TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

institute of telecommunications

Efficient Wireless Coverage Maps using Sparse Gaussian Processes

Exploring Applications of Gaussian Processes in Mobile Cellular Networks and Real-World Train Connectivity Scenarios

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Master's Degree Program Telecommunications

by

Aleksandra Dordevic, BSc
Registration Number 12002193

to the Faculty of Electrical Engineering and Information Technology

at the TU Wien

Advisor: Univ.Prof. Dipl.-Ing. Dr.techn. Markus Rupp
Assistance: Senior Scientist Dipl.-Ing. Dr.techn. Philipp Svoboda

Vienna, 20th June, 2023

Aleksandra Dordevic	Markus Rupp

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Declaration of Authorship

Aleksandra Dordevic, BSc

I hereby declare that I have written this Master's Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

Vienna, 20th June, 2023

Aleksandra Dordevic

Abstract

This thesis provides a comprehensive understanding of Gaussian Processes and their possible applications in the field of wireless networks with a specific focus on railway communication scenarios. It highlights the potential of Gaussian Processes in addressing various challenges in wireless communications, like improving network efficiency and performance but also predicting and modeling certain Key Performance Indicators. Gaussian Processes are a powerful tool that offers a flexible and non-parametric approach to capturing complex patterns and uncertainties. While Gaussian Processes are versatile and sophisticated models, their computational demands can pose challenges, particularly for large data sets, making them computationally unfeasible in such cases. To address scalability challenges, we explore Sparse Gaussian Processes to mitigate computational complexity while maintaining predictive performance. We provide an overview of existing Sparse Gaussian Process models, compare selected methods, and benchmark their performance. We then delve into specific potential use cases of Gaussian Processes within wireless networks, such as traffic prediction, localization, trajectory planning for Unmanned Aerial Vehicles, signaling reduction, and performance evaluation. Lastly, we discuss the specific scenario of railway connectivity which poses its own challenges. A measurement campaign onboard a train, and an accompanying statistical analysis of the observed Reference Signal Receive Power (RSRP) values along the train route connecting Vienna and Innsbruck are conducted. We derive a Path Loss model specific to the railway connectivity scenario. Additionally, we utilize Gaussian Processes to estimate spatial correlations between measured RSRP values, enhancing our understanding of wireless propagation effects that occur onboard trains.

Contents

Abstract v

Contents vii

1 Introduction 1

2 Gaussian Process Regression 5
2.1 Gaussian Process Regression Fundamentals . . . 6
2.2 Gaussian Process Regression Mathematical Model . . . 10
2.2.1 Prediction . . . 11
2.3 Covariance functions and hyperparameters . . . 13
2.3.1 Covariance functions . . . 13
2.3.2 Learning the hyperparameters . . . 15
2.3.3 Estimating the hyperparameters . . . 17

3 Sparse Gaussian Processes 23
3.1 Introduction to Sparse Gaussian Process Regression . . . 23
3.2 Sparse Gaussian Process Regression Mathematical Model . . . 25
3.2.1 Subset of Data . . . 25
3.2.2 Projected Process Approximation . . . 25
3.3 Choosing the inducing points . . . 28
3.3.1 Random subsampling . . . 29
3.3.2 Clustering . . . 29
3.3.2.1 K-means clustering . . . 30
3.3.2.2 K-means initialization: K-means++ . . . 31
3.3.3 Deterministic subset selection . . . 31
3.3.3.1 Maximum Entropy Sampling . . . 31
3.3.3.2 Informative Vector Machine . . . 34
3.3.4 Variational learning of inducing inputs . . . 34
3.3.4.1 Implementation - Derivation of the SGPR equations . 38
3.3.5 Optimization of inducing inputs . . . 40
3.4 Comparison of different strategies . . . 40

4 Gaussian Processes in Wireless Communication 45

4.1 Gaussian Processes in Wireless Networks . . . 45
4.1.1 Traffic Prediction . . . 45
4.1.2 Localization . . . 46
4.1.3 UAV Trajectory Planning . . . 47
4.1.4 Signalling Reduction . . . 48
4.1.5 Performance Evaluation . . . 49

5 Railway Connectivity and Gaussian Processes 51
5.1 Achieving connectivity onboard trains . . . 52
5.2 Data Acquisition and Preprocessing . . . 54
5.2.1 Localization . . . 54
5.3 Data analysis . . . 56
5.3.1 Propagation effects in wireless communication . . . 56
5.3.2 Base Station Mapping . . . 57
5.3.3 Classification . . . 59
5.3.4 Path Loss estimation . . . 61
5.3.5 Gaussian Processes in Railway Propagation Modeling . . . 62

6 Conclusion and Outlook 67

List of Figures 69

Acronyms 71

Bibliography 75

CHAPTER 1

Introduction

Gaussian Processes (GPs) have emerged as a powerful tool in machine learning and statistical modeling, offering a flexible and non-parametric approach to capture complex patterns and uncertainties. In this thesis, we explore GPs and their possible applications in the field of wireless networks, with a special focus on mobile cellular networks.

Initially introduced in the field of probability theory, they provide a framework for probabilistic modeling and regression. They allow us to model the underlying function generating our data, making them suitable for tasks such as regression, classification, and time series analysis. The history of GPs could be traced back to works of mathematicians like Andrey Kolmogorov and Norbert Wiener in the 1940s. In the work by Daniel Krige [1] from 1951 the concept of GPs for regression is used in geostatistics. For this reason, Gaussian Process Regression (GPR) is also sometimes referred to as Kriging. Since then, this method has consistently garnered attention from researchers. However, the true surge of interest in GPs occurred when kernel methods became popular in machine learning. Works by researchers such as David MacKay [2] and Carl Edward Rasmussen [3] particularly contributed to the prominence of GPs in the area of machine learning.

The popularity of GPs is due to their versatility and effectiveness in various domains. They are commonly used in regression tasks for accurate predictions with uncertainty quantification. Classification problems also benefit from the probabilistic nature of the method, which enables reliable class probability estimates and uncertainty characterization. In time series analysis they are able to capture temporal dependencies and make robust forecasts. Chapter one of this thesis focuses on GPR. We will develop the GPR mathematical model and explore the crucial aspect of model selection and hyperparameter optimization. To complement the strict mathematical formulation that will be presented, we will try and develop an intuition of the role and meaning of all components of the GPR model. Special attention will be given to the role of hyperparameters, as well as to a simple and intuitive method for their estimation.

Within the field of wireless networks, GPs find applications in several key areas, with the aim to address challenges in and improve the performance of wireless networks. GPs are just one of many methods that have emerged in a broader movement toward machine learning and artificial intelligence in wireless networks. The pervasive adoption of machine learning in wireless networks in general, but especially in mobile cellular networks, with the advent of 5G and beyond [4][5], further amplifies the importance of exploring and understanding such algorithms and their potential contributions to improving network efficiency and performance. Chapter four of this thesis delves deeper into some specific use cases of GPs in the field of wireless networks: traffic prediction, localization, Unmanned Aerial Vehicle (UAV) trajectory planning, signaling reduction, and performance evaluation. In each of these use cases, Gaussian processes offer unique advantages and have been extensively explored in relevant publications that we will examine and summarize.

It is no secret that wireless connectivity and mobile networks have become ubiquitous in our daily lives. The number of mobile users continues to grow rapidly [6], and so do their demands in terms of the amount of traffic [7] and expectations of uninterrupted connectivity. This presents challenges for network operators and researchers to provide robust and reliable wireless communication. Additionally, with the rise of the Internet of Things (IoT), the number of connected devices is expected to increase exponentially, as more and more devices become smart and rely on wireless connectivity. This further emphasizes the need for efficient and scalable solutions in wireless networks.

Massive connectivity and the accompanying explosion of data pose significant challenges for the aforementioned machine learning approaches in mobile networks. Dealing with large data sets, commonly referred to as big data, is a hurdle for many machine learning and artificial intelligence applications. GPs, in particular, are known to face scalability issues when confronted with large data sets. As a result, finding efficient ways to handle big data becomes crucial for leveraging the benefits of GPs in wireless network applications.

To address the scalability problem, chapter three of this thesis focuses on variants of GPs called Sparse Gaussian Processes (SGPs). These methods aim to overcome the computational complexity of GPs making them more scalable. The chapter provides an overview of existing literature on SGPs, summarizes and compares selected methods, and benchmarks their performance. In the telecommunications domain, where big data poses challenges for Gaussian processes, SGPs offer promising solutions to enable efficient modeling and analysis.

In the final chapter, chapter five, of this thesis, we apply GPR to a data set from a measurement campaign conducted on trains, with mobile phones as measurement devices connected to an operational mobile network. This chapter explores the often overlooked area of onboard train connectivity, where reliable network connectivity is crucial for both passenger satisfaction and operational efficiency. We will start off by clarifying why this communication scenario is challenging, and then examine potential solutions for train connectivity. We will compare the benefits, and drawbacks of each solution.

Moreover, we will analyze network performance, particularly received power levels, in different environments, including urban and rural areas. GPR is utilized to estimate the spatial correlation properties of the received signal power levels in different environments. Finally, we will use the knowledge of these spatial correlation properties to discuss a necessary sampling frequency obtained from GPR models.

In summary, this thesis aims to harness the power of GPs in the context of wireless networks, addressing challenges related to scalability, connectivity, and performance analysis. By exploring specific use cases, investigating scalable variants of GPs, and applying them to real-world train connectivity scenarios, we aim to contribute to the advancement of wireless communications research and enable improved wireless network experiences for users.

CHAPTER 2

Gaussian Process Regression

Supervised learning is a machine learning approach where a model is trained using labeled data. Labeled data consists of input samples usually combined in a vector and denoted by $\mathbf{x}$ and corresponding output labels or target values usually denoted by y. In supervised learning approaches we assume that the input vector $\mathbf{x}$ is a set of features explaining the underlying process. For example, in a task of predicting the probability an individual might suffer from heart failure, input features might be age, blood pressure, weight, and so on. From the set of input feature values and target output values, we are trying to uncover the hidden process modeling the input-output relationship. We assume this process can be modeled sufficiently well by a functional mapping $\mathbf{y} = f(\mathbf{x})$ and we are trying to learn the function f. Once we have obtained this hidden function f we can find output values for input feature sets $\mathbf{x}^*$ that we have not seen before, usually this is called the test data set. Feature sets $\mathbf{x}$ and the corresponding labeled output values y that are observations or measurements used for model fitting are usually called the training set. Real-world data is in most cases such that this hidden functional mapping will not be exact, i.e. that there is no function f that fits the data without errors.

In supervised learning, classification and regression are two primary types. Classification predicts categorical class labels, such as classifying emails as spam or not spam, handwritten digits in image recognition, or determining whether a patient has a certain disease based on medical test results. Regression, on the other hand, predicts continuous values, such as predicting housing prices based on features like location, size, and number of rooms, estimating the amount of rainfall based on temperature and humidity, or forecasting stock market prices. Other types of supervised learning include multi-label classification, which assigns multiple labels to data instances, like tagging images with multiple objects, for example, a photo with both a dog and a cat. Ordinal regression handles ordered categories, such as predicting customer satisfaction levels ranging from very unsatisfied, to somewhat satisfied, or highly satisfied. Time series forecasting predicts

future values based on historical data, like predicting future sales based on past sales records.

