginal likelihood* or *evidence*. It's independent of parameters, $\mathbf{w}$, and can be expressed as (Rasmussen & Williams, 2006, p.109)

$$p(\mathbf{y} \mid X, \boldsymbol{\theta}, H_i) = \int p(\mathbf{y} \mid X, \mathbf{w}, H_i)p(\mathbf{w} \mid \boldsymbol{\theta}, H_i)\, d\mathbf{w}. \tag{3.36}$$

This quantifies the likelihood of the observed data given the model parameters and their prior distribution. The marginal likelihood provides a measure of how well the model explains the observed data. Moving up the hierarchy, we similarly represent the posterior for the hyperparameters, with the marginal likelihood from the previous level serving as the new likelihood (Rasmussen & Williams, 2006, p.109):

$$p(\boldsymbol{\theta} \mid \mathbf{y}, X, H_i) = \frac{p(\mathbf{y} \mid X, \boldsymbol{\theta}, H_i)p(\boldsymbol{\theta} \mid H_i)}{p(\mathbf{y} \mid X, H_i)}, \tag{3.37}$$

where $p(\boldsymbol{\theta} \mid H_i)$ represents the *hyper-prior* (prior for the hyperparameters). The normalising

constant is determined by (Rasmussen & Williams, 2006, p.109)

$$p(\mathbf{y} \mid X, H_i) = \int p(\mathbf{y} \mid X, \boldsymbol{\theta}, H_i) p(\boldsymbol{\theta} \mid H_i) \, d\boldsymbol{\theta}. \tag{3.38}$$

At the highest level of the hierarchical model, we calculate the posterior distribution for the model (Rasmussen & Williams, 2006, p.109):

$$p(H_i \mid \mathbf{y}, X) = \frac{p(\mathbf{y} \mid X, H_i) p(H_i)}{p(\mathbf{y} \mid X)}, \tag{3.39}$$

where $p(\mathbf{y} \mid X) = \Sigma_i p(\mathbf{y} \mid X, H_i) p(H_i)$. We can see that in the process of Bayesian inference, we move through multiple levels of a hierarchical model. At the lowest level, we compute the posterior over the model parameters using Bayes' rule, considering the likelihood of the data given the parameters and the prior over the parameters. Similarly, at the next level, we compute the posterior over hyperparameters, treating the marginal likelihood from the previous level as the likelihood. At the highest level, we determine the posterior for the model itself by considering the relative probabilities of different model structures given the data (Rasmussen & Williams, 2006, pp.108-109).

Note that implementing Bayesian inference involves evaluating integrals, which might require approximations or specialised methods such as Markov chain Monte Carlo (MCMC). In practice, it can be challenging to compute the integral in Eq.(3.38). Therefore, an approximation called type II maximum likelihood (ML-II) is often used, involving maximising the marginal likelihood in Eq.(3.36) with respect to hyperparameters, $\boldsymbol{\theta}$, (Rasmussen & Williams, 2006, p.109).

Rasmussen & Williams (2006, p.110) explained that the choice of prior over model structures, H_i, in Eq.(3.39) is typically flat, meaning no particular model is favored a priori. They also added that the marginal likelihood, particularly in Eq.(3.36), plays a pivotal role in Bayesian inference by inherently balancing the trade-off between model fit and model complexity. This property makes the marginal likelihood essential for solving the model selection problem.

The trade-off between fitting the data and model complexity is intrinsic in Bayesian inference. Unlike other methods where an external parameter needs to be set to balance this trade-off, in Bayesian inference, the automatic adjustment is built in. Even when priors are flat over complexity, the marginal likelihood naturally leans towards selecting the simplest model that adequately explains the data. This means that the marginal likelihood can effectively guide the choice of a model complexity that matches the data well (Rasmussen & Williams, 2006, p.111).

3.5.2 Cross-validation

Now we will explore the use of cross-validation (CV) methods for model selection. The core concept involves dividing the training set into two separate sets: the training set and the *validation* set, which is used to test the performance of the model. The performance on the validation

set serves as an estimate of the generalisation error, guiding the model selection process (Rasmussen & Williams, 2006, p.111).

The hold-out method has limitations as it only uses a fraction of the data for training, potentially leading to high variance in the performance estimate, especially with a small validation set. To address this, k-fold cross-validation is commonly employed. In k-fold cross-validation, the data is divided into k equally sized subsets. One subset is used for validation while the others are used for training, and this process is repeated k times. This approach allows a larger portion of the data for training without sacrificing performance evaluation. Common values for k range from 3 to 10 (Rasmussen & Williams, 2006, p.111).

A special case of k-fold CV is *leave-one-out* cross-validation (LOO-CV), where each training case is treated as a validation case in turn. While LOO-CV can be computationally expensive, some models like Gaussian process regression offer computational shortcuts to mitigate this (Rasmussen & Williams, 2006, p.111).

Cross-validation is flexible in terms of loss functions. Although squared error loss is prevalent in regression, other loss functions can be employed. For probabilistic models like Gaussian processes, cross-validation can also be performed using the negative log probability loss (Rasmussen & Williams, 2006, p.112).

3.5.3 Model Selection for Gaussian Process

This section demonstrates the application of Bayesian inference and cross-validation to Gaussian process regression, specifically with Gaussian noise (Rasmussen & Williams, 2006, p.112).

Marginal Likelihood

Although Bayesian principles offer a coherent framework for inference, many machine learning models pose challenges due to the complexity of required computations. Gaussian process regression models with Gaussian noise stand out as an exception, as they allow analytically tractable integrals over parameters. In this subsection, the application of Bayesian inference principles to Gaussian process models will be explored, particularly in a simplified form where hyperparameters are optimised. The marginal likelihood expressions will be derived and interpreted (Rasmussen & Williams, 2006, p.112).

Due to Gaussian process models being non-parametric, understanding their parameters is not immediately straightforward. One perspective considers the noise-free latent function values at training inputs as parameters. Alternatively, the weight-space view treats parameters as weights for the linear model using basis functions, that could be chosen as the eigenfunctions of the covariance function. However, this approach might become complicated with non-degenerate covariance functions because there would be an infinite number of weights (Rasmussen & Williams, 2006, p.112).

Applying Eq.(3.35) and Eq.(3.36) for the first level of inference, the predictive distribution and marginal likelihood are computed. This leads to the result (Rasmussen & Williams, 2006,

pp.112-113)

$$\log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}} K_y^{-1}\mathbf{y} - \frac{1}{2}\log|K_y| - \frac{n}{2}\log 2\pi, \tag{3.40}$$

where $K_y = K_f + \sigma_n^2 I$ is the covariance matrix for noisy targets $\mathbf{y}$, and K_f is the covariance matrix for the noise-free latent $\mathbf{f}$, then the marginal likelihood conditioned on the hyperparameters (covariance function parameters) $\boldsymbol{\theta}$ is explicitly expressed. This expression clarifies the reason behind referring to Eq.(3.40) as the log marginal likelihood. This is because it is obtained from the process of marginalisation over the latent function (Rasmussen & Williams, 2006, p.113).

To optimise the hyperparameters through maximum marginal likelihood such that $\theta = \underset{\theta}{\operatorname{argmax}}\ \log p(\mathbf{y} \mid X, \theta)$, we calculate the partial derivatives of the marginal likelihood with respect to the hyperparameters on Eq.(3.40) (Rasmussen & Williams, 2006, p.114):

$$\begin{aligned} \frac{\partial}{\partial \theta_j}\log p(\mathbf{y} \mid X, \boldsymbol{\theta}) &= \frac{1}{2}\mathbf{y}^{\mathrm{T}} K^{-1}\frac{\partial K}{\partial \theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\operatorname{tr}\left(K^{-1}\frac{\partial K}{\partial \theta_j}\right) \\ &= \frac{1}{2}\operatorname{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}} - K^{-1})\frac{\partial K}{\partial \theta_j}\right), \end{aligned} \tag{3.41}$$

where $\boldsymbol{\alpha} = K^{-1}\mathbf{y}$ and tr stands for the *trace operation*, which calculates the sum of the diagonal terms of a matrix or the diagonal elements of a product of matrices. The computation of the marginal likelihood in Eq.(3.40) is primarily influenced by the matrix inversion of the K matrix. The logarithm of the determinant of matrix K can be calculated as a secondary result of the inversion process. Inverting a positive definite symmetric matrix with standard methods takes $O(n^3)$ time for an $n \times n$ matrix. After obtaining K^{-1}, calculating the derivatives in Eq.(3.41) only requires $O(n^2)$ time per hyperparameter. Therefore, the computational overhead of estimating derivatives is minimal, making the use of a gradient-based optimiser beneficial (Rasmussen & Williams, 2006, pp.114-115).

In each iteration of the model training phase, the parameter θ_j is updated using a gradient descent approach (Li, 2018, p.32):

$$\theta_j^t = \theta_j^{t-1} - \eta\frac{\partial}{\partial \theta_j}\log p(\mathbf{y} \mid X, \boldsymbol{\theta}), \tag{3.42}$$

where η is the *learning rate*.

The historical practice of estimating $\boldsymbol{\theta}$ by optimising the marginal likelihood has a strong foundation in spatial statistics. As the dataset size, n, grows, the hope is for data to provide more information about $\boldsymbol{\theta}$. Nevertheless, it is important to differentiate between fixed-domain asymptotics (dense observations within a specific region) and increasing-domain asymptotics (observation region grows with n). While increasing-domain asymptotics are suitable for time-series contexts, fixed-domain asymptotics seem more fitting for spatial and machine learning scenarios (Rasmussen & Williams, 2006, p.115).

The marginal likelihood might have multiple local optima, although this issue is generally not severe in the case of simple covariance functions. Practical observations suggest that while

local maxima do exist, they may not significantly hinder the optimisation process. Each local maximum corresponds to a specific interpretation of the data within the Bayesian framework. Bayesian principles suggest considering predictions from various explanations based on their posterior probabilities. For larger datasets, a dominant local optimum often emerges, making the need to average alternative explanations less necessary. Nonetheless, we should be careful to avoid getting stuck in an unfavorable local optimum (Rasmussen & Williams, 2006) (Rasmussen & Williams, 2006, p.115). However, to mitigate this concern, (D'Elia et al., 2023, p.323) pointed out that Simulated Annealing appears as a valuable technique, employing a Markov Chain Monte Carlo approach. Nevertheless, even within this framework, guaranteeing convergence to the global optimum remains a challenging task.

Cross-validation

Rasmussen & Williams (2006, p.116) described that the predictive log probability when excluding training case i is given by

$$\log p(y_i \mid X, \mathbf{y}_{-i}, \boldsymbol{\theta}) = -\frac{1}{2}\log \sigma_i^2 - \frac{(y_i - \mu_i)^2}{2\sigma_i^2} - \frac{1}{2}\log 2\pi, \tag{3.43}$$

where $\mathbf{y}_{-i}$ denotes all targets except the i-th one. The mean (μ_i) and variance (σ_i^2) are calculated using Eq.(3.20) and Eq.(3.21), respectively, with training sets defined as $(X_{-i}, \mathbf{y}_{-i})$. This results in the LOO (leave-one-out) log predictive probability given by (Rasmussen & Williams, 2006, p.116)

$$L_{\text{LOO}}(X, \mathbf{y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \log p(y_i \mid X, \mathbf{y}_{-i}, \boldsymbol{\theta}). \tag{3.44}$$

Rasmussen & Williams (2006, p.117) explained that the LOO (leave-one-out) log predictive probability, L_{LOO}, represented by the Eq.(3.44), is often referred to as the log *pseudo-likelihood*. In each of the n LOO-CV rotations, the inference in the Gaussian process model with fixed hyperparameters involves calculating the inverse covariance matrix. This allows the evaluation of predictive mean and variance using Eq3.20) and Eq.(3.21), respectively. Unlike parametric models, there is no parameter fitting required. The significant realisation is that when applying predictions repetitively as in Eq.(3.20) and Eq.(3.21), the expressions remain nearly identical. They suggested that this is due to the need to compute inverses of covariance matrices with a single column and row that are removed in each step. This operation can be performed effectively by utilising partitioning-based inversion techniques (see Eq.(A.1-A.2)). The expressions for the LOO-CV predictive mean and variance can be derived from these considerations, and hence becomes

$$\begin{aligned} \mu_i &= y_i - \frac{[K^{-1}\mathbf{y}]_i}{[K^{-1}]_{ii}}, \\ \sigma_i^2 &= \frac{1}{[K^{-1}]_{ii}}. \end{aligned} \tag{3.45}$$