Gaussian Processes (GPs) can be used as a powerful regression tool and we will explore them in the rest of this chapter. One thing to note is that, in this work, but also many others concerning Gaussian Process Regression (GPR), the $\mathbf{x}$ is referred to be a measurement location instead of a set of features. There is little difference in the approach no matter the interpretation of the input vector $\mathbf{x}$ in a strict mathematical sense. However, naming $\mathbf{x}$ a location implies that we are dealing with spatial coordinates, which works particularly well in many tasks that deal with real-world geographical data, as for example in chapter 5. We assume we have some area of interest and sensors, or measurement positions, dispersed throughout it and some quantity we would like to measure, or infer the value of, on the whole area. This target quantity could be temperature, air quality, or in the case of mobile networks, received power levels. As in this work, we are mostly focused on this GPR use case, we will use the terms training or test point or training or test location, or input interchangeably.

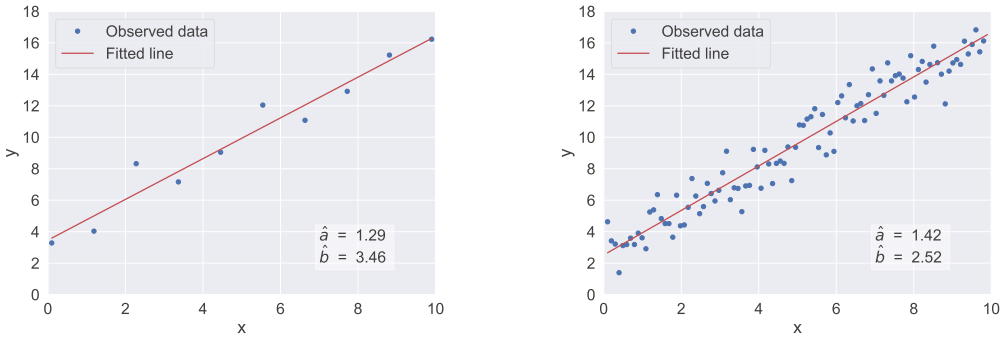
In essence GPR is an interpolation technique, albeit a sophisticated one. Other interpolation techniques include for example spline and polynomial regression. Spline interpolation involves fitting a piecewise-defined curve consisting of multiple cubic polynomial functions that join smoothly at the data points, while polynomial regression fits a single polynomial function to the data. Both of these methods can be useful for certain types of data, but they have limitations. One of the main advantages of GPR is its ability to model complex, non-linear relationships between variables, without making strict assumptions about the underlying data distribution. It can also be used to model noisy data, by taking into account the uncertainty associated with each observation. Its ability to provide an uncertainty measure for each prediction is another key advantage of GPR. This means that it can estimate how confident it is in its predictions, based on the amount of noise, or uncertainty, in the data. This is particularly useful in situations where there is measurement noise, as it can help to identify when the predictions are less reliable. Neither spline nor polynomial regression have this advantage.

2.1 Gaussian Process Regression Fundamentals

We will start of by defining Gaussian Processes in a single sentence and than break down each of the components in that definition. In short: *A Gaussian Processes is a non-parametric Bayesian model that allows for flexible and probabilistic regression by modeling the distribution of possible functions that could explain the observed data.*

First, we will tackle the difference between a parametric and a non-parametric model. A good overview of popular machine learning models and their classification into parametric and non-parametric can be found in [8]. A parametric model in machine learning is a type of model that assumes a specific form for the relationship between input and output variables, such as a linear or polynomial equation. The model is defined by a fixed number of parameters that are learned from the training data. Parametric models

are often simpler and more interpretable than non-parametric models, but may not be as flexible in handling complex or unknown data distributions. An example of a parametric model is shown in Figure 2.1. The relationship between input x and output y is linear, but noisy. We solve for the parameters by using least squares. With a growing number of observations the estimation error decreases, but the total number of parameters remains the same - two.



(a) with 10 observations (b) with 100 observations

Figure 2.1: A linear parametric model
$a_{true} = 1.4$
$b_{true} = 2.7$

A non-parametric model is a type of model that does not make an assumption that the underlying relationship between input and output variables can be explained by a **fixed** number of parameters. Non-parametric models can sometimes still be fully explained by a set of parameters, but their number is not predetermined and usually depends on the size of the training dataset. Non-parametric models are particularly useful when the distribution of the data is unknown or complex as they can use more flexible functions to estimate the input-output relationship from the data. In GPs, the number of *parameters* to be estimated is the same as the number of points in the input dataset. That means that, with growing training set size, the number of parameters also incrases and is therefore not fixed. The exact meaning of these parameters will be explained later. Sometimes in literature, GPs are described as non-parametric models with an infinite number of parameters. This can be explained as the following, if we had an input training set of an infinite cardinality, the number of parameters would also be infinite. In reality, data sets can be large but never infinite, so one must not worry about estimating an infinite number of parameters.

Secondly, we will show how GPs fit into the Bayesian framework. A Bayesian model is a probabilistic model that uses Bayes' theorem to update the model's belief or probability distribution based on new evidence or data, as shown in Figure 2.2. The Bayesian cookbook is more easily explained in the framework of a parametric model, so we will temporarily use the parametric notation. Suppose we are tasked with estimating p

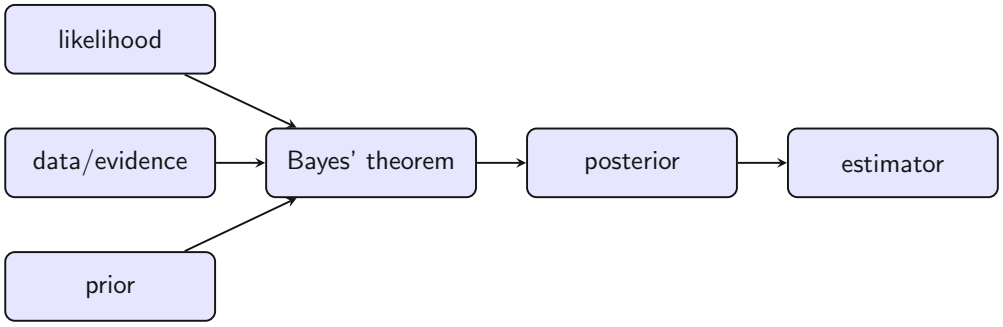


Figure 2.2: Bayes' flow

parameters $\boldsymbol{\theta} = [\theta_1, ..., \theta_p]^T$ from N observed data points $\boldsymbol{x} = [x_1, ..., x_N]^T$, and we know the prior and likelihood functions. One simple example of such a model is linear regression with Bayesian inference, where our parameter vector has two elements $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$, namely the slope and the intercept, similar to the coefficients a and b from the previous example. The difference to the previous model is that, in the Bayesian approach, both of the coefficients are random variables. Another key difference is that the data itself is also considered to be random with an appropriate probability distribution. By using Bayes' theorem:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \underbrace{f(\mathbf{x}|\boldsymbol{\theta})}_{\text{likelihood}} \underbrace{f(\boldsymbol{\theta})}_{\text{prior}} = \underbrace{f(\boldsymbol{\theta}|\mathbf{x})}_{\text{posterior}} \underbrace{f(\mathbf{x})}_{\text{evidence}} \tag{2.1}$$

we can calculate the posterior distribution $f(\boldsymbol{\theta}|\mathbf{x})$, which is the primary goal of any Bayesian method. Using the obtained posterior, we can then derive different estimators for the parameter vector, such as for example the Minimum Mean Square Error (MMSE) or Maximum A Posteriori (MAP) estimator. The MMSE estimator minimizes the expected Mean Squared Error (MSE), and it is given by $\hat{\boldsymbol{\theta}}_{MMSE} = \mathbb{E}[\boldsymbol{\theta}|\mathbf{x}]$. The MAP estimator is the parameter value for which the posterior distribution is at its maximum, i.e. the mode of the posterior distribution $\hat{\boldsymbol{\theta}}_{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} f(\boldsymbol{\theta}|\mathbf{x})$.

In a non-parametric model the Bayes flow remains the same, we are however challenged with an unknown length or interpretation of the parameter vector. In the Bayesian framework we assume the prior $f(\boldsymbol{\theta})$ and the conditional distribution $f(\mathbf{x}|\boldsymbol{\theta})$ are known or predetermined in some way. The choice of the prior contains knowledge of the parameters prior to seeing any data. This might be counter-intuitive, as we are tasked with providing a full probability distribution of parameters before learning anything from the data. This is an easily solvable problem in some cases, we can just specify a so called non-informative prior. That would be for example a very wide uniform distribution. However, in GPs the choice of the prior cannot be that non-informative and simple. We will discuss the choice of the prior shortly.

And lastly, we show how a GP can be formulated as a probability distribution of possible

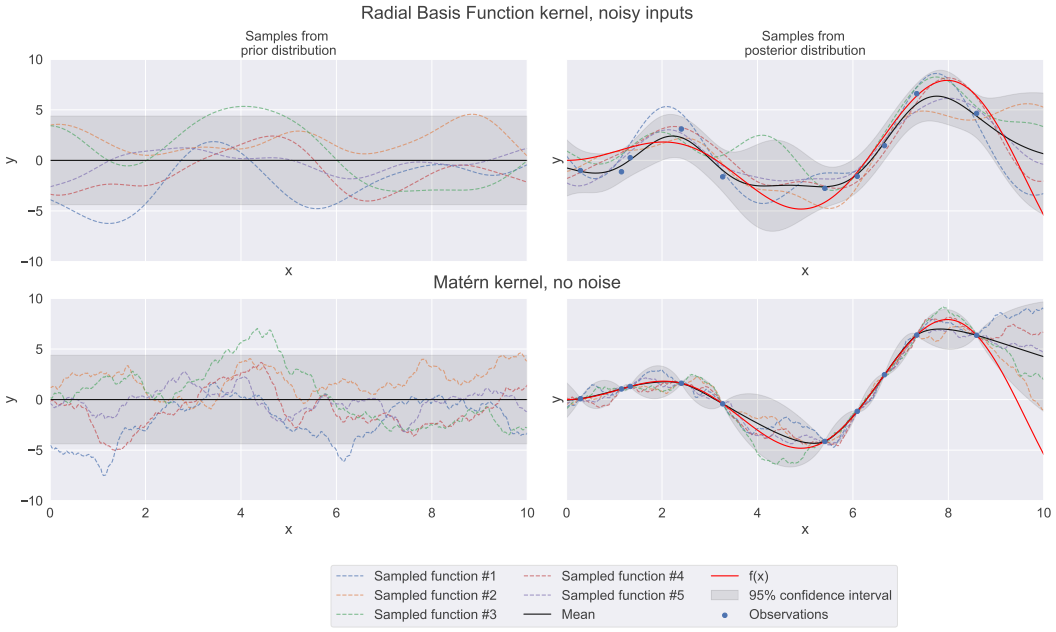


Figure 2.3: Prior and posterior samples with different kernel choices

functions that explain the observed data. A Gaussian process is a probabilistic model that defines a distribution over functions. It assumes that any set of function evaluations follows a multivariate Gaussian distribution, with the covariance between the evaluations determined by some known function, that we call the covariance or kernel function. Samples from the GP prior and posterior are full functions, as shown in Figure 2.3. The shape of the function samples is determined by the choice of kernel, which contains our prior knowledge about the shape of the function. In Figure 2.3 we see two examples for the Matérn and Radial Basis Function (RBF) kernel. The prior samples have different shapes. The posterior samples retain the overall shape but are passing through the data points. In the case of the Matérn kernel samples are assumed to have no noise so all posterior samples pass exactly through the given data points, and at those locations, the uncertainty is zero. In the case of the RBF kernel, we assume a low noise level so posterior samples do not all pass exactly through the data points, rather only close to them. In this noisy case, the uncertainty interval is also no longer zero at any point.