Rasmussen & Williams (2006, p.117) also emphasised that it becomes evident that, in fact, the mean μ_i is independent of y_i, as it should be. The computational cost of computing these quantities is $O(n^3)$ for computing the inverse of K plus an additional $O(n^2)$ for the entire LOO-CV procedure when K^{-1} is known. Hence, the computational overhead for the LOO-CV quantities is minimal. By substituting these expressions into Eq.(3.43) and Eq.(3.44), we obtain a performance estimator that can be optimised with respect to hyperparameters for model selection. Specifically, we can calculate the partial derivatives of L_{LOO} with respect to the hyperparameters (applying Eq.(A.3)) and utilise conjugate gradient optimisation. For this purpose, the partial derivatives of the LOO-CV predictive mean and variances from Eq.(3.45) with respect to the hyperparameters is required:

$$\begin{aligned} \frac{\partial \mu_i}{\partial \theta_j} &= \frac{[Z_j \boldsymbol{\alpha}]_i}{[K^{-1}]_{ii}} - \frac{\boldsymbol{\alpha}_i [Z_j K^{-1}]_{ii}}{[K^{-1}]_{ii}^2}, \\ \frac{\partial \sigma_i^2}{\partial \theta_j} &= \frac{[Z_j K^{-1}]_{ii}}{[K^{-1}]_{ii}^2}, \end{aligned} \tag{3.46}$$

where $\boldsymbol{\alpha} = K^{-1}\mathbf{y}$ and $Z_j = K^{-1}\partial K/\partial \theta_j$. The partial derivatives of Eq.(3.44) are derived using the chain rule, resulting in (Rasmussen & Williams, 2006, p.117)

$$\begin{aligned} \frac{\partial L_{\text{LOO}}}{\partial \theta_j} &= \sum_{i=1}^{n} \frac{\partial}{\partial \mu_i} \left(\log p(y_i \mid X, \mathbf{y}_{-i}, \boldsymbol{\theta})\right) \frac{\partial \mu_i}{\partial \theta_j} + \frac{\partial}{\partial \sigma_i^2} \left(\log p(y_i \mid X, \mathbf{y}_{-i}, \boldsymbol{\theta})\right) \frac{\partial \sigma_i^2}{\theta_j} \\ &= \sum_{i=1}^{n} \frac{\alpha_i [Z_j \boldsymbol{\alpha}]_i - \frac{1}{2}\left(1 + \frac{\alpha_i^2}{[K^{-1}]_{ii}}\right)[Z_j K^{-1}]_{ii}}{[K^{-1}]_{ii}}. \end{aligned} \tag{3.47}$$

The computational complexity for computing the inverse of K is $O(n^3)$, and the complexity for the derivative Eq.(3.47) is also $O(n^3)$ per hyperparameter. Therefore, the computational load of the derivatives is higher for the LOO-CV method compared to the method based on the marginal likelihood, Eq.(3.41), (Rasmussen & Williams, 2006, p.117).

In Eq.(3.43), the log of the validation density has been used as a cross-validation measure of fit. In other words, the negative log validation density refers to a loss function. Other loss functions, for example, the common squared error, could also be considered. Nevertheless, squared error as a loss function only relies on predicted mean values and disregards validation set variances. This limitation results in one degree of freedom remaining undetermined by LOO-CV based on squared error loss, as the predicted mean is unaffected by the scale of the covariances. This issue does not occur with the full predictive distribution, which is influenced by the covariance function's scale (Rasmussen & Williams, 2006, p.118).

Comparing the pseudo-likelihood of the LOO-CV approach with the marginal likelihood discussed earlier, Rasmussen & Williams (2006, p.118) mentioned that it is worth considering the circumstances under which each method might be preferred. Both methods have similar computational requirements. While this aspect has been under-explored empirically, it is inter-

esting to note that the marginal likelihood offers the probability of observations given model assumptions. In contrast, the LOO-CV result provides an estimate for the log predictive probability, even if the model assumptions are not entirely satisfied (Rasmussen & Williams, 2006, p.118).

Chapter 4

Data Analysis Methodology

The objectives involve (i) generating an interpolation curve that includes a confidence interval for the data (interpolating) and (ii) extending the analysis to cover $\mathbf{x}$ values outside the observed range (extrapolating or predicting). The Gaussian process was applied to obtain results and achieve the goals. We also analysed the characteristics and behaviour of the various kernel functions. This methodology is commonly applied to time series data, where $\mathbf{x}$ typically denotes time intervals, for example, days.

First, we utilise the Gaussian process with diverse kernels on linearly generated data that includes some noise following a $N(0, 1)$ distribution. This analysis aims to assess how the different kernels influence the posterior distribution (see Appendix C for the implementation).

For a stock market data, the process involves several steps. Initially, gathering stock market data as pairs, consisting of dates and their corresponding closing prices. Then, proceeding to select a suitable kernel or covariance function. Following this, the process includes learning the hyperparameters associated with the chosen kernel. Subsequently, deriving the interpolation curve along with its associated confidence band. Finally, making predictions of the stock market. These steps are explained in more detail as the following sections (see Appendix D for the implementation).

4.1 Data Collection

The stock market data for Alphabet Inc. (GOOG) was obtained from the Yahoo Finance website (accessed at: Alphabet Inc. (GOOG) Stock Price, or see Appendix D). It contained the data from 18 Jully 2022 to 18 July 2023. The collected data underwent a cleaning process to create pairs (x_i, y_i), where x_i represents the date and y_i represents the closing price on that specific date. The analysis was centered on a relatively short time frame, consisting of 50 data points, representing a small window of time. This dataset is then represented as $\{(x_i, y_i) \mid i = 1, 2, 3, \ldots, n\}$, where n = 50. This limited dataset is chosen for further analysis and modeling. However, it is important to note that various time window durations were also experimented with during the investigation.

4.2 Kernel or Covariance Function Selection

Choosing an appropriate kernel or covariance function that would define the relationships between the data points. This function determines how much influence nearby data points have on each other. There are several common choices for this function, including
squared exponential kernel:

$$K(x, x') = \sigma^2 e^{-\frac{|x-x'|^2}{2l^2}} \tag{4.1}$$

squared exponential kernel with noise:

$$K(x, x') = \sigma^2 e^{-\frac{|x-x'|^2}{2l^2}} + \sigma_n^2 \delta(x, x') \tag{4.2}$$

squared exponential kernel with linear trends and noise:

$$K(x, x') = \sigma^2 e^{-\frac{|x-x'|^2}{2l^2}} + [s_1^2 + s_2^2(x-h)(x'-h)] + \sigma_n^2 \delta(x, x'). \tag{4.3}$$

Hyperparameters, denoted as σ, s_1, l, and others, play a crucial role in defining the behaviour of the model. These parameters must be carefully tuned to achieve the best possible representation of the given dataset. This optimisation process is known as *hyperparameter learning*, which is the next step.

The squared exponential kernel, also known as the Gaussian kernel or Radial Basis Function (RBF) kernel, is widely used in Gaussian Process modeling due to its versatile properties and effective behaviour. It is a stationary kernel that assigns high similarity between data points that are close to each other in the input space and decreases rapidly as the distance increases. This behaviour allows it to capture both smooth and complex relationships in the data. Additionally, the kernel is infinitely differentiable, which makes it mathematically tractable and well-suited for various optimisation and inference techniques. Its smoothness and flexibility make it a robust choice for interpolation, extrapolation, and modeling functions with unknown or complex patterns. However, its choice also depends on the specific characteristics of the data and the problem at hand.

Note that the kernel in Eq.(4.3) is similar to the kernel proposted by Gonzalves et al. (2019, p.26) for their GP-ARX model. The distinction lies in the choice of the kernel function itself: while they employed an exponential kernel, we opted for the squared exponential kernel as part of our kernel function. The squared exponential kernel could exphasise the covariance between the data that have a high covariance since the function contains the term $|\mathbf{x} - \mathbf{x}'|^2$ rather than just $|\mathbf{x} - \mathbf{x}'|$. In addition, we performed unsupervised learning Gaussian process in this research.

In TensorFlow, the squared exponential and linear trends kernel can be defined by the command `tfp.math.psd_kernels.ExponentiatedQuadratic` and `tfp.math.psd_kernels.Linear`, respectively.

Other kernel functions as outlined in Table 3.1 are also used to experiment. One example is the local periodic kernel, obtained by taking the product of the periodic and squared exponential

kernels. Additionally, we explored the impact of the rational quadratic kernel on the posterior distribution.

4.3 Hyperparameters Learning and Optimisation

Hyperparameters of the chosen kernel need to be learned. These hyperparameters affect the shape and characteristics of the covariance function. This step often involves optimisation techniques to find the hyperparameters that best fit the data.
Suppose we define the kernel matrix:

$$K_{ik} = K(\mathbf{x}_i, \mathbf{x}_k). \tag{4.4}$$

Bayes' theorem is utilised to find the optimal hyperparameters that maximise the marginal likelihood. Recall the log marginal likelihood from Eq.(3.40):

$$\log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^{\mathrm{T}} K_{ik}^{-1}\mathbf{y} - \frac{1}{2}\log|K_{ik}| - \frac{n}{2}\log 2\pi. \tag{4.5}$$

The learning process employs the *Adam optimiser* (Kingma & Ba, 2017, pp.2-3) using the command `tf.keras.optimizers. Adam` implemented by TensorFlow (2023) for optimisation. Note that other optimisations, such as *steepest descent* or *gradient descent*, can also be used. When a hyperparameter θ is involved, the process improves the parameter by iteratively adjusting it in the direction that minimises the loss function. The improved parameter is given by (applying Eq.(3.42))

$$\theta_j^t = \theta_j^{t-1} - \eta \frac{\partial}{\partial \theta_j} \log p(\mathbf{y} \mid X, \boldsymbol{\theta}), \tag{4.6}$$

where t is the epoch and η is the learning rate. After performing certain calculations, the gradients (derivatives) term in Eq.(4.6) can be determined as follows (from Eq.(3.41):

$$\frac{\partial}{\partial \theta_j} \log p(\mathbf{y} \mid X, \boldsymbol{\theta}) = \frac{1}{2}\,\mathrm{tr}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^{\mathrm{T}} - K_{ik}^{-1})\frac{\partial K_{ik}}{\partial \theta_j}\right), \tag{4.7}$$

where $\alpha = K_{ik}^{-1}\mathbf{y}$ and tr is the trace operation. The Eq.(4.6) is iterated until the hyperparameters show negligible changes. A common choice for the learning rate is 10^{-6} to ensure stability during the iteration process. Depending on the specific problem, achieving satisfactory hyperparameters might require more than 20,000 iterations. However, for this research, the loss function did not significantly change after conducting 2,000 epochs.

This approach ensures that the hyperparameters are chosen in a way that optimally fits the model to the data. However, there is no guarantee that we will achieve the *global* optimum for the hyperparameters.

4.4 Curve and Confidence Band Formation (Interpolation)

Once the hyperparameters are determined, they are used to create the interpolation curve. This curve represents the estimated underlying function that best fits the data. Additionally, a confidence band is computed around the curve. The band indicates the range of uncertainty and provides a measure of how confident we are in the curve's accuracy.

Recalling the mean for $f(x)$ (the value of the curve, y_{guess}) for a specific value x can be expressed (by Eq.(3.24)) as

$$y_{\text{guess}} = \bar{f}(\mathbf{x}) = \sum_{k=1}^{n} \alpha K(\mathbf{x}, \mathbf{x}_k), \tag{4.8}$$

where $\alpha = K_{ik}^{-1} y_k$, and the variance is given by Eq.(3.23):

$$\sigma^2 = \mathbb{V}[f] = K(\mathbf{x}, \mathbf{x}) - \sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}_k)^{\mathrm{T}} K(\mathbf{x}, \mathbf{x})^{-1} K(\mathbf{x}_k, \mathbf{x}), \tag{4.9}$$

The confidence interval, ϵ, with a confidence level of 95% for the given point is determined by twice the standard deviation (2σ). Therefore, the posterior will be

$$\bar{f}(\mathbf{x}) \pm 2\sigma. \tag{4.10}$$

It is advisable to avoid choosing $\mathbf{x}$ values that coincide with $\mathbf{x}_i$ values, as this could lead to artificially small error bars for the input data.

4.5 Prediction (Extrapolation)

While the interpolation curve fits within the observed data range, it is also valuable to extend the analysis to predict values of $y_{*\text{guess}}$ beyond the observed range. This is known as extrapolation and could be useful for making predictions about future stock prices.

In this research, the derived function y and the confidence band obtained from Eq.(4.8) and Eq.(4.9), respectively, were utilised to make predictions for the next 5 days beyond the range of observed data.

Similar to the interpolation process, the mean prediction for $f(\mathbf{x}_*)$ at a test input $\mathbf{x}_*$ from Eq.(3.24), the value of the curve for a specific value $\mathbf{x}_*$ can be expressed as

$$y_{*\text{guess}} = \bar{f}(\mathbf{x}_*) = \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_i, \mathbf{x}_*), \tag{4.11}$$

where $\alpha = K^{-1}\mathbf{y}$, and the variance (or confidence band) is given by Eq.(3.23):

$$\sigma_*^2 = \mathbb{V}[f_*] = K(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^{\mathrm{T}} K(\mathbf{x}_*, \mathbf{x}_*)^{-1} \mathbf{k}_*, \tag{4.12}$$

where $\mathbf{k}_* = \mathbf{k}(\mathbf{x}_*)$ is the vector of covariances between the test point and the n training points for one test point $\mathbf{x}_*$. The confidence interval, ϵ_*, with a confidence level of 95% for the given point is determined by twice the standard deviation ($2\sigma_*$). Hence, the prediction will be

$$\bar{f}(\mathbf{x}_*) \pm 2\sigma_*. \tag{4.13}$$

These steps guide the process of creating an interpolation curve that captures the underlying pattern and making predictions (extrapolation) of stock market data while accounting for uncertainty through confidence intervals.

4.6 Implementation with TensorFlow

In TensorFlow, both `tfd.GaussianProcess` (TensorFlow, 2023) and `tfd.GaussianProcessRegressionModel` (TensorFlow, 2023) are classes from TensorFlow Probability (TFP) (TensorFlow, 2022) that are used for Gaussian Process modeling, but they serve different purposes in the context of modeling and inference. `tfd.GaussianProcess` represents a Gaussian Process distribution without observed data. It models the prior distribution of the Gaussian Process, useful for analysing its properties independently of specific data. On the other hand, `tfd.GaussianPro cessRegressionModel` is tailored for Gaussian Process regression tasks, handling observed data for prediction at new points. It requires observed data, a kernel, and observation noise variance, enabling posterior inference and sampling.

The Algorithm 2 are distinct steps in Gaussian Process modeling, which is based on an example in TensorFlow (2023). Initially, the kernels are defined and a `tfd.GaussianProcess` object is created for hyperparameters optimisation. The kernels' hyperparameters are then optimised to better fit the observed data (in this research, Adam optimiser was applied). After optimisation, the optimised kernel can directly be used when constructing the `tfd.GaussianProcessRegressionModel`. This model enables predictions and regression tasks using the optimised kernel and observed data. The two steps (kernel optimisation and Gaussian Process regression modeling) are logically connected, and there is no requirement to redefine the kernel after optimisation for the `tfd.GaussianProcessRegressionModel`. The implementation codes (adapted from an example in TensorFlow (2023)) of all step using TensorFlow are shown in Appendix C and Appendix D and the results will be visualised in the next chapter.
Initially, we examined the performance of various kernel functions using linear trend data that was generated with Gaussian noise $N(0, 1)$. The following cases were investigated:

(a) Squared exponential kernel without noise

(b) Squared exponential kernel with noise

(c) Squared exponential + linear kernels with noise

Algorithm 2 Gaussian process regression model in TensorFlow (based on TensorFlow (2023))

1: **procedure** PERFORMING GAUSSIAN PROCESS REGRESSION AND MAKING PREDICTIONS USING TENSORFLOW
2: **input:** $\mathbf{x}$ (observed inputs), $\mathbf{y}$ (observed data), $\mathbf{x}_*$ (test inputs), $\mathbf{y}_*$ (test data)
3: Define kernels with initial hyperparameters.
4: Create a `tfd.GaussianProcess` object with the kernel and observed data.
5: Optimise the hyperparameters of the kernel using gradient descent or another optimisation method (in our research, Adam optimiser was used). This step updates the kernel's hyperparameters to better fit the observed data.
6: Use the optimised kernel to construct a `tfd.GaussianProcessRegressionModel`.
7: Obtain the mean and standard deviation from the Gaussian process regression model
8: Plot the mean with 95% confident band (2σ)
9: **end procedure**

Then the analysis of Alphabet Inc. (GOOG) stock market data was divided into different cases, each focusing on different kernel configurations:

(a) Squared exponential kernel without noise

(b) Squared exponential + noise kernel

(c) Squared exponential + linear + noise kernel

(d) Squared exponential + constant kernels without noise

(e) Squared exponential + constant + noise kernel

(f) Squared exponential + linear + constant + noise kernel

(g) Squared exponential + linear + local periodic + noise kernel

(h) Squared exponential + local periodic + rational quadratic + noise kernel

(i) Squared exponential + linear + + local periodic + rational quadratic + noise kernel

The posterior predictive distribution in a conjugate GP regression model can be implemented by using command `tfp.distributions.GaussianProcessRegressionModel`, which is a class in TensorFlow that offers a convenient way to perform Gaussian process regression. This class represents the distribution over function values at certain points in an index set, conditioned on noisy observations at another set of points. Specifically, it assumes a Gaussian process prior, where $f \sim GP(m, k)$ with independent and identically distributed (iid) normal noise on observed function values. The posterior inference in this model can be carried out analytically. This distribution is parameterised by the mean vector m and the covariance matrix k (TensorFlow, 2023).

However, we will see in the next chapter that some resulting posterior means converged towards zero. To avoid this issue, it is common to subtract the mean of the observations from

the data before applying Gaussian processes to avoid zero-mean extrapolation results for distant future predictions. This approach leads to *standardisation* technique, and hence we standardised our data before training the models. Standardisation is a rule of thumb in machine learning. When dealing with numeric features, it is often beneficial to *scale* them into the range of [-1, +1] to ensure fair treatment within a learning algorithm. Failing to perform this transformation can lead to difficulties in the algorithm's effort to find suitable parameters, especially when some features have significantly larger values than others. The standardisation, expressed as $z(y) = (y - \bar{y})/\sigma$, where $\bar{y}$ represents the average of the set Y and σ is the standard deviation of the set Y, results in feature values with a mean of zero and a standard deviation of one. This transformation, commonly known as *standardising* a variable (or *z-scores* in social sciences), brings features to a comparable scale, thereby aiding the learning algorithm in its task (Neal, 1996, p.26). Typically, this process helps models perform better. Standardisation can indeed prevent posterior means, in the context of Gaussian processes or other regression models, from converging towards zero, resulting in more accurate and meaningful predictions. Nevertheless, it is important to note that Gaussian processes are predominantly employed for near-future predictions in real-world scenarios, rather than for predictions extending far into the future. Note also that when dealing with regression problems, a rule of thumb is to standardise or normalise the data before training models. Therefore, we also standardised the observations before applying the Gaussian processes with some different kernels.

In summary, `tfd.GaussianProcess` is more versatile, targeting the representation of the Gaussian Process distribution itself, while `tfd.GaussianProcessRegressionModel` is designed for regression with observed data. TensorFlow's class `tfp.distributions.GaussianProcessRegressionModel` simplifies the implementation of Gaussian process regression and provides the posterior predictive distribution for the model. The choice between them depends on the modeling objectives.

Chapter 5

Results and Discussion

The provided Python codes that implement the outlined procedure can be found in the Appendix C and Appendix D. The code in Appendix C was used to analyse the behaviours of the squared exponential, linear, and noise kernel functions while the code in Appendix D has been tested to predict using 50 days of the Alphabet Inc. (GOOG) stock market data stored in "`GOOG.csv`" (see Yahoo Finance (2023) or Appendix D), with an additional 5 days of data representing the "future" data. This "future" data represents a set of data points that the model has not encountered during its training phase. The results are illustrated in the following sections. Note that we conducted experiments across various time frames. However, for brevity, we present the results (and the corresponding codes) for only one time window here (and in Appendices C and D). In addition, the results for the kernels (d)-(h) are shown in Appendix E.

5.1 Results

5.1.1 Gaussian Process with a Linear Trend Data

We first analysed the behaviours of each kernel functions. For simplicity, we generated a dataset from a linear function with a Gaussian noise following a $N(0, 1)$ distribution as shown in Figure 5.1.

Squared Exponential Kernel without Noise

A Gaussian process regression model with squared exponential (RBF) kernel without noise was implemented, resulting in Figure 5.2.

Figure 5.2(a) illustrates a substantial decrease in the negative log marginal likelihood during the beginning of the training stage, then stabilising from around epoch 100 at approximately 110. The hyperparameters optimised through this process were then used to generate the predictive mean and (95%) confidence band (uncertainty), demonstrated in Figure 5.2(b). It can be seen that the confidence band exhibited a narrow span within the interpolation region, while it widened considerably in the extrapolation region. This is because of the close proximity of

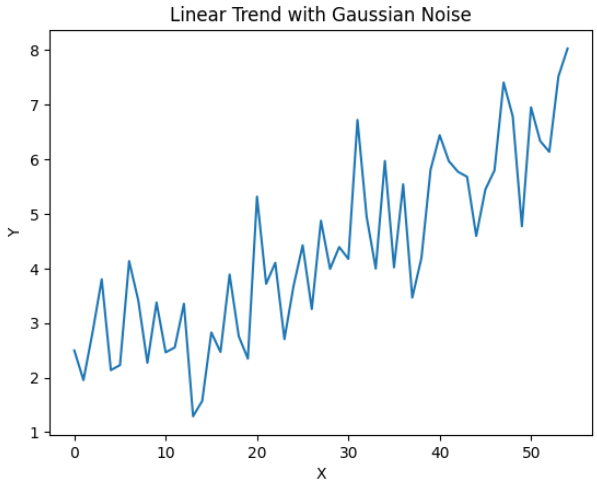


Figure 5.1: A linear trend data generated with Gaussian noise following a $N(0, 1)$ distribution.

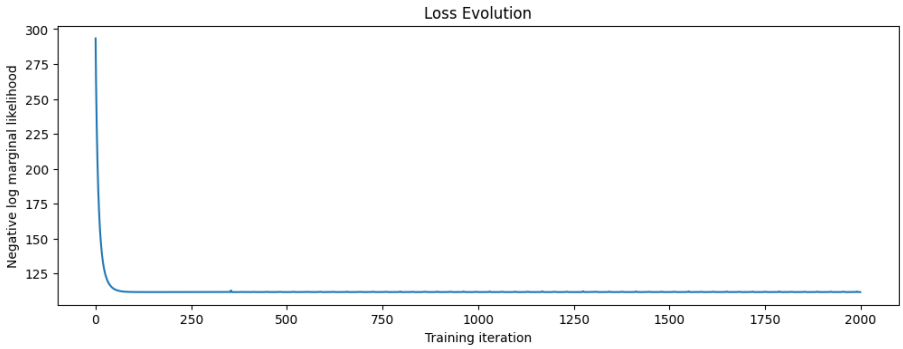
observation data points, indicating that the target values should closely resemble those neighboring points, aligning with our expectations. In addition, the high uncertainty in the extrapolation region implies a wide range of potential outcomes, since we make predictions based on the observations without additional information.

Note that the convergence of the posterior mean in the Gaussian process regression model, employing a squared exponential kernel without noise, tended to approach zero on the test inputs. In spite of this limitation, this kernel remained well-suited for interpolation, particularly when dealing with scenarios requiring the filling or imputation of missing data gaps.

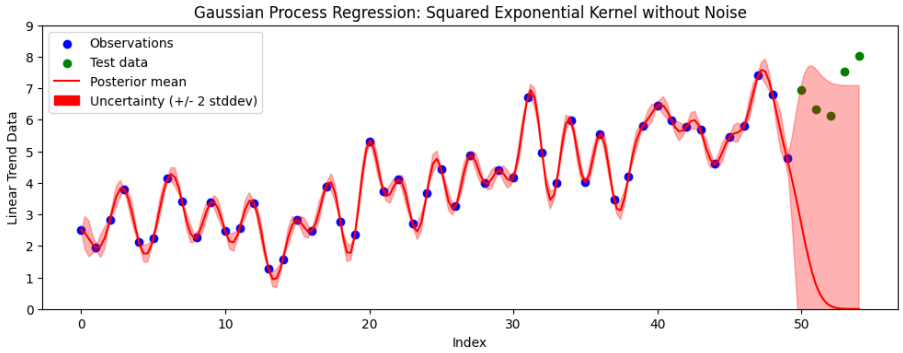
Squared Exponential Kernel with Noise

Figure 5.3 shows the posterior distribution with 95% confidence band ($\pm 2\sigma$). Figure 5.3(a) depicts a noticeable reduction in the negative log marginal likelihood during the initial training phase, with stabilisation occurring around epoch 250 at an approximate value of 70. Notably, halting the iterations at 200 would preclude the attainment of better hyperparameters. These optimised hyperparameters were subsequently employed to generate the predictive mean and 95% confidence band (uncertainty) as displayed in Figure 5.3(b). The graph shows that the predictive mean function was roughly linear. Moreover, the confidence band was consistent between the interpolation and extrapolation domains. The interpolation domain showcased higher uncertainty than the noise-free kernel model due to the presence of noisy observations. Consequently, the predictive mean might not precisely intersect the data points, yet maintains smooth.

Note that the posterior mean's convergence in the Gaussian process regression model, utilising a squared exponential kernel with noise, similarly approached zero on the test inputs. However, this convergence occurred at a slower rate compared to the noise-free model, as shown in Figure 5.3(c). Hence, despite the smoothness of the posterior mean, this model is not suitable for making predictions in the extrapolation region.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 120.