As we also see from 2.3, by selecting different covariance functions, that characterize the correlations between nearby locations, GPR adds prior knowledge about the shape and the spatial consistency of the underlying function. Therefore, the proper kernel choice and proper choice of values of kernel parameters are essential to obtain good interpolation results. In the case of GPR there is no easy way out in terms of specifying a prior, such as for example setting a wide uniform distribution as a non-informative prior. That being the case, choosing the best kernel is usually a trial-and-error operation as there is no widely accepted automatic model selection procedure as of yet. Some additional

discussion on different kernel options and their benefits and drawbacks will be provided in the next section. The method for learning the kernel hyperparameters will also be discussed in detail in the following text.

2.2 Gaussian Process Regression Mathematical Model

After gaining some intuition on what GPR is we will now delve into the concrete mathematical description of the model. Assume we are given a set of real input vectors $\mathbf{x}_i \in \mathbb{R}^D,\ i = 1, 2,\ ...,\ N$ and a set of noisy observations for each input vector $y_i \in \mathbb{R},\ i = 1, 2,\ ...,\ N$. Let us suppose that there is some true, underlying latent function, $f : \mathbb{R}^D \to \mathbb{R}$ explaining the observed outputs, such that the measurements y_i are only corrupted by random Gaussian noise. We are only discussing cases where the measurement noise is considered to be Gaussian, which can be written as:

$$y_i = f(\mathbf{x}_i) + \epsilon_i \tag{2.2}$$

The observation error ϵ_i is additionally assumed to be statistically independent of $f(\mathbf{x})$ at all observation locations $\mathbf{x}$, and mutually statistically independent, and zero mean with some possibly unknown variance σ_ϵ, which we write as:

$$\epsilon_i \sim\ \mathcal{N}(0, \sigma_\epsilon^2) \tag{2.3}$$

$$\mathrm{cov}(\epsilon_i, \epsilon_j) = \sigma_\epsilon^2 \delta_{i,j} = \begin{cases} \sigma_\epsilon^2 & i = j \\ 0 & i \neq j \end{cases} \tag{2.4}$$

Additionally, we will model $f(\mathbf{x})$ as a random process with some statistical structure, a process so defined is called a Gaussian Process. Formally, we define it, following [3], as:

Definition 2.2.1. A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

And we can briefly write this as:

$$f(\mathbf{x}) \sim\ \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \tag{2.5}$$

with a mean function $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$ defined as:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})] \tag{2.6}$$

$$\mathrm{cov}(f(\mathbf{x}), f(\mathbf{x}')) = k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \tag{2.7}$$

As per definition 2.2.1 we can easily determine the joint probability distribution for any finite set of J function values, combined in a random vector $\mathbf{f} \coloneqq [f(\mathbf{x}_1)...f(\mathbf{x}_J)]^T = [f_1...f_J]^T$ as:

$$\begin{aligned} p(\mathbf{f}) &= p(f(\mathbf{x}_1), ..., f(\mathbf{x}_J)) = p(f_1, ..., f_J) = \\ &= \frac{1}{\sqrt{(2\pi)^J \det(\mathbf{K})}} \exp\left(-\frac{1}{2}(\mathbf{f} - \boldsymbol{m})^T \mathbf{K}^{-1} (\mathbf{f} - \boldsymbol{m})\right) \end{aligned} \tag{2.8}$$

where:

$$\boldsymbol{m} = \mathbb{E}[\mathbf{f}] = [m(\mathbf{x}_1) \ ... \ m(\mathbf{x}_J)]^T \tag{2.9}$$

$$\mathbf{K} = \mathbf{K}(\mathbf{X}, \mathbf{X}) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \dots & k(\mathbf{x}_1, \mathbf{x}_J) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \dots & k(\mathbf{x}_2, \mathbf{x}_J) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_J, \mathbf{x}_1) & k(\mathbf{x}_J, \mathbf{x}_2) & \dots & k(\mathbf{x}_J, \mathbf{x}_J) \end{bmatrix} \tag{2.10}$$

with $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_J]^T$, in which the data is simply ordered in a matrix form. With this vector-matrix notation we can write 2.5 equivalently as:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \boldsymbol{m}, \mathbf{K}) \tag{2.11}$$

Having the statistical properties of latent function values and the additive noise defined we can now easily define the statistics of the random observations y_i too. From 2.2 we get:

$$\mathbb{E}[y_i] = \mathbb{E}[f(\mathbf{x}_i)] = m(\mathbf{x}_i) \tag{2.12}$$

$$\text{cov}(y_i, y_j) = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) + \text{cov}(\epsilon_i, \epsilon_j) = k(\mathbf{x}_i, \mathbf{x}_j) + \sigma_\epsilon^2 \delta_{i,j} \tag{2.13}$$

Or, defining a vector of all observations $\mathbf{y} = [y_1, ..., y_N]^T$, all latent function values $\mathbf{f} = [f_1, ..., f_N]^T$ and all noise values $\boldsymbol{\epsilon} = [\epsilon_1, ..., \epsilon_N]^T$ that are now of length N and not some arbitrary J, we can write more compactly:

$$\mathbf{y} = \mathbf{f} + \boldsymbol{\epsilon} \tag{2.14}$$

$$\mathbb{E}[\mathbf{y}] = \boldsymbol{m} \tag{2.15}$$

$$\text{cov}(\mathbf{y}) = \mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N \tag{2.16}$$

$$\mathbf{y} \sim \ \mathcal{N}(\mathbf{y}; \boldsymbol{m}, \mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N) \tag{2.17}$$

If $\mathbf{f}$ is already observed, the distribution becomes trivially:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}; \mathbf{f}, \sigma_\epsilon^2 \mathbf{I}_N) \tag{2.18}$$

2.2.1 Prediction

Following the discussion on Bayesian models we can formulate the estimation problem through Bayes' theorem. Here, we define the *parameters of the model* as the N latent function values, at the input locations $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$. The locations are not random, but the observations y_i, function f_i and noise ϵ_i values are. Additionally, a covariance function is usually defined with some parameters, stacked in a vector that we will denote by $\boldsymbol{\theta}$. These parameters are called hyperparameters and their choice and optimization methods will be explained in Sections 2.3.1 and 2.3.2. In short, they are found through a log marginal likelihood optimization procedure. It is very important to keep in mind

that we are now dealing with two sets of parameters that are treated differently. The first is the set of N latent function values that are treated as parameters to be estimated, as in the previously mentioned Bayesian flow shown in Figure 2.2 and Bayes' theorem 2.1. The second is the set of kernel hyperparameters that can be considered fixed for the purposes of this section, as if given by an oracle. The distributions needed for the regression task are now:

$$p(\mathbf{y}, \mathbf{f}; \mathbf{X}, \sigma^2, \boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{f}; \sigma^2) p(\mathbf{f}; \mathbf{X}, \boldsymbol{\theta}) \tag{2.19}$$

Even though it is important to keep in mind the above distributions depend on non-random variables such as the measurement locations $\mathbf{X}$, the noise variance and the kernel hyperparameters $\boldsymbol{\theta}$ in the following text we will usually drop the above notation and use a shorter version:

$$p(\mathbf{y}, \mathbf{f}) = \underbrace{p(\mathbf{y}|\mathbf{f})}_{\substack{\text{likelihood} \\ \text{- known}}} \underbrace{p(\mathbf{f})}_{\substack{\text{prior} \\ \text{- known}}} = \underbrace{p(\mathbf{f}|\mathbf{y})}_{\text{posterior}} \underbrace{p(\mathbf{y})}_{\substack{\text{evidence} \\ \text{- known}}} \tag{2.20}$$

For the choice of the prior mean we usually take the zero vector $\mathbf{0}_{N\times 1}$, which leads to:

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}; \mathbf{0}, K) \tag{2.21}$$
$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}; \mathbf{0}, K + \sigma_\epsilon^2 \mathbf{I}_N) \tag{2.22}$$
$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}; \mathbf{f}, \sigma_\epsilon^2 \mathbf{I}_N) \tag{2.23}$$

As in every Bayesian model we are looking for the posterior which we can now easily derive by using known properties of Gaussian distributions [3]:

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{f}, \mathbf{y})}{p(\mathbf{y})} = \frac{p(\mathbf{y}|\mathbf{f}) p(\mathbf{f})}{p(\mathbf{y})} = \mathcal{N}(\mathbf{f}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{2.24}$$
$$\boldsymbol{\Sigma} = (\sigma_\epsilon^{-2} \mathbf{I}_N + \mathbf{K}^{-1})^{-1} \tag{2.25}$$
$$\boldsymbol{\mu} = \boldsymbol{\Sigma}(\sigma_\epsilon^{-2} \mathbf{I}_N)\mathbf{y} = \sigma_\epsilon^{-2} \boldsymbol{\Sigma} \mathbf{y} \tag{2.26}$$