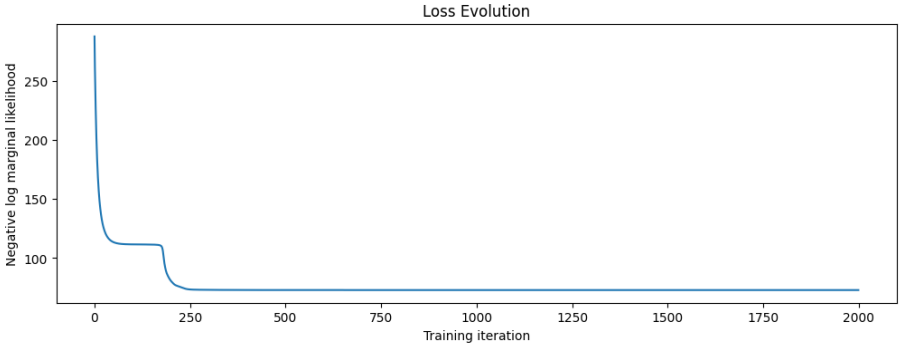
(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

Figure 5.2: The result after 2,000 epochs of training the Gaussian process with squared exponential kernel without noise. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model.

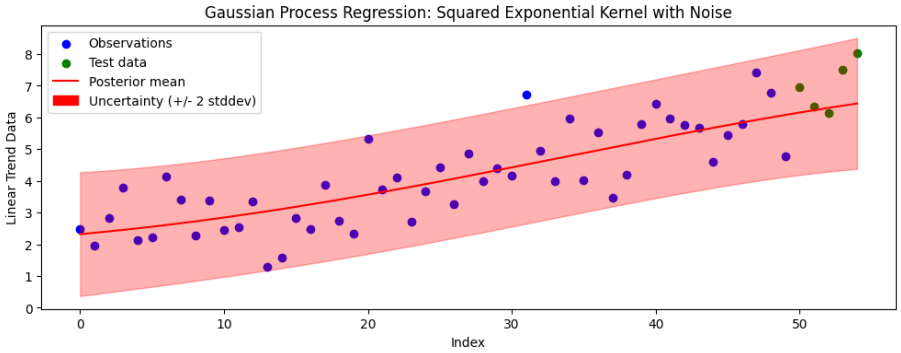
Squared Exponential Combined with Linear Kernel and Noise

Figure 5.4 shows the posterior distribution with 95% confidence band ($\pm 2\sigma$). In Figure 5.4(a), there was a noticeable decrease in the negative log marginal likelihood during the initial training phase, followed by stabilisation from around epoch 100 at a value of roughly 70. These optimised hyperparameters were then used to generate posterior mean and the 95% confidence band (uncertainty) of the Gaussian process regression model, as illustrated in Figure 5.4(b). From the graph, it is apparent that the predictive mean was not strictly linear, but it closely followed the observed data points. However, the overall trend of the posterior mean was approximately linear. Furthermore, the uncertainty was lower compared to the uncertainty of the previous model without the linear kernel. Nonetheless, in the extrapolation regions, the confidence band gradually increases due to the lack of additional information beyond the observed data.

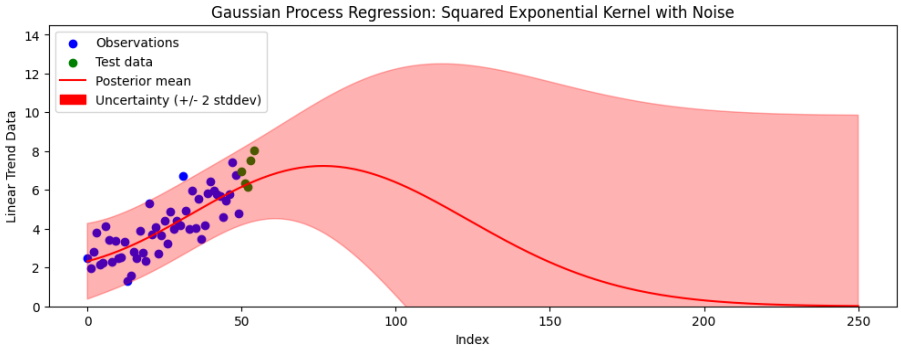
It is also important to observe that the posterior mean of the Gaussian process regression model, employing a squared exponential and linear kernel with noise, did not tend to approach



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 250.

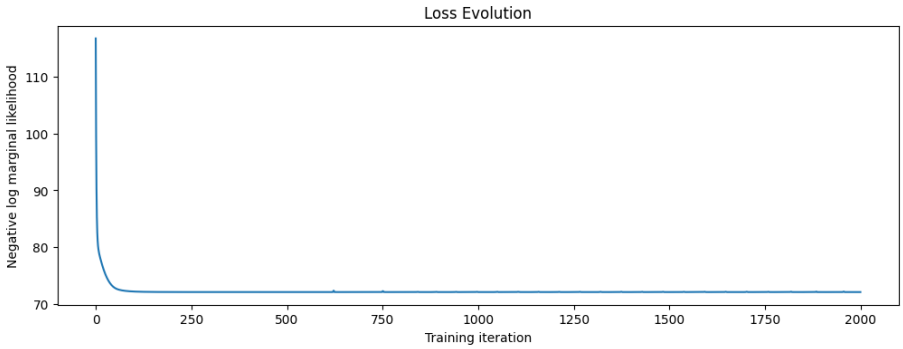


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

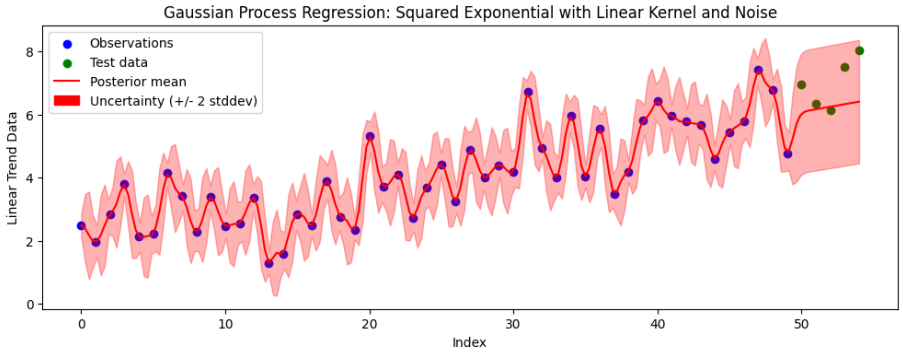


The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

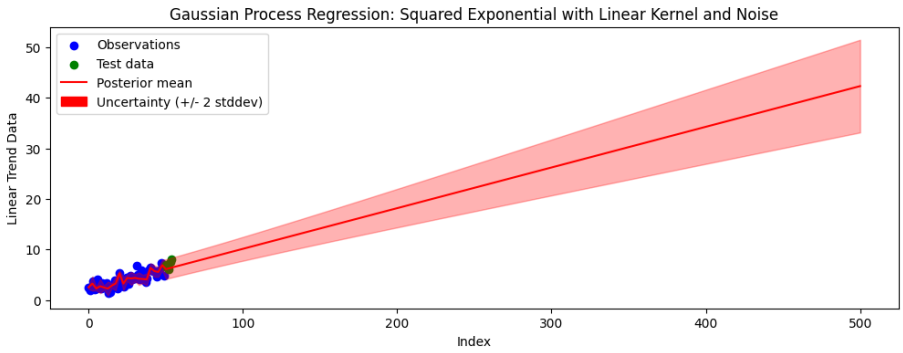
Figure 5.3: The result after 2,000 epochs of training the Gaussian process with squared exponential + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 120.



(b) The posterior by the Gaussian process regression model with squared exponential kernel without noise, including 95% confidence band (2σ).



(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

Figure 5.4: The result after 2,000 epochs of training the Gaussian process with squared exponential + linear + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.

zero on the test inputs like the other models without the linear kernel. Instead, it demonstrated a consistent increase in both posterior mean and variance as the index values increased, as shown in Figure 5.4(c). Consequently, this implies that this model is more suitable for making predictions.

Overall, this analysis reveals that the Gaussian process regression model using a squared exponential and linear kernel with noise performs better in making predictions. On the other hand, the model employing only the squared exponential kernel (without linear components or noise) appears adept at imputing or filling in missing data gaps.

5.1.2 Gaussian Process with the Alphabet Inc. (GOOG) stock market data

In the previous section, we investigated the impact of different kernels on the posterior distribution. In this section, we will analyse the outcomes of employing various kernel functions in Gaussian process regression on the Alphabet Inc. (GOOG) stock market data. An illustrative representation of the dataset used in this study is shown in Figure 5.5. The dataset comprised 55 data points, with 50 being observations and the remaining 5 serving as test data points. For the scope of this research, our analysis focused only on the 'Date' and 'Close' features.

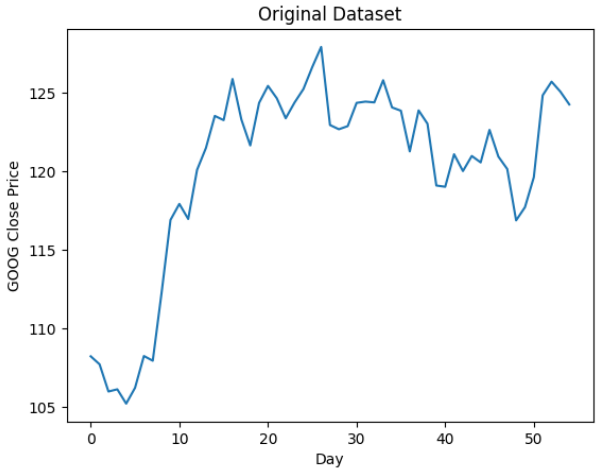
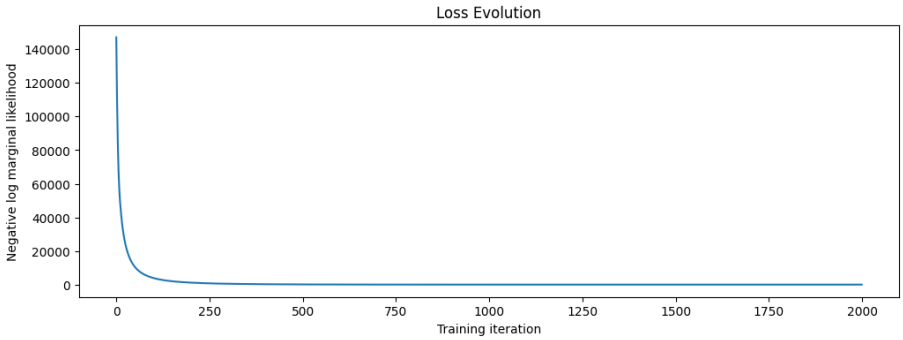


Figure 5.5: An example of the GOOG stock market dataset to be analysed in this research.

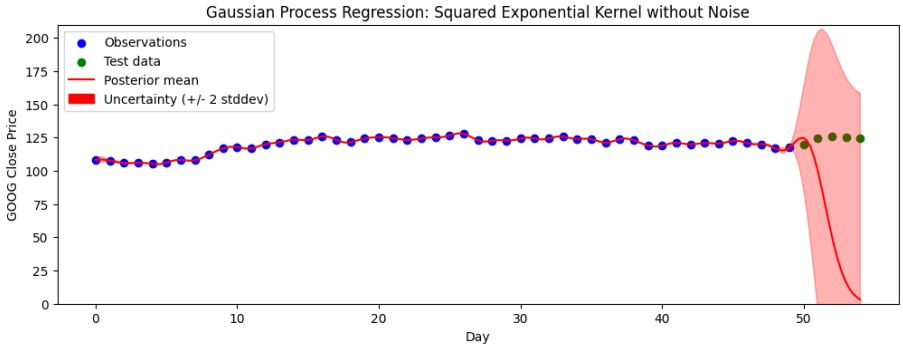
Squared Exponential Kernel without Noise

Figure 5.6 shows the result of applying the Gaussian process using the squared exponential kernel without noise.

The behaviour of the Gaussian process regression model's posterior mean, using a squared exponential kernel without noise, was characterised by its convergence towards zero on the test inputs (Figure 5.6(b)), as we expected. Despite this behaviour, the kernel remains well-suited for interpolation purposes, especially when addressing scenarios involving missing or incomplete data that require imputation or filling. This concept was discussed in the previous section.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 250.



(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

Figure 5.6: The result after 2,000 epochs of training the model with squared exponential kernel without noise. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model using squared exponential kernel without noise.

Squared Exponential Kernel with Noise

The result of applying the Gaussian process using the squared exponential kernel with noise is illustrated in Figure 5.7.

We can see from the Figure 5.7(b) that the posterior mean aligned fairly well with the observed dataset. Nearly all of the observed data points fell within the 95% confidence band. However, the model's predictive capacity appears to be less accurate, as a substantial portion of the actual data points (displayed in green) lied outside the 95% confidence band.

For a more comprehensive insight, let's expand our analysis to observe the behaviour of the model across a wider range of test inputs and examine the corresponding converging values. The graph in Figure 5.7(c) provides a clear illustration of the posterior mean's behaviour within the Gaussian process regression model that utilised a squared exponential kernel with noise. The trajectory showcased a clear trend of convergence towards zero on the test inputs.

To prevent the extrapolation from having a zero mean in predictions far into the future, a rule

of thumb is to initially standardise the data before applying the Gaussian process. Figure 5.8 shows the result of applying this approach to the Gaussian process with the squared exponential with noise kernel. We can see that the convergence of the posterior mean became the mean of the observations as depicted in Figure 5.8(b). Furthermore, the confidence band was narrower than that of the training without standardisation.

Squared Exponential with Linear Kernel and Noise

Figure 5.9 illustrates the result of applying the Gaussian process using the squared exponential with linear and noise kernel.

The result revealed in Figure 5.9(b) demonstrates a posterior mean that is relatively smooth and closely follows the observed dataset, similar to the model that lacks a linear kernel. The vast majority of observed data points were enveloped by the 95% confidence band. Although, the model's extrapolation was less accurate than that of the interpolation, the actual test data points were closer to the 95% confidence band compared to the previous model.

We extended the analysis to gain a more detailed understanding of the behaviour of the posterior mean and its associated converging values across additional test inputs as shown in Figure 5.9(c). The figure suggests that the posterior mean of the Gaussian process regression model using a squared exponential with a linear and noise kernel converged to the mean of the observations on the test inputs. Furthermore, although the extrapolation's posterior variance was higher than the interpolation's posterior variance, it was significantly less than the extrapolation's posterior variance of the Gaussian process regression model using a squared exponential + noise kernel (without a linear kernel).

For regression tasks, a common practice is to standardise or normalise the data prior to model training. Accordingly, we standardised the observations before implementing the Gaussian process, and the outcomes are illustrated in Appendix E. It is evident that the posterior mean aligned closely with the mean of the observations.

We have also used other kernel functions, such as a local periodic kernel, which is the multiplication of the squared exponential kernel and the periodic kernel, and the quadratic rational kernel. The results are shown in Appendix E.

5.2 Discussion

The unsupervised Gaussian process applied to our collected data lacked information about the stock market and time-series data. The primary constraints imposed were ensuring that the drawn curve closely aligned with the observed data points and/or maintained smoothness (by the squared exponential kernel function). This explains the rapid increase in error bars for predicting values in the extrapolation region. This observation is in line with the explanation presented by D'Elia et al. (2023, pp.325-326).

After conducting extensive analysis using various time windows on the dataset, the results have yielded several noteworthy findings, which are presented in the following subsections.

5.2.1 Filling Missing Data using Squared Exponential without Noise

By selecting the squared exponential kernel (without noise) (Eq.(4.1)) for the Gaussian process, the model would generate a curve that closely follows the data, regardless of the noise. This characteristic makes it suitable for imputing or filling in ”missing data” tasks in various applications.

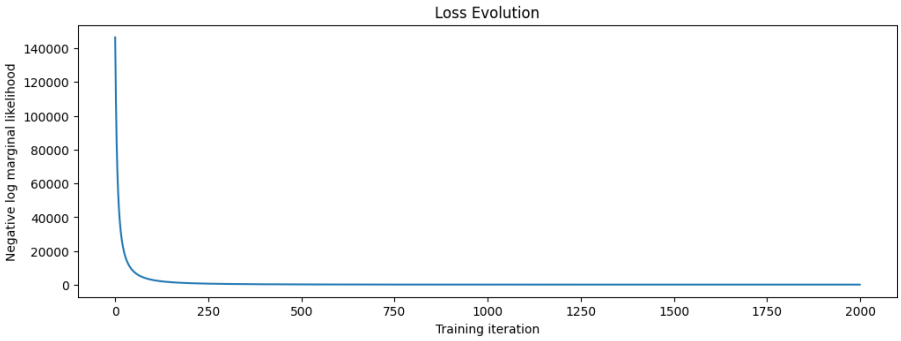
5.2.2 Predicting using Squared Exponential with Noise

In various real-world modeling situations, it is typical to not have direct access to function values, but instead to work with their noisy versions. Therefore, it is more appropriate to use the kernel described in Eq.(4.2) rather than the one in Eq.(4.1) for the real-world data.

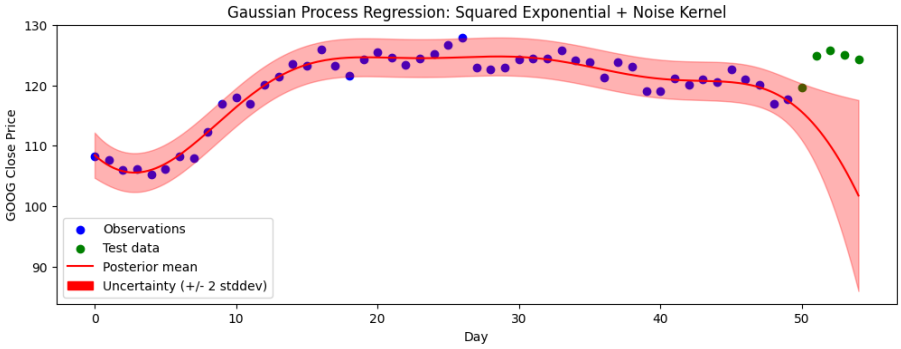
5.2.3 Predicting using Squared Exponential with Linear and Noise Kernel

Opting for the squared exponential kernel, which accounts for both linear trends and noise in the data (Eq.(4.3)), the Gaussian process would extract a ”trend line” from the data. In addition, the confidence interval indicates the level of noise in the day-to-day trading data. Including the linear trend from the kernel will result in the model's predictions converging towards the mean value. This implies that the model's predictions align more sensibly with the underlying data patterns and tend to converge towards meaningful outcomes by incorporating the linear trend from the kernel. In contrast, the predictions of the model tend to converge towards zero without the linear kernel.

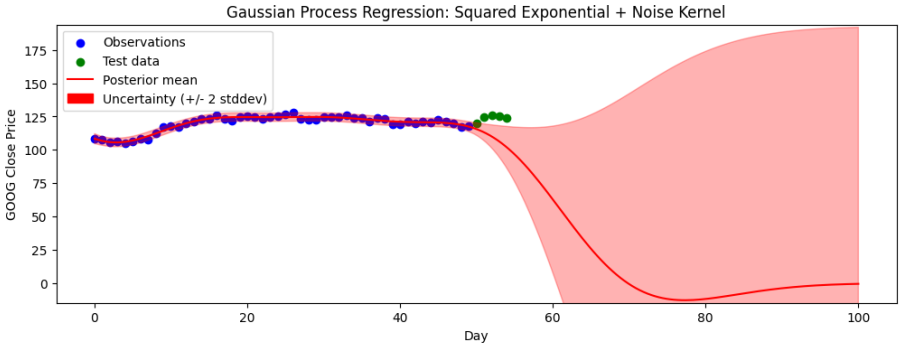
To avoid the zero mean of the extrapolation for distant future predictions, it is a common practice to first standardise the data and then inverse-standardise it back in after applying the Gaussian process. Typically, standardisation makes models perform better. However, in practical applications, the Gaussian process approach is often employed for near-future predictions rather than those far into the future.



(a) The loss function (negative log marginal likelihood) updated in each epoch.

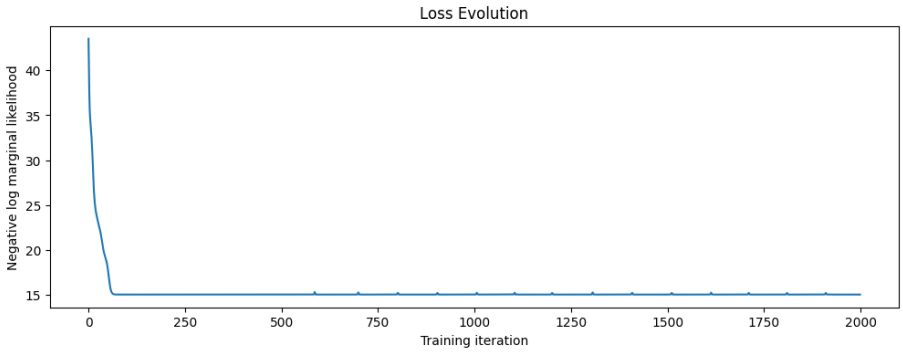


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

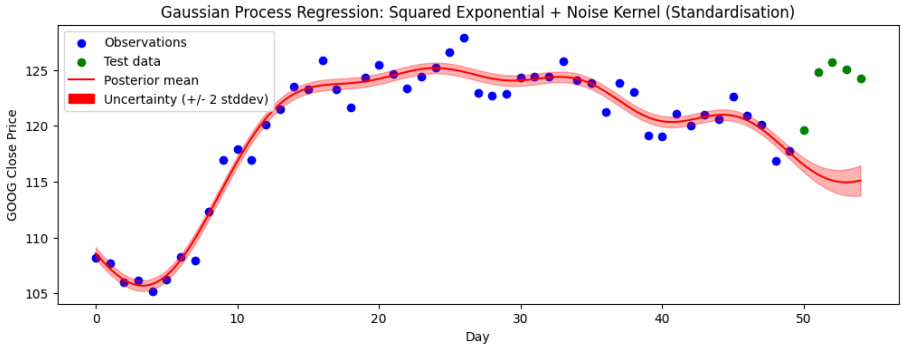


(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

Figure 5.7: The result after 2,000 epochs of training the model with squared exponential + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior distribution of the model on further test inputs.

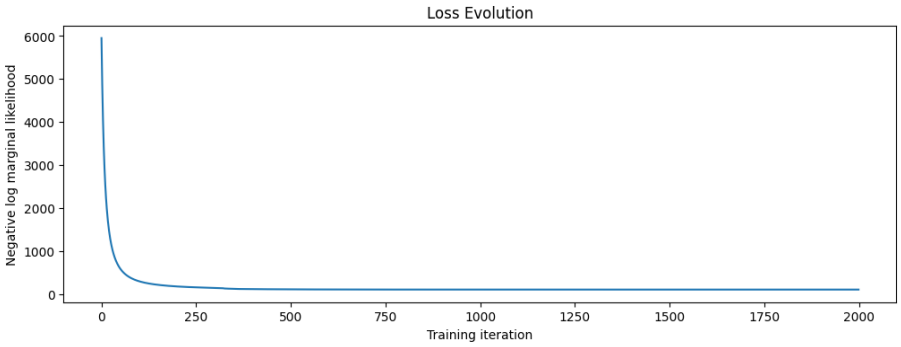


(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 100.

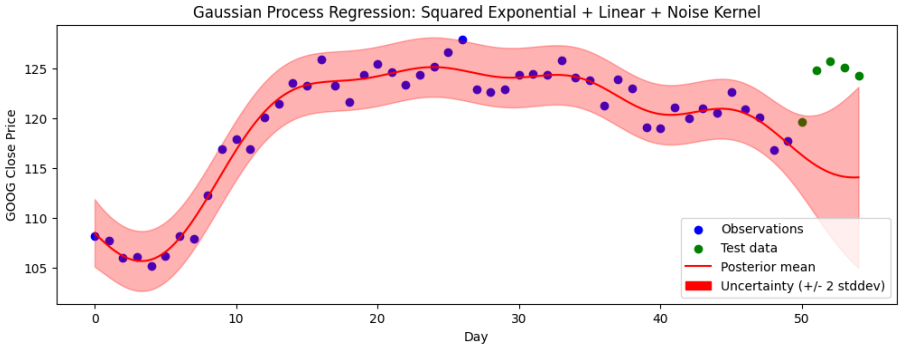


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

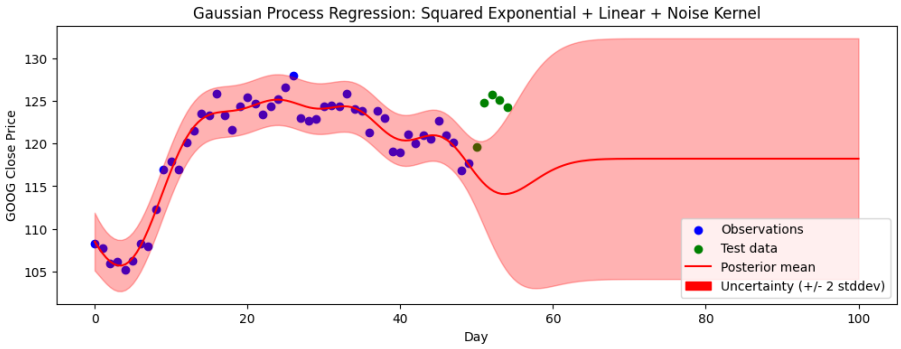
Figure 5.8: The result after 2,000 epochs of training the with squared exponential + noise kernel when the data was standardised. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 400.



(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).



(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

Figure 5.9: The result after 2,000 epochs of training the model with squared exponential + linear + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior distribution of the model on additional test inputs.

Chapter 6

Conclusion and Further Research

6.1 Conclusion

In conclusion, in contemporary applications, noisy time-series data is prevalent, often devoid of a definitive theoretical foundation. The core aim involves encapsulating the data within a time-dependent function for interpolation and predictive purposes. We applied the Gaussian process in this research to leverage diverse covariance functions for fitting observed data and generating predictions with confidence interval bands. Moreover, the study sought to uncover the impact of specific covariance functions on outcomes.