Of course, we are not only interested in the latent function values at the training locations but also in test locations where no noisy observations are available. The distribution of function values is jointly Gaussian no matter whether the chosen locations are observed or not. Using the prior assumption on the function value and standard multivariate Gaussian properties we get [3]:

$$p(f(\mathbf{x}^*)|\mathbf{f}; \mathbf{X}, \mathbf{x}^*, \boldsymbol{\theta}) = \mathcal{N}(f(\mathbf{x}^*); \mu_{prior}, \sigma^2_{prior}) \tag{2.27}$$
$$\sigma^2_{prior} = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T \mathbf{K}^{-1} \mathbf{k}(\mathbf{x}^*) \tag{2.28}$$
$$\mu_{prior} = \mathbf{k}(\mathbf{x}^*)^T \mathbf{K}^{-1} \mathbf{f} \tag{2.29}$$

where $\mathbf{k}(\mathbf{x}^*) = [k(\mathbf{x}^*, \mathbf{x}_1), k(\mathbf{x}^*, \mathbf{x}_2), ..., k(\mathbf{x}^*, \mathbf{x}_N)]^T$ is the covariance vector evaluated at the test location $\mathbf{x}^*$. The posterior can be found by integrating out the latent outputs at

the training data as:

$$p(f(\mathbf{x}^*)|\mathbf{y}) = \int p(f(\mathbf{x}^*)|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f} = \mathcal{N}(f(\mathbf{x}^*); \mu_{posterior}, \sigma^2_{posterior}) \tag{2.30}$$

$$\begin{aligned}\sigma^2_{posterior} &= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T\mathbf{K}^{-1}(\mathbf{\Sigma} - \mathbf{K})^{-1}\mathbf{K}^{-1}\mathbf{k}(\mathbf{x}^*) \\ &= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*)^T(\sigma^2_\epsilon\mathbf{I}_N + \mathbf{K})^{-1}\mathbf{k}(\mathbf{x}^*) \qquad (2.31)\\ \mu_{posterior} &= \mathbf{k}(\mathbf{x}^*)^T\mathbf{K}^{-1}\mathbf{\Sigma}(\sigma^2_\epsilon\mathbf{I}_N)\mathbf{y} = \sigma^2_\epsilon\mathbf{k}(\mathbf{x}^*)^T\mathbf{K}^{-1}\mathbf{\Sigma}\mathbf{y} \\ &= \mathbf{k}(\mathbf{x}^*)^T(\sigma^2_\epsilon\mathbf{I}_N + \mathbf{K})^{-1}\mathbf{y} \qquad (2.32)\end{aligned}$$

These can also easily be combined into a vector format as:

$$p(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*; \boldsymbol{\mu}_{posterior}, \mathbf{\Sigma}_{posterior}) \tag{2.33}$$

$$\mathbf{\Sigma}_{posterior} = \mathbf{K}(\mathbf{X}^*, \mathbf{X}^*) - \mathbf{K}(\mathbf{X}^*, \mathbf{X})(\sigma^2_\epsilon\mathbf{I}_N + \mathbf{K})^{-1}\mathbf{K}(\mathbf{X}, \mathbf{X}^*) \tag{2.34}$$

$$\boldsymbol{\mu}_{posterior} = \mathbf{K}(\mathbf{X}^*, \mathbf{X})(\sigma^2_\epsilon\mathbf{I}_N + \mathbf{K})^{-1}\mathbf{y} \tag{2.35}$$

One drawback of GPR is immediately visible from 2.34 and 2.35. In order to calculate the predictive distribution we have to find an inverse of an $N \times N$ matrix, where N is the number of training samples. Matrix inversion is a computationally demanding operation, resulting in an $\mathcal{O}(N^3)$ computational complexity.

2.3 Covariance functions and hyperparameters

We have now defined the statistical properties of all involved quantities with some given covariance, or kernel, function $k(\mathbf{x}, \mathbf{x}')$. As we have seen in the previous section, the GP model heavily depends on the choice of this kernel function. We have so far considered it known, as if given by an oracle. In the following section, we will discuss possible choices for this kernel, or covariance function. As we will see, most of them are parameterized by some finite number of so-called hyperparameters. This following section will additionally discuss the role and choice of hyperparameters too.

2.3.1 Covariance functions

A kernel function is a general name for a function that takes as input two data points and maps them into $\mathbb{R}$

$$k(\mathbf{x}, \mathbf{x}') : \mathbb{R}^D \times \mathbb{R}^D \longrightarrow \mathbb{R} \tag{2.36}$$

with this mapped value usually interpreted as a measure of their similarity. However, not all arbitrary functions can serve as valid covariance functions. In order for a functional mapping to be a valid covariance function it must be positive semidefinite, so that the resulting matrix $\mathbf{K}$ is also a valid covariance matrix, i.e. a positive semidefinite matrix. Additionally, a real kernel is said to be symmetric if $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$ and again valid covariance functions must be symmetric so that they result in a valid symmetric covariance matrix $\mathbf{K}$. A stationary kernel is a function only of the distance between the

Covariance functions overview	
kernel name	mathematical formula
constant	σ^2
linear	$\sum_{d=1}^{D} \sigma_d^2 x_d x'_d$
polynomial	$(\mathbf{x}^T\mathbf{x} + \sigma^2)^p$
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\theta} r\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}}{\theta} r\right)$
exponential	$\exp\left(-\frac{r}{\theta}\right)$
γ - exponential	$\exp\left(-\left(\frac{r}{\theta}\right)^{\gamma}\right)$
rational quadratic	$\left(1 + \frac{r^2}{2\alpha\theta^2}\right)^{-\alpha}$
squared exponential	$\exp\left(-\frac{r^2}{2\theta^2}\right)$
neural network	$\sin^{-1}\left(\frac{2\mathbf{x}^T\Sigma\mathbf{x}'}{\sqrt{(1+2\mathbf{x}^T\Sigma\mathbf{x})(1+2\mathbf{x}'^T\Sigma\mathbf{x}')}}\right)$

Table 2.1: Overview of common covariance functions

two data points $r = |\mathbf{x} - \mathbf{x}'|$ where $|.|$ denotes the Euclidean distance between the two points. This is not a requirement but we will focus mostly on stationary kernels.

Kernel functions are used in kernel methods, such as Support Vector Machines (SVMs) and GPs, to transform the input data into a higher-dimensional feature space where linear algorithms can be more effective. Different kernel functions can be used to capture different types of patterns in the data, such as linear, polynomial, or exponential similarities. The choice of kernel function has a significant impact on the performance of a machine learning model and there are several commonly used kernel functions. An overview is given in Table 2.1 [3]. Note that in 2.7 and 2.13 the covariance between outputs, i.e. latent function values at different measurement locations, is described via the kernel function between the inputs, i.e. the locations themselves. This is a crucial idea in all kernel methods.

Most covariance functions are additionally defined by a finite set of parameters $\boldsymbol{\theta}$. For example, the constant kernel is defined by the sole parameter $\theta = \sigma$, while the Matérn kernel is governed by two parameters $\boldsymbol{\theta} = [\theta, \nu]^T$. The resulting kernel mappings for the same inputs will differ for different choices of these parameters. These are usually called hyperparameters and learning the optimal values of hyperparameters, also called the training procedure or simply training, is a crucial step in fitting a GP regressor to observed data. The widely accepted training procedure is based on the minimization of the log marginal likelihood of the model, which we will show in detail later in section 2.3.2.

A Radial Basis Function (RBF), also called a squared exponential covariance function as in the Table 2.1, is probably the most popular kernel choice. It measures the similarity between two data points based on their distance from each other in the input space. The RBF kernel function has a bandwidth or length-scale parameter θ that determines how quickly the similarity decreases as the distance between the data points increases.

For inputs that are close to each other, i.e. their Euclidian distance is close to 0, the covariance between the function values will be almost 1 denoting a high similarity between the function values. By varying the length-scale parameter we can vary the distance at which the function values become independent of each other. An illustration of this kernel function for varying lenght-scale parameters is shown in Figure 2.4. Additionally, an RBF kernel for two dimensional inputs and a fixed value of $\theta = 1$ is shown in 2.5. The functions drawn from a GP with an RBF kernel are infinitely differentiable which makes them very smooth.

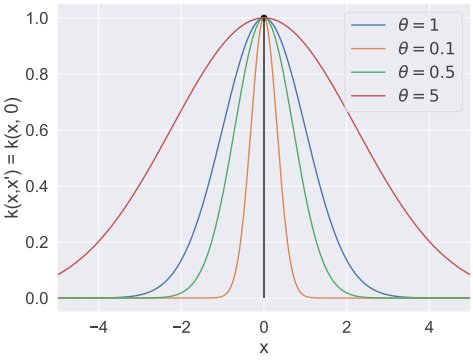


Figure 2.4: RBF kernel shape for different length-scales

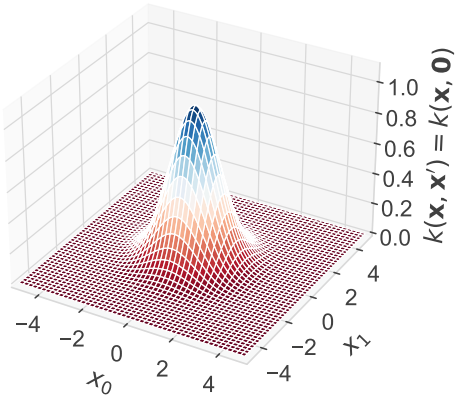


Figure 2.5: RBF kernel shape for two dimensional inputs and $\theta = 1$

2.3.2 Learning the hyperparameters

We will first try to develop some intuition on the meaning of these hyperparameters for the most interesting kernel functions. In Figure 2.3 we can see the resulting mean function of a GP for different values of the length-scale parameter with an RBF kernel, along with the true underlying function shown in red. We assume that there is some white Gaussian noise in the data too. For smaller values, like for example $\theta = 0.1$ the shape of the mean function corresponds to the shape of the kernel placed and scaled at each of the data points. In the interstitial region, there is considered to be no information from the data. That means that all of the observed data points are further away than the decorrelation distance. In these regions the predictor defaults to zero, which is assumed to be the prior mean. Also in this region, we can notice that the uncertainty is at its' maximum. Again, this is because we have no information on the underlying process in this region in the observed data. In this case, the correlation distance is too small compared to the distance between data samples for any reasonable function fitting to occur. For $\theta = 0.5$ the shape of the kernel is still somewhat visible, but the different kernel shapes blend into each other resulting in a continuous shift from zero in the mean function. For $\theta = 5$ the correlation distance is much larger than the spacing between data points, resulting in a very smooth and slowly varying mean function. One can

imagine that for even larger length scales the mean prediction would blend into an almost constant function, explaining the data only as white noise. As we can see, the best fit to the data is achieved for value $\theta = 1$. However, this is clear to us only because we are able to see the underlying function generating the data. Without knowing the true underlying process all of the examples shown are plausible. For example, the underlying function could be fast varying and best explained with $\theta = 0.1$, but we could only have a few data points to work with so the true process is impossible to capture because our data set is too small and dispersed along the x-axis.

Usually, we are interested in the function fit in cases where the underlying process is unknown, so we must have a criterion on how to choose the best hyperparameters when only training data is available. The most common criterion is choosing the hyperparameters such that the log marginal likelihood is maximized. This problem is usually inverted in a way such that we are instead minimizing the negative log marginal likelihood. Using the already derived distribution of the marginal likelihood in 2.22, and keeping in mind that the logarithm preserves the location of the minimum we get the function to be minimized, denoted by $\mathcal{L}$ as:

$$\begin{aligned}\mathcal{L} &= -\log p(\mathbf{y}) = -\log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N) = \\ &= \frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N)^{-1}\mathbf{y} + \frac{1}{2}\log\det(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N) + \frac{N}{2}\log 2\pi\end{aligned} \tag{2.37}$$

and the optimization criterion as:

$$\boldsymbol{\theta}_{opt} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathcal{L}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N)^{-1}\mathbf{y} + \frac{1}{2}\log\det(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N)\right] \tag{2.38}$$

Minimization of the negative log marginal likelihood is commonly used for hyperparameter optimization in GPs because it provides an effective and principled approach to finding optimal hyperparameter values. In GPs, the log marginal likelihood quantifies the goodness-of-fit of the model to the training data. By maximizing the log marginal likelihood, we are effectively finding the hyperparameters that yield the best model fit to the data. The log marginal likelihood incorporates both the model's ability to explain the training data and the complexity of the model. It strikes a balance between model complexity and data fit, avoiding overfitting or underfitting. The first term $\frac{1}{2}\mathbf{y}^T(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N)^{-1}\mathbf{y}$ is the only one that incorporates the data, and it represents how well the model fits to it. The second term $\frac{1}{2}\log\det(\mathbf{K} + \sigma_\epsilon^2 \mathbf{I}_N)$ on the other hand, does not depend on the observed data at all. This term can be interpreted as the complexity penalty, as it depends only on the chosen covariance function.

Minimizing the negative log marginal likelihood is equivalent to maximizing the likelihood of the observed data under the GP model. This is however not a foolproof method, as the marginal likelihood could have multiple local optima. Some data could be explained by multiple combinations of hyperparameters equally well. For example, a smoother function with larger noise or a fast-varying function with low noise levels. Or initial training data might be insufficient to reach a good enough conclusion. As we have mentioned

Figure 2.6: Prediction with an RBF kernel with different length-scale parameters

previously, in Figure 2.3 if the *true* hyperparameter is $\theta = 0.1$ but the training data set consists only of the pictured samples, we have no way to determine the function with such sparse data. The main drawback of this optimization procedure is again immediately visible from 2.37. The function we are to minimize demands calculating an inverse of an $N \times N$ matrix and calculating its determinant, where N is the cardinality of our data set. Both of these operations yield $\mathcal{O}(N^3)$ computational complexity.

2.3.3 Estimating the hyperparameters

We have already seen that the hyperparameters of the kernel can be learned through the optimization of the marginal likelihood. There are however ways to estimate and understand the hyperparameters that do not rely on the marginal likelihood. One such idea comes from the geometry of stochastic processes. In [9] it is shown that the expected number of upcrossings of a level u on a unit interval of a one dimensional stochastic process generated with a kernel function $k(r)$ is given by:

$$\mathbb{E}[N_{u,1}] = \frac{1}{2\pi}\sqrt{\frac{-k''(0)}{k(0)}}\exp\left(-\frac{u^2}{2k(0)}\right) \tag{2.39}$$

Number of upcrossings of a level u is the number of times a process passes through the value u with a positive slope. This expectation depends on the value u but only through

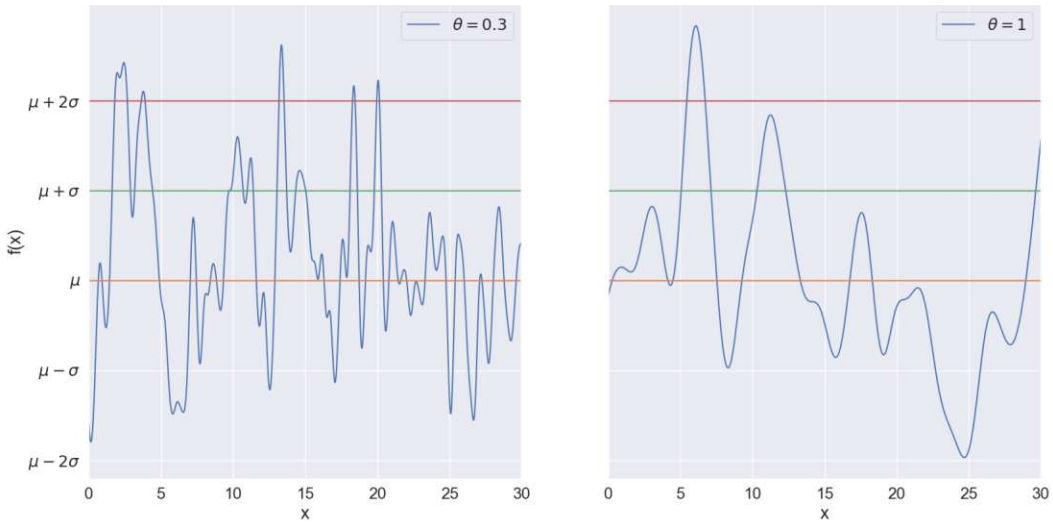


Figure 2.7: Upcrossings count for two sample functions

the u^2 factor in the exponential. From this it is clear that upcrossings are treated the same whether this level value is positive or negative. This makes sense, as a function mirrored over the x-axis is still a valid realisation of a stochastic process defined by the same kernel. The number of upcrossings would be equal to the number of downcrossings, passing through u with a negative slope, of a mirrored function. We conclude that whether we count up- or downcrossings would not make a big difference with regards to estimation of hyperparameters. Actually, if a function passes through u upwards at some point, it must also at some point pass through u downwards. So counting up- or downcrossings on a finite interval of a function realization boils down to an even or odd parity error. If we were to estimate N_u from a single realisation we could count all crossings of a level u and divide it by two, only introducing a 0.5 error in the worst case. For an RBF kernel specifically, the kernel function is twice differentiable, so we can find the exact expression for this expectation:

$$\mathbb{E}[N_{u,1}] = \frac{1}{2\pi\theta} \exp\left(-\frac{u^2}{2}\right) \tag{2.40}$$

Another way to intuitively understand the formula given by 2.39 and 2.40 is to understand the hyperparameters as defining how quickly the function oscillates. If a function is oscillating fast, it will pass through a predefined level frequently, and the amount of oscillations is steered by some kernel hyperparameters. We also notice that as the level u increases in magnitude, the expected number of upcrossings decreases. In Figure 2.7 we can see both these effects taking place. In the left plot the lenght-scale parameter is shorter and the function sample oscillates quickly, generating a larger amount of crossings than the sample on the right plot. Also, for higher levels u the probability of the function reaching this value decreases. From this it is also clear that if we set u too high up

we will probably not see any crossings, and that if we are interested in the parameter estimation from the number of crossings we also have to have some idea of the mean and variance of the underlying Gaussian Process.

In a realistic scenario, we are not able to calculate the expectation of N_u since we only observe one realization of the process. If we approximate that the expectation is equal to the observed number of upcrossings on an interval of length d we get:

$$\begin{aligned} N_{u,d} &= dN_{u,1} \\ N_{u,d} &\approx \frac{d}{2\pi\theta} \exp\left(-\frac{u^2}{2}\right) \\ \theta &\approx \frac{d}{2\pi N_{u,d}} \exp\left(-\frac{u^2}{2}\right) \end{aligned} \tag{2.41}$$

We are now wondering how good is this approximation. To understand this, we sampled 20 functions from a GP prior and counted the number of upcrossings on multiple levels u. In Figure 2.8 we can see the result of this experiment. On the x-axis is the ratio between the length of the sampled functions, denoted by d, and the true length-scale parameter θ. On the y-axis is the Mean Absolute Percentage Error (MAPE), defined as:

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=0}^{n} \frac{|\theta - \hat{\theta}|}{\theta} \tag{2.42}$$

The choice of u has to be sensible regarding the probability of the function reaching this level. We do not want to set u such that the probability of this value is very low, and it should be somewhere in the $\pm$ one standard deviation range. This can be seen in Figure 2.8 as the approximations for the level $\pm 2\sigma$ are much worse than for the other cases. We expect that the longer our sample functions are, the better the approximation will be. In Figure 2.8, we can see also this effect taking place. The longer our sampled functions are, the better the parameter approximation becomes. A comforting fact is that the estimation works well enough for many levels of u so we only have to have a rough idea of the function mean and variance.

When our sampled function is short compared to θ it is a frequent occurence that we would have no crossings on any predetermined level. As the value N_u cannot be zero according to the formula, we can use any arbitrarily small non-zero number. Here we set it to 0.5 in all such cases. Additionally, we notice here that if the sampled function is short compared to the lenght-scale parameter, there is little information about the function to begin with, so we expect the hyperparameter approximation to be quite poor no matter which estimation or optimization method we choose.

The approximations seen in Figure 2.8 have a substantial variance, and might not achieve errors that are small enough to satisfy proper function fitting criteria, but they do serve as a good, if somewhat crude approximation of the hyperparameters. They are in all

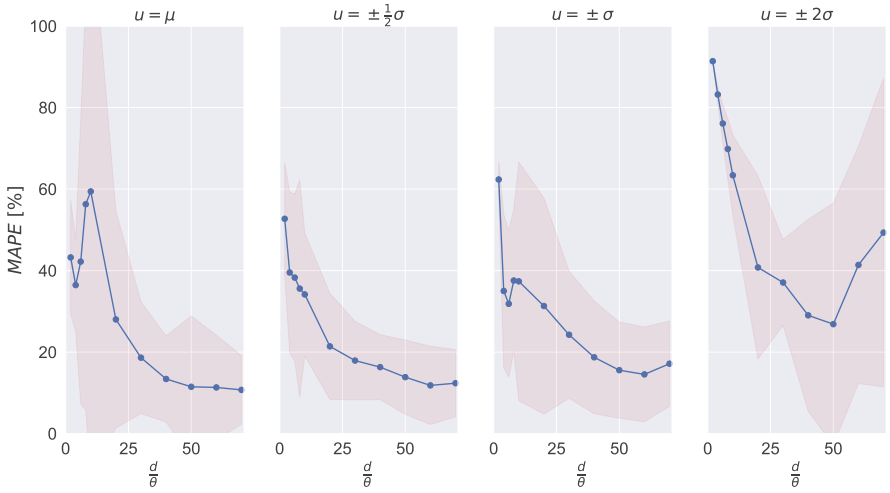


Figure 2.8: Approximation error of the length-scale parameter over different sample lengths

cases in the same order of magnitude as the original kernel length-scale. The importance of estimation of the length-scale will become clearer later in chapter 3.

Here we discussed the method for kernel parameter optimization only for one-dimensional (1D) functions, for higher dimensions we cannot define function upcrossings that easily. There are however results that generalize results shown in this section for N-dimensional Gaussian Random Fields.[10] The equivalent method for higher dimensions is out of scope of this thesis. However, an exploration of these methods is an interesting idea for further research.

As a teaser we would like to show how this parameter estimation could work in a two-dimensional (2D) case we show Figure 2.9. The rows in Figure 2.9 represent samples from a GP prior with kernel parameters specified on the left. The uppermost and lowermost rows have so-called isotropic kernels, for which the length-scale parameter is equal in all dimensions of the input space. In the middle row, we have an example of a so-called anisotropic kernel, with the length-scale equal to 1 in the x-axis direction and equal to 3 in the y-axis dimension. Note how, because of the different correlation distances in the two directions the resulting function is stretched in the y-axis direction compared to the isotropic kernel case.