Focusing on Alphabet Inc. (GOOG) stock market data, this research employed Gaussian processes to describe the data and forecast its trajectory, assessing the reliability of predictions through confidence intervals across different time frames. The results implies that the unsupervised Gaussian process application to our collected data revealed its limitations due to the absence of stock market or time-series data knowledge. Despite its constraint to align with observed data points through the squared exponential kernel's smoothness, it led to rapid error bar expansion in the extrapolation region. Extensive analysis across different time windows brought forth several significant insights. The choice of squared exponential kernel without noise resulted in a curve closely describing the data, suitable for addressing "missing data" challenges. Meanwhile, for real-world noisy data, the squared exponential kernel with noise was considered more appropriate. However, incorporating the squared exponential kernel with both linear and noise kernels, the Gaussian process could extract a trend line and its associated confidence interval effectively. The presence of a linear trend in the kernel led to predictions toward mean convergence, whereas predictions tended towards zero in its absence. This means that combining the linear kernel helps make predictions more sensible. In practice, to prevent extrapolation results from having a zero mean for distant future predictions, a rule of thumb is standardisation which generally helps models perform better. However, Gaussian processes are typically used for near-future predictions in real-world applications rather than for predictions far into the future. These findings emphasise the utility of Gaussian processes in interpreting and predicting complex time-series data across diverse domains.

6.2 Further Research

Further studies in this domain hold significant potential for advancing our understanding and application of Gaussian processes in various contexts. One avenue of exploration involves the integration of domain-specific knowledge to enhance the performance of Gaussian process models. Incorporating relevant features, expert insights, or contextual information can improve the accuracy and reliability of predictions. D'Elia et al. (2023, p.326) also emphasised that unsupervised learning could reach great results for interpolating data between measurements. Nonetheless, when the goal is to enhance predictive capabilities in data modelling, incorporating domain knowledge becomes essential. This transition leads us to supervised or guided learning topic, where the inclusion of prior information from underlying knowledge contributes to more powerful predictions.

Furthermore, we might incorporate traditional machine learning or deep learning techniques alongside Gaussian processes to enhance predictive models. The machine learning or deep learning approaches could be employed to derive the underlying functions for the observed data, which may subsequently serve as priors or be utilized in supervised learning, particularly when specific knowledge is lacking.

Moreover, exploring more advanced kernel functions beyond the ones considered in this study could also yield valuable insights. Investigating non-stationary kernels or hybrid combinations of kernels might provide better models for certain types of data. Additionally, understanding the impact of hyperparameter priors and their implications on model performance would refine the optimisation process.

Finally, further study could also involve the application of Markov Chain Monte Carlo (MCMC) methods to enhance Gaussian process modeling. This could lead to improved parameter estimates and more accurate predictive intervals. In addition, for future research, an intriguing direction is to investigate of Gaussian Markov Processes (GMPs) in the context of time series data (Rasmussen & Williams, 2006, Appendix B). Especially when dealing with one-dimensional index sets like the temporal domain, GMPs could offer valuable insights into the intricate interplay between temporal dependencies and spatial correlations. This exploration could lead to innovative approaches for modeling and forecasting complex time-evolving systems. Delving into GMPs within the time series framework has the potential to advance our understanding of their dynamics and properties, thereby enhancing our ability to model and analyse intricate time-dependent data structures.

Appendix A

Mathematical Background

A.1 Matrix Identites

Consider an invertible matrix A of size $n \times n$ and its inverse A^{-1}. These matrices can be partitioned into the following form (Rasmussen & Williams, 2006, p.201):

$$A = \begin{bmatrix} P & Q \\ R & S \end{bmatrix}, \qquad A^{-1} = \begin{bmatrix} \widetilde{P} & \widetilde{Q} \\ \widetilde{R} & \widetilde{S}, \end{bmatrix} \tag{A.1}$$

where P and $\widetilde{P}$ are $n_1 \times n_1$ matrices and S and $\widetilde{S}$ are $n_2 \times n_2$ matrices, where $n = n_1 + n_2$. The components of A^{-1} can be calculated by (Rasmussen & Williams, 2006, p.201)

$$\begin{aligned} \widetilde{P} &= P^{-1} + P^{-1}QMRP^{-1} \\ \widetilde{Q} &= -P^{-1}QM \\ \widetilde{R} &= -MRP^{-1} \\ \widetilde{S} &= M, \end{aligned} \tag{A.2}$$

where $M = (S - RP^{-1}Q)^{-1}$.

A.1.1 Matrix Derivatives

The derivatives of the components of an inverse matrix is given by (Rasmussen & Williams, 2006, p.202)

$$\frac{\partial}{\partial\theta}K^{-1} = -K^{-1}\frac{\partial K}{\partial\theta}K^{-1}, \tag{A.3}$$

where $\partial K/\partial\theta$ represents a matrix consisting of element-wise derivatives of the matrix K with respect to the parameter vector θ. This matrix captures how each element of the covariance matrix K changes in response to variations in the elements of the parameter vector θ (Rasmussen & Williams, 2006, p.202).

Appendix B

Proofs of Creating New Kernels from Old

B.1 Combining Kernels by Summation

In scenarios where two independent random processes, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, are combined into a single process $f(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$, the resulting covariance function $k(\mathbf{x}, \mathbf{x}')$ can be expressed as the sum of the covariance functions of the individual processes, $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')+ k_2(\mathbf{x}, \mathbf{x}')$. This approach is useful, for instance, in combining kernels with distinct characteristic length-scales, offering a way to incorporate different scales of variation within Gaussian process modeling (Rasmussen & Williams, 2006, p.95).

B.2 Combining Kernels by Multiplication

When two independent random processes, $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$, are multiplied to form the process $f(\mathbf{x}) = f_1(\mathbf{x})f_2(\mathbf{x})$, the resulting covariance function $k(\mathbf{x}, \mathbf{x}')$ can be expressed as the product of the covariance functions of the individual processes, $k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$. Furthermore, this concept can be extended, so that $k^p(\mathbf{x}, \mathbf{x}')$ is a valid covariance function for any positive integer p. This approach provides a versatile way to construct new covariance functions by combining existing ones through multiplication (Rasmussen & Williams, 2006, p.95).

Appendix C

Code for Linear Trend Data

The Python code for implementing Gaussian processes with a linear trend data can be found at: https://github.com/Jivacharoen/MSc-Dissertation/blob/main/Linear_Trend%3B_Gaussian_Process_Regression_using_TensorFlow.ipynb.
Some parts of this code was modified from an example in TensorFlow (2023) and Roelants (2019).

Appendix D

Code for the Alphabet Inc. (GOOG) Stock Market data

The Python code for implementing Gaussian processes with the Alphabet Inc. (GOOG) stock market data (for one time frame) can be found at: https://github.com/Jivacharoen/MSc-Dissertation/blob/main/Stock_Market_Data%3B_Gaussian_Process_Regression_using_TensorFlow.ipynb. Some parts of this code has been adapted from an example provided in TensorFlow (2023) and Roelants (2019).

The dataset (`GOOG.csv`) used for this implementation can be found here: https://github.com/Jivacharoen/MSc-Dissertation/blob/main/GOOG.csv, or access at: https://finance.yahoo.com/quote/GOOG/history?period1=1658102400&period2=1689724800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true

Appendix E

Results of Using Other Kernels

E.1 Standardisation: Squared Exponential Kernel without Noise

Figure E.1 shows the result of using our standardised data to train the Gaussian process with squared exponential kernel without noise.

E.2 Standardisation: Squared Exponential + Linear + Noise Kernel

Figure E.2 illustrates the result of using our standardised data to train the Gaussian process with squared exponential + linear + noise kernel.

E.3 Squared Exponential + Constant Kernel without Noise

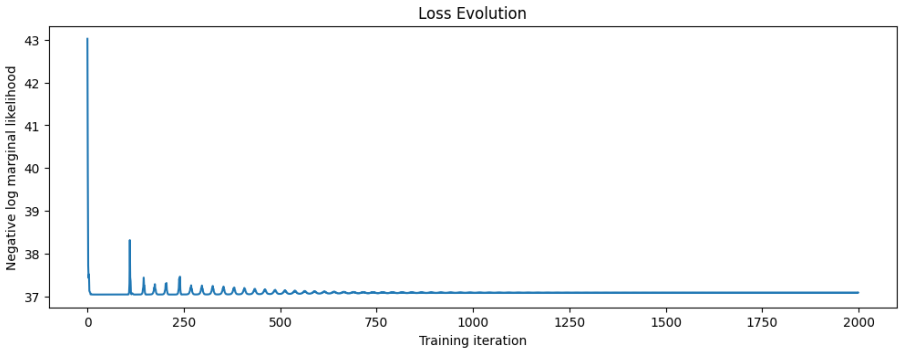
Now, if a constant kernel is added to the current kernel, the resulting posterior distribution with 95% confidence band ($\pm 2\sigma$) will be shown in Figure E.3.

Figure E.3 implies that the addition of the constant kernel caused the posterior mean to converge not to zero, but rather to the mean of the observations.

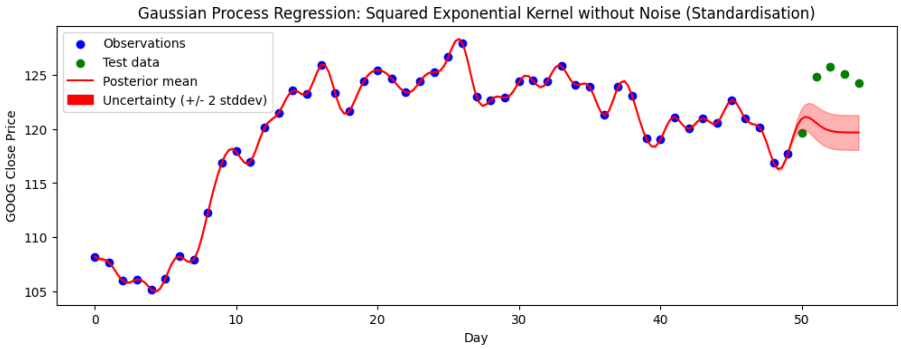
E.4 Squared Exponential + Constant + Noise Kernel

If the constant kernel is added to the current kernel, the resulting posterior distribution with 95% confidence band ($\pm 2\sigma$) will be depicted in Figure E.4.

Let's broaden our analysis by extending range of test inputs and investigating the associated converging values, as displayed in Figure E.4(c). The graphical representation vividly illustrates the posterior mean's behaviour within the Gaussian process regression model employing a squared exponential with a constant and noise kernel. The trajectory demonstrated a consistent trend of convergence towards 0 on the test inputs.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 700.



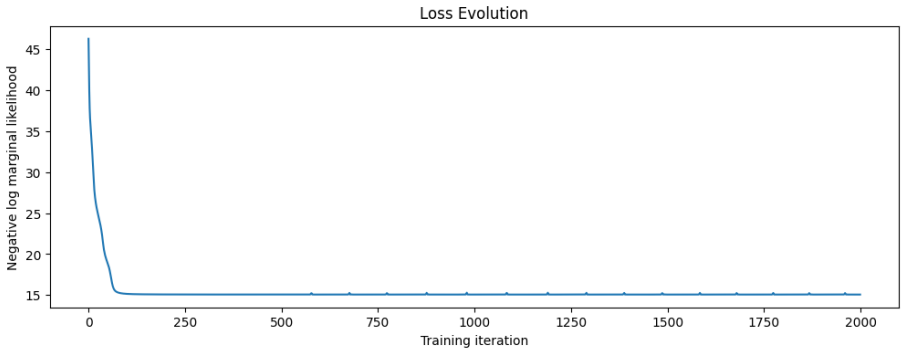
(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

Figure E.1: The result after 2,000 epochs of training the Gaussian process with squared exponential kernel without noise when the data was standardised. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model.

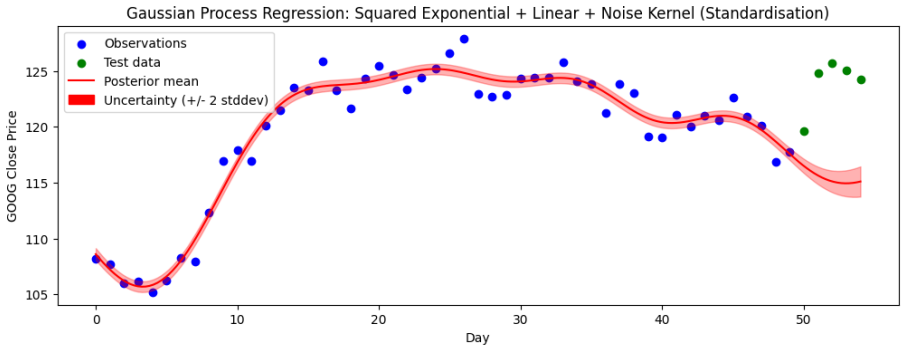
E.5 Squared Exponential + Linear + Constant + Noise Kernel

The constant kernel was added to the current kernel, resulting in posterior distribution with 95% confidence band ($\pm 2\sigma$) as illustrated in Figure E.5. Similar to the model without the constant kernel, the posterior mean continued to converge towards the mean of the observations as shown in Figure E.5(c). Consequently, the inclusion of the constant kernel did not yield substantial improvement to the posterior mean.

Now, let's try other kernels. We have also combined the local periodic and rational quadratic kernels. The results are presented in the following sections.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 125.



(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

Figure E.2: The result after 2,000 epochs of training the Gaussian process with squared exponential + linear + noise kernel when the data was standardised. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model.

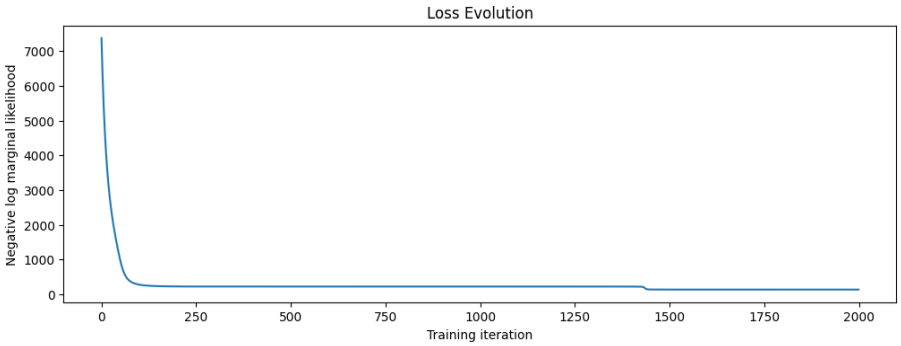
E.6 Squared Exponential + Linear + Local Periodic + Noise Kernel

The local periodic is given by (Roelants, 2019)

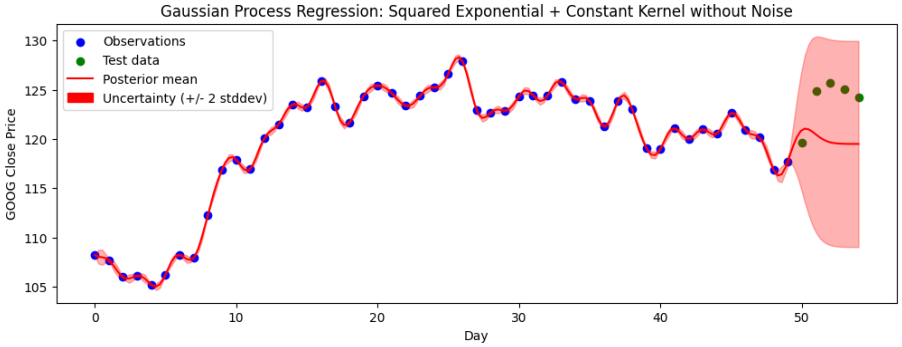
$$k_{\text{Local Periodic}}(\mathbf{x}, \mathbf{x}') = k_{\text{Periodic}}\, k_{\text{RBF}} = \sigma^2 e^{-\frac{2}{l_{\text{p}}^2}\sin^2\left(\frac{\pi|\mathbf{x}-\mathbf{x}'}{p}\right)} e^{-\frac{|\mathbf{x}-\mathbf{x}'|^2}{2l_{\text{rbf}}^2}}, \tag{E.1}$$

where l_{p} is the length-scale of the periodic function and l_{rbf} is the length-scale of the squared exponential function. The local periodic kernel function exhibits periodicity while allowing gradual variations over time. Typical periodic functions do not exhibit exact repetition. To introduce flexibility into our model, the incorporation of a local kernel, such as the squared-exponential, in combination with the periodic kernel can be considered. This augmentation enables the modeling of functions that display local periodicity, wherein the repeating segment's shape may evolve over time (Duvenaud, 2014).

The result of using kernel $k = k_{\text{RBF}} + k_{\text{Linear}} + k_{\text{Local Periodic}} + k_{\text{Noise}}$ is depicted in Figure E.6.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 125.



(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

Figure E.3: The result after 2,000 epochs of training the Gaussian process with squared exponential + constant kernel without noise. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model.

We can see that the posterior mean and variance were not as smooth as the posterior without the local periodic kernel. However, it converged to the mean of the observations.

E.7 Squared Exponential + Local Periodic + Rational Quadratic + Noise Kernel

The rational quadratic kernel (Eq.(3.33)) is typically used to model the *(small) medium term irregularities*. The squared exponential form could also be used for this component, but it turns out that the rational quadratic works better (gives higher marginal likelihood), probably because it can accommodate several length-scales (Rasmussen & Williams, 2006, p.120). The kernel was added to the previous kernel (without the linear kernel), and hence becomes

$$k = k_{\text{RBF}} + k_{\text{Local Periodic}} + k_{\text{RQ}} + k_{\text{Noise}}. \tag{E.2}$$

This combined kernel was used in the Mauna Loa Atmospheric Carbon Dioxide example in the book of Rasmussen & Williams (2006, pp.118-122) and Roelants (2019). The result associated to this kernel is illustrated in Figure E.7.

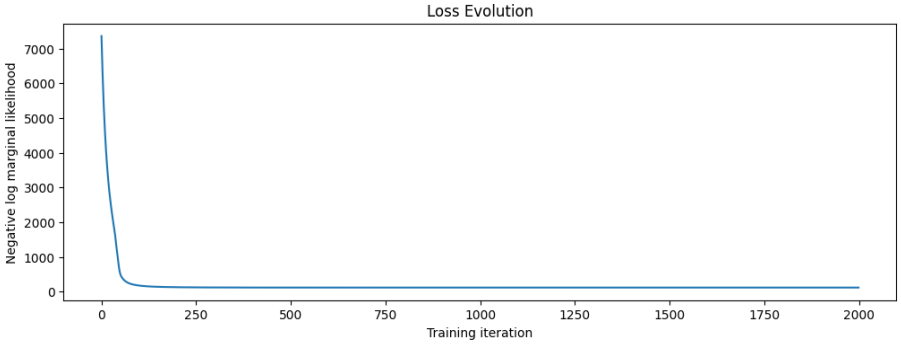
It is worth to note that the posterior mean of the Gaussian process, which incorporated the squared exponential kernel along with local periodic, short-medium term irregularities, and noise kernels, tended to converge towards the mean of the observations although when the linear kernel was excluded, as shown in Figure E.7(c). However, the interpolation posterior mean displayed noticeable fluctuations.

E.8 Squared Exponential + Linear + Local Periodic + Rational Quadratic + Noise Kernel

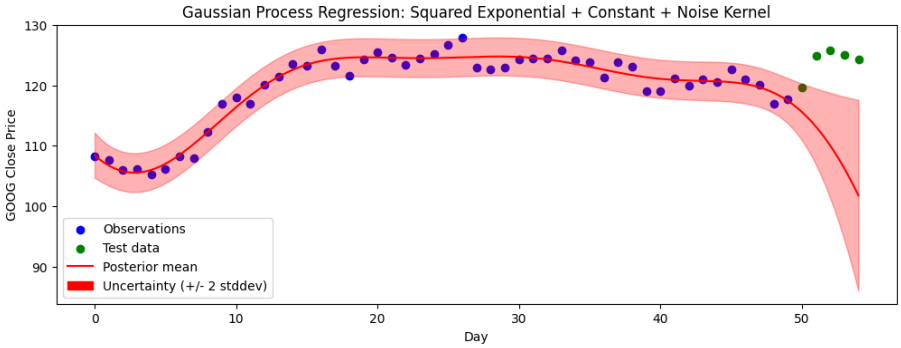
Now, we modify the kernel in Eq.(E.2). A linear kernel was added to the kernel:

$$k = k_{\text{RBF}} + k_{\text{Linear}} + k_{\text{Linear}} + k_{\text{Local Periodic}} + k_{\text{RQ}} + k_{\text{Noise}}. \tag{E.3}$$

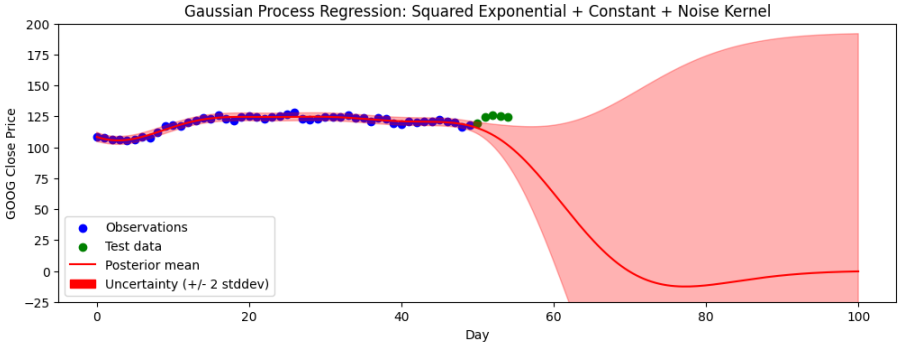
Figure E.8 shows the result of using this combined kernel functions. Surprisingly, even when the linear kernel was included in the Gaussian process along with the squared exponential, linear, local periodic, short-medium term irregularities, and noise kernels, the posterior mean tended to converge to zero.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 125.

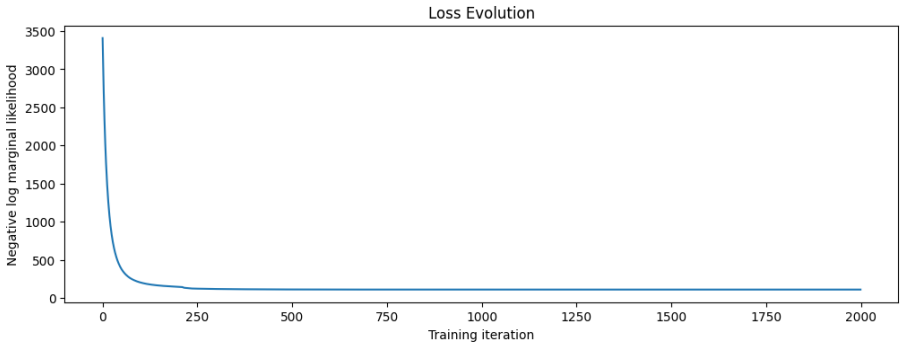


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

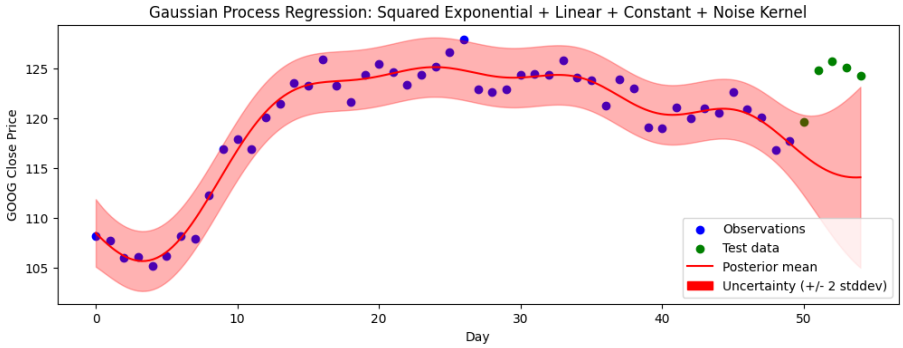


(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

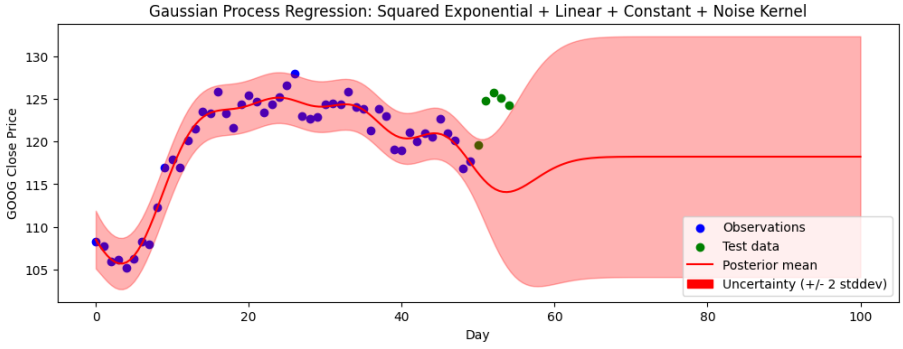
Figure E.4: The result after 2,000 epochs of training the Gaussian process with squared exponential + constant + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 250.