In Figure 2.9 represented in black are contour lines, i.e. lines composed of points at which the function sample is equal to some value u. These contour lines are the equivalent of function up- and downcrossings of level u in the 1D case. We notice very similar behavior as in the 1D case argued with Figure 2.7. For shorter length-scales, the number and density of these lines are higher. We also notice that for higher levels u the number and density of the contour lines again decreases. This behaviour is equivalent to what we

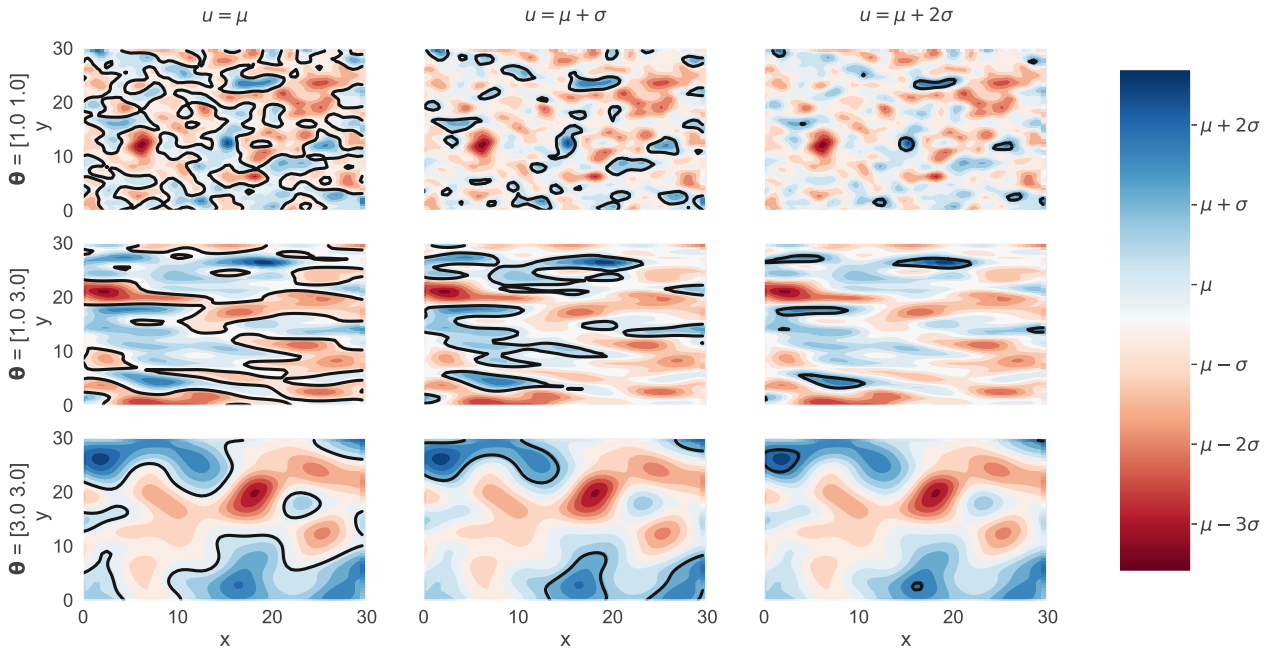


Figure 2.9: Contour lines of samples generated with different length-scale parameters

have observed in the one-dimensional case, indicating that this method is in some way extendable to higher dimensions.

CHAPTER 3

Sparse Gaussian Processes

3.1 Introduction to Sparse Gaussian Process Regression

In the previous chapter, the predictive posterior and the marginal likelihood of the GP model have been derived. The predictive posterior is again:

$$p(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.1}$$

$$\begin{aligned}\boldsymbol{\Sigma} &= \mathbf{K}(\mathbf{X}^*, \mathbf{X}^*) - \mathbf{K}(\mathbf{X}^*, \mathbf{X})(\sigma_\epsilon^2 \mathbf{I}_N + \mathbf{K}_{NN})^{-1}\mathbf{K}(\mathbf{X}, \mathbf{X}^*) = \\ &= \mathbf{K}_{**} - \mathbf{K}_{*N}(\sigma_\epsilon^2 \mathbf{I}_N + \mathbf{K}_{NN})^{-1}\mathbf{K}_{N*}\end{aligned} \tag{3.2}$$

$$\begin{aligned}\boldsymbol{\mu} &= \mathbf{K}(\mathbf{X}^*, \mathbf{X})(\sigma_\epsilon^2 \mathbf{I}_N + \mathbf{K}_{NN})^{-1}\mathbf{y} = \\ &= \mathbf{K}_{*N}(\sigma_\epsilon^2 \mathbf{I}_N + \mathbf{K}_{NN})^{-1}\mathbf{y}\end{aligned} \tag{3.3}$$

This time we use a slightly different notation from 2.35 and 2.34, dropping the posterior in the index and adding the index $\mathbf{K}_{NN}$ to the full covariance matrix to indicate its dimensions. The covariance matrix $\mathbf{K}_{**}$ and vectors $\mathbf{K}_{*N}$, $\mathbf{K}_{*N}$ evaluated at the test locations also have a changed shortened notation in the index. We are however not too concerned with the test set size, so we do not specifically denote it in our notation. With this new notation, we also have the negative log marginal likelihood from 2.37 as:

$$\mathcal{L} = -\log p(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T(\mathbf{K}_{NN} + \sigma_\epsilon^2 \mathbf{I}_N)^{-1}\mathbf{y} + \frac{1}{2}\log\det(\mathbf{K}_{NN} + \sigma_\epsilon^2 \mathbf{I}_N) + \frac{N}{2}\log 2\pi \tag{3.4}$$

As stated in the previous chapter, the main drawback of the GP model stems from these equations. The posterior and the log marginal likelihood needed for hyperparameter optimization demand calculating an inverse of an $N \times N$ matrix and calculating its determinant, where N is the cardinality of our training data set. Both of these operations yield $\mathcal{O}(N^3)$ computational complexity.

Because the computational inefficiency of GPs is such a large hurdle to employing them for any large data sets plenty of research has been done on methods to reduce it. Sparse

Gaussian Process (SGP) methods are a variation of Gaussian Processes that aim to reduce the computational complexity of the model and they are the topic of this chapter. Most SGP methods achieve the reduction of computational complexity by selecting a smaller subset of the data, consisting of M locations, and using them to approximate the full GP model. These methods mostly differ in the strategies for choosing this subset of M points, where M should ideally be much smaller than N. In the current literature on the topic of SGP, this approximating subset has different names. Most commonly they are referred to as inducing inputs, sometimes auxiliary or support inputs. The set of inducing inputs is sometimes called the active set, and the remaining points are the inactive set. Let us introduce this set of M auxiliary inducing points, whose locations are $\mathbf{X}_M$, and corresponding latent function values are ordered in a vector $\mathbf{f}_M$. The locations $\mathbf{X}_M$ do not have to necessarily be either from the training or test locations, for this reason, they are sometimes also called pseudo-inputs. As we will see presently, however, most methods rely on choosing this subset from the training data.

In addition to inducing point methods, there are also methods that approximate the covariance matrix directly. One popular example is the Nyström matrix approximation, which is a low-rank matrix approximation commonly used in kernel methods in machine learning. The Nyström approximation does not systematically change the kernel function, but only replaces occurrences of the covariance matrix $\mathbf{K}_{NN}$ with its approximation $\tilde{\mathbf{K}}_{NN}$. This approximation we will not explore, as it has its issues. As noted in [3], the approximated predicted variance might turn out to be negative because there is no systematic change in the covariance function.

The size of the active set M should be significantly smaller than the size of the complete training set. However, determining an appropriate value for M solely based on the training data remains an unsolved challenge. It is somewhat intuitively clear but will be shown later too, that, in general, the larger we allow M to be the better the resulting approximation will be. As we would like to strike a balance between model complexity and accuracy, the selection of the inducing points set size is a matter that requires careful consideration. There are two approaches to tackling this issue. One approach is approximating or somehow guessing the appropriate M from prior knowledge about the data. If the resulting model trained with this chosen set of inducing inputs lacks sufficient accuracy, M is increased, and the entire procedure is repeated. One might, for example, compute the Mean Squared Error (MSE) of the sparse model and if it is higher than some desired threshold repeat the SGP procedure with a larger M. Alternatively, a greedy selection procedure can be employed, where points are iteratively added to the current set of inducing inputs based on a greedy criterion. At each step, the corresponding log marginal likelihood is computed, and the hyperparameters might be optimized too. Once there is no sufficient change in the log marginal likelihood, the algorithm is stopped. In general, greedy algorithms are more complex as the log marginal likelihood has to be calculated and hyperparameters optimized multiple times. Therefore, we would like to develop some intuition on how large M should be based on prior knowledge to reduce the need for restarting the mentioned procedures.

3.2 Sparse Gaussian Process Regression Mathematical Model

There are two main SGPR methods we will explore. The first group we will call Subset of Data (SD), as in [3] and [11]. The second one we will call the Projected Process (PP) Approximation, as in [3]. In [11] this method is named the Deterministic Training Conditional (DTC) Approximation. For each of these methods, different subset selection criteria could be used.

Other notable methods that we will not explore in this thesis include the Subset of Regressors (SR) method [12] [13], which selects a subset of the training data to use as basis functions, and the Bayesian Committee Machine (BCM) [14], which combines multiple sparse Gaussian process models to improve performance.

3.2.1 Subset of Data

The Subset of Data (SD) method is fairly easy to grasp. Instead of using all of the N data points we have in the training set, we choose only M of them based on some criteria and keep the same GP model. The resulting covariance matrix is now 2.7 evaluated only at the M locations $\mathbf{X}_M$. The resulting GP predictor is:

$$p(\mathbf{f}^*|\mathbf{y}) = \mathcal{N}(\mathbf{f}^*; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{3.5}$$

$$\boldsymbol{\Sigma}_{SD} = \mathbf{K}_{**} - \mathbf{K}_{*M}(\sigma_\epsilon^2 \mathbf{I}_M + \mathbf{K}_{MM})^{-1}\mathbf{K}_{M*} \tag{3.6}$$

$$\boldsymbol{\mu}_{SD} = \mathbf{K}_{*M}(\sigma_\epsilon^2 \mathbf{I}_M + \mathbf{K}_{MM})^{-1}\mathbf{y} \tag{3.7}$$

and the corresponding negative log marginal likelihood is:

$$\begin{aligned} \mathcal{L}_{SD} &= -\log p(\mathbf{y}) = -\log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{K}_{MM} + \sigma_\epsilon^2 \mathbf{I}_M) = \\ &= \frac{1}{2}\mathbf{y}^T(\mathbf{K}_{MM} + \sigma_\epsilon^2 \mathbf{I}_M)^{-1}\mathbf{y} + \frac{1}{2}\log\det(\mathbf{K}_{MM} + \sigma_\epsilon^2 \mathbf{I}_M) + \frac{M}{2}\log 2\pi \end{aligned} \tag{3.8}$$

The resulting $N - M$ data points are simply disregarded, which is clearly wasteful of data. In spite of this, it might be an appropriate solution if we are able to get sufficiently accurate results with just the M data points.