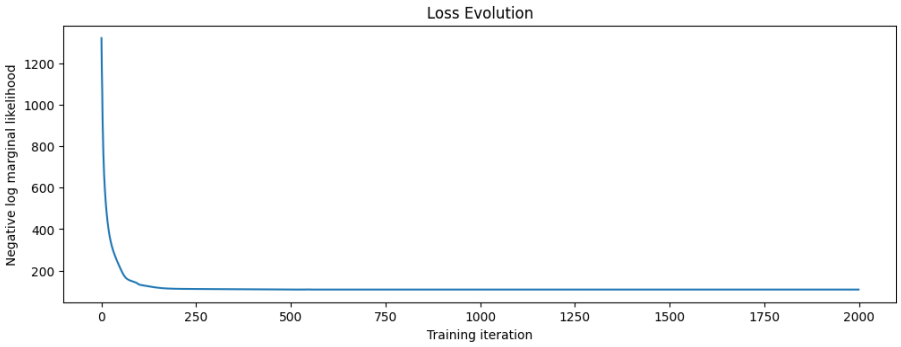


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

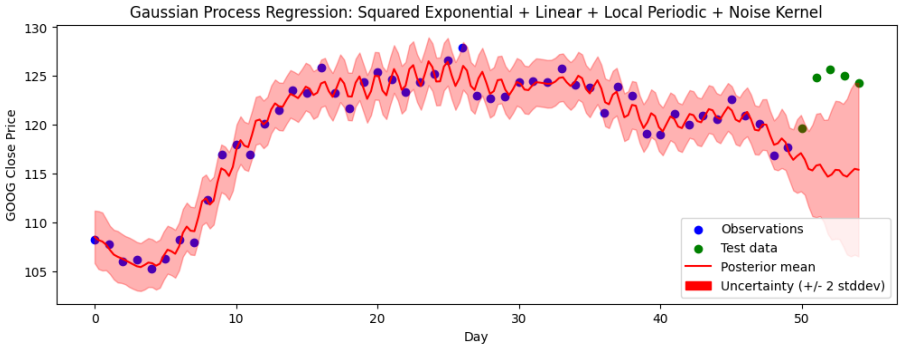


(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

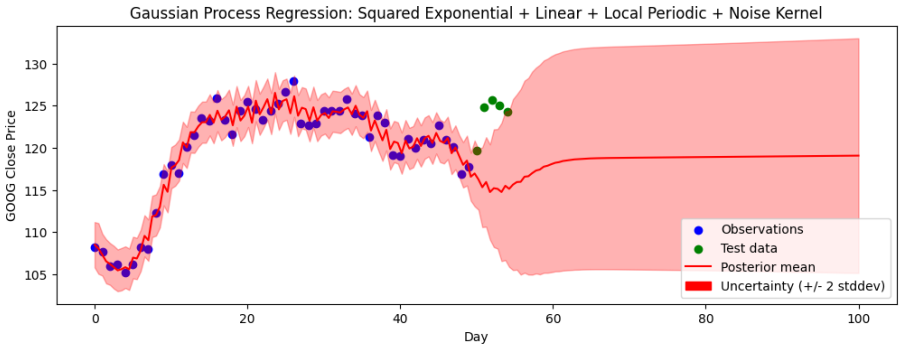
Figure E.5: The result after 2,000 epochs of training the Gaussian process with squared exponential + linear + constant + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 200.

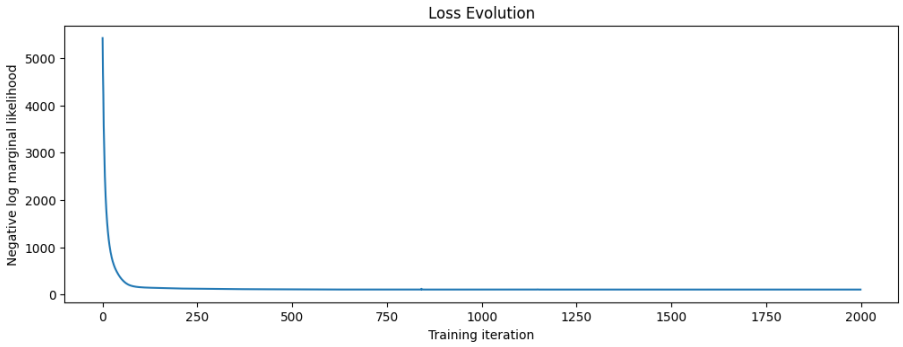


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).

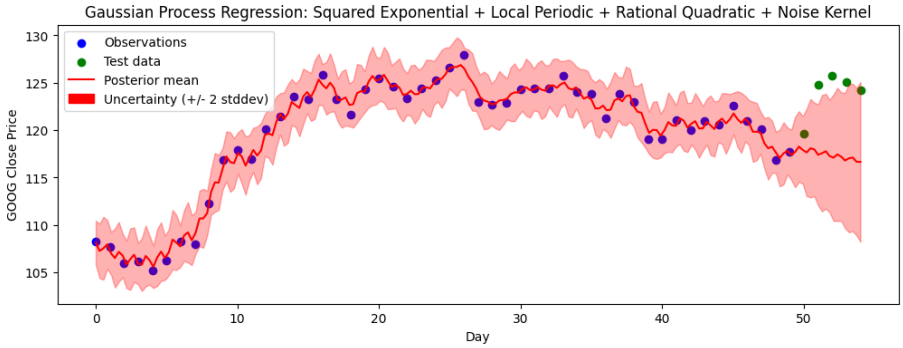


(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

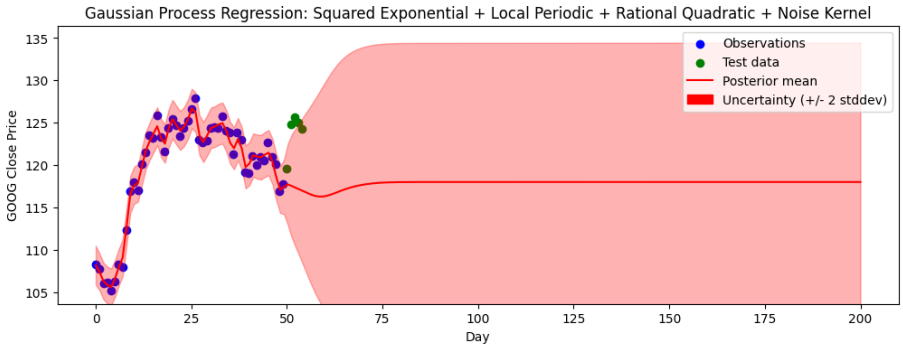
Figure E.6: The result after 2,000 epochs of training the Gaussian process with squared exponential + linear + local periodic + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 200.

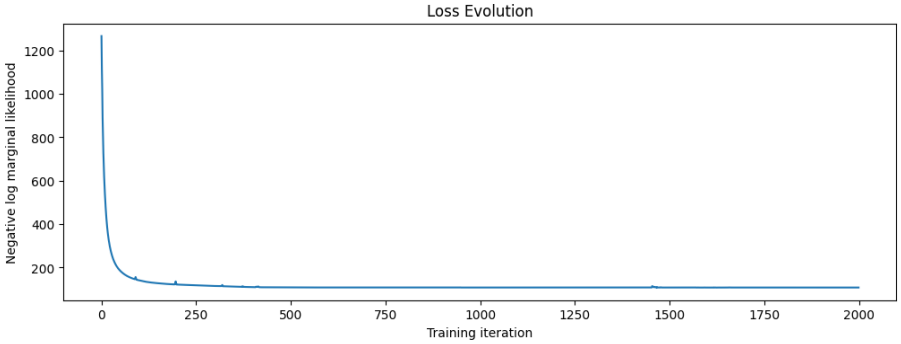


(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

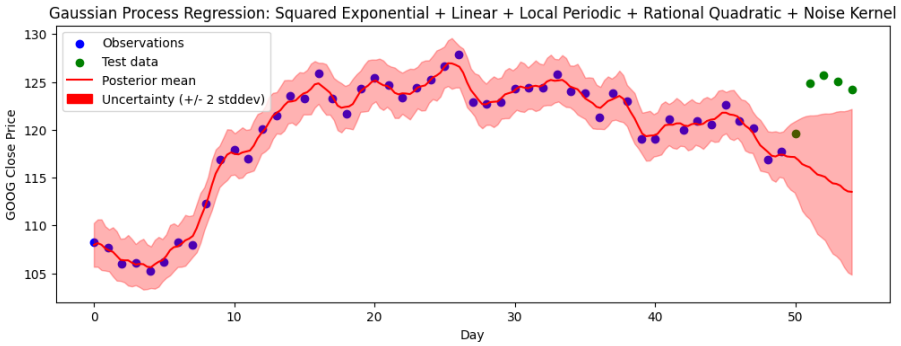


(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

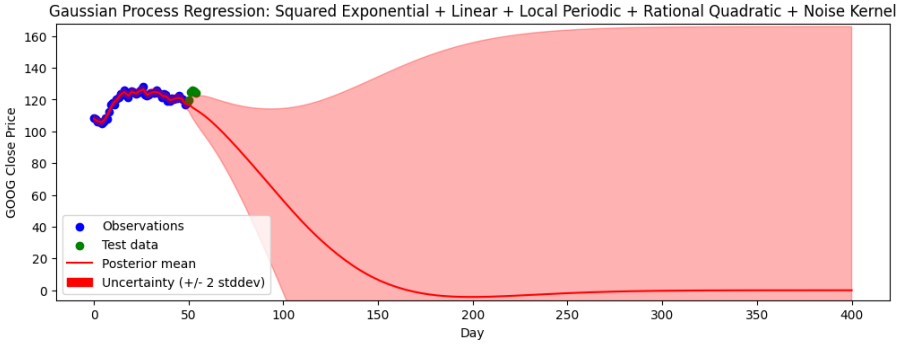
Figure E.7: The result after 2,000 epochs of training the Gaussian process with squared exponential + local periodic + rational quadratic + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.



(a) The loss function (negative log marginal likelihood) updated in each epoch. It leveled out at around epoch 300.



(b) The posterior by the Gaussian process regression model, including 95% confidence band (2σ).



(c) The posterior by the Gaussian process regression model, including 95% confidence band (2σ) on further test inputs.

Figure E.8: The result after 2,000 epochs of training the Gaussian process with squared exponential + linear + local periodic + rational quadratic + noise kernel. (a) The loss function (negative log marginal likelihood) for each iteration. (b) The posterior distribution of the model. (c) The posterior of the model on further test inputs.

Bibliography

Duvenaud, D. (2014). *Automatic Model Construction with Gaussian Processes*. Ph.D. thesis. University of Cambridge.

Duvenaud, D. (2014). *The Kernel Cookbook: Advice on Covariance functions*. [Online]. [Accessed 27 August 2023]. Available from: https://www.cs.toronto.edu/∼duvenaud/cookbook/.

D'Elia, M. et al. (2023). *Stochastic Methods in Advanced Scientific Computing*. United Kingdom: Chapman and Hall/CRC.

Gonzalves, J. et al. (2019). Financial Applications of Gaussian Processes and Bayesian Optimisation. Available at SSRN: https://ssrn.com/abstract=3344332 or http://dx.doi.org/10.2139/ssrn.3344332

Kaelbling, L. (2019). *Feature representation*. Lecture notes distributed in 6.036 Introduction to Machine Learning. 18 December, Massachusetts Institute of Technology.

Kingma, D.P. & Ba, J.L. (2017). Adam: A Method for Stochastic Optimization. *Machine Learning*. **9**. arXiv no:1412.6980. [no paginations].

Li, H. (2018). *Gaussian Process Regression - A Machine Learning Approach to Derivative Pricing*. MSc thesis. Imperial College London.

Neal, R.M. (1996). *Bayesian Learning for Neural Networks*. Lecture Notes distributed in 118 Statistics. New York: Springer.

Rasmussen, C.E. & Williams, C.K.I. (2006). *Gaussian Processes for Machine Learning*. 2nd edition. Massachusetts Institute of Technology: MIT Press.

Roelants, P. (2018). *Multivariate Normal Distribution*. [Online]. [Accessed 27 August 2023]. Available from: https://peterroelants.github.io/posts/multivariate-normal-primer/.

Roelants, P. (2019). *Gaussian Processes (1/3) - From Scratch*. [Online]. [Accessed 27 August 2023]. Available from: https://peterroelants.github.io/posts/gaussian-process-tutorial/.

Roelants, P. (2019). *Gaussian Processes (2/3) - Fitting a Gaussian Process Kernel*. [Online]. [Accessed 27 August 2023]. Available from: https://peterroelants.github.io/posts/gaussian-process-kernel-fitting/.

Roelants, P. (2019). *Gaussian Processes (3/3) - Exploring Kernels*. [Online]. [Accessed 27 August 2023]. Available from: https://peterroelants.github.io/posts/gaussian-process-kernels/.

TensorFlow. (2022). *Gaussian Process Regression in TensorFlow Probability*. [Online]. [Accessed 27 August 2023]. Available from: https://www.tensorflow.org/probability/examples/Gaussian_Process_Regression_In_TFP

TensorFlow. (2023). *Adam Optimizer*. [Online]. [Accessed 27 August 2023]. Available from: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam

TensorFlow. (2023). *Gaussian Process*. [Online]. [Accessed 27 August 2023]. Available from: https://www.tensorflow.org/probability/api_docs/python/tfp/distributions/GaussianProcess

TensorFlow. (2023). *Gaussian Process Regression Model*. [Online]. [Accessed 27 August 2023]. Available from: https://www.tensorflow.org/probability/api_docs/python/tfp/distributions/GaussianProcessRegressionModel#posterior_predictive

Yahoo Finance. (2023). Alphabet Inc. (GOOG) Stock Market. *Yahoo Finance*. [Online]. [Accessed 18 July 2023]. Available from: https://finance.yahoo.com/quote/GOOG/history?period1=1658102400&period2=1689724800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true