3.2.2 Projected Process Approximation

Another variant of SGPs does not disregard the full data set, but instead uses the inducing inputs as auxiliary random variables. As in 2.30 the posterior GP can be represented by the predictive Gaussian distribution:

$$p(\mathbf{f}^*|\mathbf{y}) = \int_{-\infty}^{\infty} p(\mathbf{f}^*|\mathbf{f})p(\mathbf{f}|\mathbf{y})d\mathbf{f}. \tag{3.9}$$

We now introduce the vector $\mathbf{f}_M$ of latent function values evaluated at the $\mathbf{X}_M$ inducing input locations. We are assuming these latent function values are drawn from the same

joint Gaussian distribution as the training function values $\mathbf{f}$. In this case, we get an expanded predictive posterior as:

$$p(\mathbf{f}^*|\mathbf{y}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(\mathbf{f}^*|\mathbf{f}_M, \mathbf{f})p(\mathbf{f}|\mathbf{f}_M, \mathbf{y})p(\mathbf{f}_M|\mathbf{y})d\mathbf{f}d\mathbf{f}_M \tag{3.10}$$

Ideally, $\mathbf{f}_M$ would be a sufficient statistic of $\mathbf{f}$. In statistics, a sufficient statistic is a function of the observed data that captures all the relevant information about an unknown parameter. It summarizes the data in a way that retains all the information necessary for making inferences about the parameter of interest. More formally, a statistic $T(\mathbf{X})$ is considered sufficient for a parameter $\boldsymbol{\alpha}$ if the conditional distribution of the data $\mathbf{X}$, given $T(\mathbf{X})$, does not depend on $\boldsymbol{\alpha}$, formally written as:

$$p(\mathbf{X}|T(\mathbf{X}), \boldsymbol{\alpha}) = p(\mathbf{X}|T(\mathbf{X})) \tag{3.11}$$

In other words, once the sufficient statistic is known, any further information in the data regarding the parameter does not provide additional insights. The concept of sufficiency is important in statistical inference because it allows for data reduction and simplification without loss of information.

In our case, if $\mathbf{f}_M$ would be a sufficient statistic of $\mathbf{f}$, that would mean that:

$$p(\mathbf{f}^*|\mathbf{f}_M, \mathbf{f}) = p(\mathbf{f}^*|\mathbf{f}_M) \tag{3.12}$$

Which we can easily determine, by referencing 3.11, and drawing parallels to our current problem. Another consequence would be that:

$$p(\mathbf{f}|\mathbf{f}_M, \mathbf{y}) = p(\mathbf{f}|\mathbf{f}_M) \tag{3.13}$$

which is not immediately visible, so we will derive it. Let us start with the following:

$$\begin{aligned} p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{y}) &= \int_{-\infty}^{\infty} p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{y}, \mathbf{f})d\mathbf{f} = \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{f}^*, \mathbf{f}_M, \mathbf{f})p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{f})d\mathbf{f} = \\ &= \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{f})p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{f})d\mathbf{f} \end{aligned} \tag{3.14}$$

where we use the fact that $\mathbf{y}$ is just a noisy version of $\mathbf{f}$, so $p(\mathbf{y}|\mathbf{f}^*, \mathbf{f}_m, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})$. We can additionally write the resulting integral as:

$$\begin{aligned} p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{y}) &= \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{f})p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{f})d\mathbf{f} = \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{f})p(\mathbf{f}^*|\mathbf{f}_M, \mathbf{f})p(\mathbf{f}_M, \mathbf{f})d\mathbf{f} = \\ &= \int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{f})p(\mathbf{f}^*|\mathbf{f}_M)p(\mathbf{f}_M, \mathbf{f})d\mathbf{f} = p(\mathbf{f}^*|\mathbf{f}_M)\int_{-\infty}^{\infty} p(\mathbf{y}|\mathbf{f}_M, \mathbf{f})p(\mathbf{f}_M, \mathbf{f})d\mathbf{f} = \\ &= p(\mathbf{f}^*|\mathbf{f}_M)\int_{-\infty}^{\infty} p(\mathbf{y}, \mathbf{f}_M, \mathbf{f})d\mathbf{f} = p(\mathbf{f}^*|\mathbf{f}_M)p(\mathbf{y}, \mathbf{f}_M) \end{aligned} \tag{3.15}$$

where we used 3.12. Using 3.15 and Bayes' rule we can now write:

$$p(\mathbf{f}^*|\mathbf{f}_M, \mathbf{y}) = \frac{p(\mathbf{f}^*, \mathbf{f}_M, \mathbf{y})}{p(\mathbf{f}_M, \mathbf{y})} = p(\mathbf{f}^*|\mathbf{f}_M) \tag{3.16}$$

Meaning that, given $\mathbf{f}_M$, any vector $\mathbf{f}^*$ is conditionally independent of the observed data $\mathbf{y}$. Since $\mathbf{f}^*$ are function values at any set of test locations, the same holds for the latent function values at our training inputs, $\mathbf{f}$, which gives precisely 3.13.

As in practice, $\mathbf{f}_M$ is not a sufficient statistic of $\mathbf{f}$, by using these assumptions we will get an approximate posterior, that we will denote $q(\mathbf{f}^*)$. Using 3.12 and 3.13 in 3.10 we get:

$$\begin{aligned} q(\mathbf{f}^*) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(\mathbf{f}^*|\mathbf{f}_M)p(\mathbf{f}|\mathbf{f}_M)p(\mathbf{f}_M|\mathbf{y})d\mathbf{f}d\mathbf{f}_M = \int_{-\infty}^{\infty} p(\mathbf{f}^*|\mathbf{f}_M)p(\mathbf{f}_M|\mathbf{y})d\mathbf{f}_M = \\ &= \int_{-\infty}^{\infty} p(\mathbf{f}^*|\mathbf{f}_M)\phi(\mathbf{f}_M)d\mathbf{f}_M = \int_{-\infty}^{\infty} q(\mathbf{f}^*, \mathbf{f}_M)d\mathbf{f}_M \end{aligned} \tag{3.17}$$

Where in the third step we replace $p(\mathbf{f}_M|\mathbf{y})$ by $\phi(\mathbf{f}_M)$. This is another approximation that we will use and in general, understand that $p(\mathbf{f}_M|\mathbf{y}) \neq \phi(\mathbf{f}_M)$. We will instead suppose that $\phi(\mathbf{f}_M)$ is a variational Gaussian distribution $\mathcal{N}(\mathbf{f}_M; \boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M)$. If it is so defined, the approximate predictive posterior becomes using standard Gaussian results, as in [15]:

$$q(\mathbf{f}^*) = \mathcal{N}(\mathbf{f}*; \boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q) \tag{3.18}$$

$$\boldsymbol{\Sigma}_q = \mathbf{K}_{**} - \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\mathbf{K}_{M*} + \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\boldsymbol{\Sigma}_M\mathbf{K}_{MM}^{-1}\mathbf{K}_{M*} \tag{3.19}$$

$$\boldsymbol{\mu}_q = \mathbf{K}_{*M}\mathbf{K}_{MM}^{-1}\boldsymbol{\mu}_M \tag{3.20}$$

Clearly, the quality of this approximation will depend on the choice of the locations $\mathbf{X}_M$, but also the choice of the mean $\boldsymbol{\mu}_M$ and covariance matrix $\boldsymbol{\Sigma}_M$ at those locations.

In [16] and [17], the so-called Projected Process Approximation is derived. This approximation chooses the variational distribution $\phi(\mathbf{f}_M)$ such that the Kullback-Leibler (KL) divergence, $\mathrm{KL}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y}))$, between the approximate distribution $q(\mathbf{f})$ and the true posterior $p(\mathbf{f}|\mathbf{y})$ is minimized. Using this criterion, the resulting distribution is:

$$\phi(\mathbf{f}_M) = \mathcal{N}(\mathbf{f}^*; \boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M) \tag{3.21}$$

$$\boldsymbol{\Sigma}_M = \mathbf{K}_{MM}(\mathbf{K}_{MM} + \frac{1}{\sigma_\epsilon^2}\mathbf{K}_{MN}\mathbf{K}_{NM})^{-1}\mathbf{K}_{MM} \tag{3.22}$$

$$\boldsymbol{\mu} = \frac{1}{\sigma_\epsilon^2}\mathbf{K}_{MM}(\mathbf{K}_{MM} + \frac{1}{\sigma_\epsilon^2}\mathbf{K}_{MN}\mathbf{K}_{NM})^{-1}\mathbf{K}_{MN}\mathbf{y} \tag{3.23}$$

All of the covariance matrices are defined the same as in 2, with the index defining at which locations, M for the inducing input locations, N for training locations, and $*$ for test locations. The resulting predictive distribution 3.18 is the same in the mean as the Subset of Regressors (SR) one, but they differ in the predictive variance. For the PP model, as the SR model, it takes $\mathcal{O}(M^2N)$ time to carry out the necessary matrix computations. After that, the prediction of the mean for a new test point takes $\mathcal{O}(M)$ time and the predictive variance takes $\mathcal{O}(M^2N)$ time.

The resulting negative log marginal likelihood is:

$$\begin{aligned}\mathcal{L}_{PP} &= -\log p(\mathbf{y}) = -\log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN} + \sigma_\epsilon^2 \mathbf{I}_N) = \\ &= -\log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{Q}_{NN} + \sigma_\epsilon^2 \mathbf{I}_N) = \\ &= \frac{1}{2}\mathbf{y}^T(\mathbf{Q}_{NN} + \sigma_\epsilon^2 \mathbf{I}_N)^{-1}\mathbf{y} + \frac{1}{2}\log\det(\mathbf{Q}_{NN} + \sigma_\epsilon^2 \mathbf{I}_N) + \frac{N}{2}\log 2\pi \end{aligned} \tag{3.24}$$

where $\mathbf{Q}_{NN} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}$ is the new approximated covariance matrix. This is incidentally the form of the Nyström approximation too.

The PP model, however, does not specify how to choose the locations $\mathbf{X}_M$ of the inducing inputs, just how to find the optimal distribution $\phi(\mathbf{f}_M)$ for the given $\mathbf{X}_M$. Both authors in [16] and in [17] suggest some possible greedy algorithms for choosing the active set $\mathbf{X}_M$. The one in [17], is equivalent to the greedy active set algorithm described in [18] as part of a so-called Informative Vector Machine (IVM). This particular greedy algorithm we will discuss in the next section, along with some other potential methods for active set selection.

If we compare 3.24 and 2.37 we see that when the PP Approximation is applied the prior of the GP model is modified. This implies that the inducing inputs $\mathbf{X}_M$ play the role of extra kernel hyperparameters, similar to the already mentioned $\boldsymbol{\theta}$. These extended hyperparameters now fully parametrize the approximate covariance matrix $\mathbf{Q}_{NN}$.

3.3 Choosing the inducing points

As we have seen in the previous section, there are two main ways to build an Sparse Gaussian Process Regression model. They both rely on choosing a smaller subset of the observed data, the active set, consisting of M data points. They do not specify, however, how the active set is to be chosen from the observed data. The methods for building the active set will be explored in this section. As a short overview, we present some of the most popular solutions that we will explore in the following:

- Random sampling: In this method, a random subset of the data is selected as inducing points.
- Clustering, specifically K-means clustering: This method involves clustering the data into (K) groups and using the cluster classification for selecting representative points from each cluster as inducing points. Usually, a single representative point from each cluster is chosen for the active set.
- Deterministic subset selection: This method involves selecting a subset of the data based on some deterministic criteria. For example, the Maximum Entropy Sampling method selects inducing points that have the maximum entropy. Another proposed deterministic subset selection method is the Informative Vector Machine.

- Variational learning: This method involves approximating the posterior distribution of the inducing points using a variational distribution. The inducing points are then treated as variational parameters to be optimized. The inducing points and the approximate predictive distribution are selected by minimizing the Kullback-Leibler (KL) divergence between the approximate posterior and the exact one.

3.3.1 Random subsampling

Random subsampling, also known as random subset selection, is a technique that randomly selects a smaller subset of the training data by assigning equal probabilities to each individual point. Although it may appear simplistic, random subsampling is widely used to address the challenges of large datasets in various applications. For instance, in the case of imbalanced datasets, random subsampling can be employed to create balanced subsets by randomly undersampling the majority class mitigating the class imbalance problem during model training.

In our specific use case, for the selection of inducing inputs in SGPR, random subsampling serves as the simplest yet effective solution. By using random subsampling as a benchmark, we can evaluate the performance of more advanced subset selection algorithms. If a more complex model fails to significantly outperform random subsampling, it raises the question of whether the additional computational effort required by the sophisticated approach is justified. In general, when comparing different solutions to a problem, it is advisable to include simple approaches, such as random subsampling here, to gain insights into the relative performance of more intricate methods. This allows for a comprehensive assessment and comparison of the effectiveness of proposed solutions in addressing the problem at hand.

3.3.2 Clustering

Clustering is a wide area of machine learning and there are many proposed algorithms available in literature.[19] It is a type of unsupervised learning where similar data points are grouped together based on their characteristics or features. The goal of clustering is to find natural groupings within the data without any prior knowledge of the labels or categories.

Clustering can be used for SGPR to reduce the training data set and identify a representative subset of the input data points, for example by choosing a single exemplary point from each cluster, to use in the regression model. One thing to note is that, as the primary goal of SGP methods is the reduction of computational complexity, the clustering algorithm used should also be relatively fast and simple. An extensive study of different clustering algorithms for this purpose has been performed in [20]. In this work, the author recommends the use of K-means clustering, as it is a simple and computationally cheap algorithm shown to outperform other clustering methods in was compared to on real-world measurements.

3.3.2.1 K-means clustering

K-means clustering, sometimes called Lloyd's algorithm [21], is one of the most commonly used clustering algorithms. It works reasonably well in many scenarios, is numerically stable, and due to its simple structure it is fast compared to most other clustering algorithms. One of the downsides is that the algorithm's success depends highly on its initialization. To combat this issue, a good initialization strategy is necessary, such is for example K-means++. This initialization strategy will be explained in Section 3.3.2.2. Another characteristic of K-means is that the number of clusters, K, has to be known or somehow predetermined. This is usually denoted as one of its downsides but in SGP that is not necessarily the case. In SGP our goal is to reduce the original data set, not uncover the underlying cluster structure of the training set. So in a way, being able to control exactly how many resulting cluster centers we get is an upside for this specific application. With the use of K-means clustering, we can easily set $K = M$, where M is the desired number of inducing inputs.

The goal of K-means is to partition N input data samples into K clusters based on some distance measure $d(\cdot, \cdot)$. Usually, the choice for the distance measure is the squared L_2 norm, i.e. Euclidian distance $d(\mathbf{x}, \mathbf{x}') = ||\mathbf{x} - \mathbf{x}'||_2^2$. The clusters are defined by their centroid $\boldsymbol{\mu}_k$ and composed of all points whose distance to the corresponding centroid is smaller than to any other one of the remaining $K - 1$ centroids. Let us denote the set of all points assigned to cluster k as $\mathcal{C}_k$, and the cardinality of this set, i.e. number of elements in it, as $|\mathcal{C}_k|$.

The algorithm starts with an initial set of the K centroids and assigns each of the data points to one of them based on the distance measure. In each following iteration the cluster centroids are updated according to:

$$\boldsymbol{\mu}_k = \frac{\sum_{\mathbf{x} \in \mathcal{C}_k} \mathbf{x}}{|\mathcal{C}_k|} \tag{3.25}$$

After the centroid update, every data point is reassigned to the clusters based on the new centroid positions. Data points are assigned to the cluster to which centroid they are the closest.

The algorithm iteratively updates the centroids and assigns each data point to the nearest cluster centroid as described until it converges. The convergence is usually determined by the objective function J defined by:

$$J = \sum_{k=1}^{K} \sum_{\mathbf{x} \in \mathcal{C}_k} d(\mathbf{x}, \boldsymbol{\mu}_k) \tag{3.26}$$

sometimes also called the distortion measure. [22] As the goal is to minimize this objective function, once it's update becomes smaller than a predetermined threshold ϵ, i.e. $|J_{new} - J_{old}| < \epsilon$, between two iterations the algorithm stops.

3.3.2.2 K-means initialization: K-means++

As the K-means algorithm's result and needed number of iterations depend greatly on the initial choice of the cluster centers, a scheme was proposed in [23] to somewhat mitigate this issue. K-means++ is an extension to the original algorithm that provides another iterative solution for initialization. Given the N data points, we wish to choose K for our initial cluster centroids. We start by choosing the first one, $\boldsymbol{\mu}_1$, by drawing a random sample uniformly from the data set. For the remaining initial cluster centers, we assign a probability P_n to every remaining data point $\mathbf{x}_n$ according to 3.27

$$P_n = \frac{d_S(\mathbf{x}_n)}{\sum_{n'=1}^{N} d_S(\mathbf{x}_{n'})} \tag{3.27}$$

where $d_S(\mathbf{x}_n)$ is the distance of $\mathbf{x}_n$ to the nearest already chosen center up to the current iteration i, i.e.

$$d_S(\mathbf{x}_n) = \arg\min_i ||\mathbf{x}_n - \boldsymbol{\mu}_i||_2^2 \tag{3.28}$$

The next centroid is randomly chosen from the remaining data points based on their assigned probabilities P_n. The computation of probabilities P_n and the selection of the next centroid $\boldsymbol{\mu}_i$ is repeated until all K initial centroids are chosen.

3.3.3 Deterministic subset selection

Deterministic subset selection is a family of techniques used in SGP to, again, identify a representative subset of the input data points to use as inducing inputs. The difference compared to Random Subsampling is that in this family of algorithms, we define some deterministic criteria for choosing the points in our inducing set. The defined criteria we will denote by Δ_j, with j being the index of the point at which the criteria is evaluated, if applicable.

3.3.3.1 Maximum Entropy Sampling

One deterministic subset selection method that could be used in SGP regression is Maximum Entropy Sampling (MES), first described in [24]. The MES problem arose in spatial statistics, as has the GPR model itself. Spatial statistics is a branch of statistics that focuses on analyzing and interpreting data with spatial or geographic attributes. It is widely used in fields such as environmental science, urban planning, meteorology, and geology, to understand spatial patterns, relationships, and processes. Values of a spatial variable are observed at different locations and at possibly different times, giving in total N observations. These samples are modeled as correlated random variables. The MES problem is a problem of selecting a subset of prespecified size D from the set of these correlated random variables, such that the chosen subset is the most informative, i.e. it has maximal possible entropy. In other words, it selects the subset that is the most uncertain and could potentially provide the most useful information to the model. If the set of correlated random variables are jointly Gaussian, the MES problem is

equivalent to choosing the subset such that the determinant of the covariance matrix of the subset is maximized. This is directly clear from the general form of the Entropy of Multivariate Gaussian Distribution, as in lemma 3.3.1. As we assume that the dimension D is prespecified, the term $\frac{D}{2} + \frac{D}{2}\log 2\pi$ is a constant. The only term that we are able to change is the determinant of the covariance matrix.

Lemma 3.3.1 (**Entropy of a Multivariate Gaussian Distribution**). *The entropy of a multivariate Gaussian distribution* $\mathcal{N}(\mathbf{x}_{D\times 1}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ *is* $\frac{1}{2}\log\det\boldsymbol{\Sigma} + \frac{D}{2} + \frac{D}{2}\log 2\pi$.

$$
\begin{aligned}
H(\mathbf{x}) &= -\mathbb{E}[\log(\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}))] = \\
&= -\mathbb{E}\left[\frac{1}{\sqrt{(2\pi)^D \det\boldsymbol{\Sigma}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\right] = \\
&= \frac{D}{2}\log 2\pi + \frac{1}{2}\log\det\boldsymbol{\Sigma} + \frac{1}{2}\mathbb{E}\left[(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] = \\
&= \frac{D}{2}\log 2\pi + \frac{1}{2}\log\det\boldsymbol{\Sigma} + \frac{D}{2}
\end{aligned}
$$

Where we used the following equality:

$$
\begin{aligned}
\mathbb{E}\left[(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right] &= \mathbb{E}\left[\mathrm{Tr}\left((\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)\right] = \\
&= \mathbb{E}\left[\mathrm{Tr}\left((\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T(\mathbf{x}-\boldsymbol{\mu})\right)\right] = \\
&= \mathrm{Tr}\left(\mathbb{E}\left[\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})^T(\mathbf{x}-\boldsymbol{\mu})\right]\right) = \\
&= \mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\mathbb{E}\left[(\mathbf{x}-\boldsymbol{\mu})^T(\mathbf{x}-\boldsymbol{\mu})\right]\right) = \\
&= \mathrm{Tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\right) = \\
&= \mathrm{Tr}(\mathbf{I}) = D
\end{aligned}
\tag{3.29}
$$

In [25] it is shown that this problem of choosing a subset of jointly Gaussian random variables, such that the determinant of the subset covariance matrix is maximized is NP-hard. In computer science, NP-hard refers to a class of computational problems that are known to be computationally difficult. The NP stands for Non-deterministic Polynomial-Time, which means there is no known polynomial-time algorithm for solutions of NP-hard problems. They generally require exponential time to find an exact solution. In other words, as the size of the problem increases, the time required to solve it grows exponentially. Many important optimization problems in various domains have been proven to be NP-hard, such as the traveling salesman problem, the knapsack problem, and the graph coloring problem.

While there might not be efficient algorithms for solving NP-hard problems exactly, approximate algorithms or heuristics can often provide reasonably good solutions within

a reasonable amount of time. We will not delve into such approximate algorithms at this time. The NP-hardness of the MES problem indicates that even approximate solutions will require ample computational effort. As already discussed in 3.3.2, the primary goal of SGP is the reduction of computational complexity, so the active set selection method should also be relatively simple.

To evaluate whether MES as a strategy offers any benefit to other active set selection algorithms, we will instead use an alternative greedy algorithm. To select the subset of data points for SGPR, we will use a greedy algorithm that selects one data point at a time based on its expected information gain and adds it to the active set. The algorithm greedily selects the data point with the highest expected information gain based on the sparse predictor achieved with the current active set. This greedy MES method works by first fitting a Projected Process GPR using the current active set. The model's uncertainty is then quantified using the predictive variance, which measures the model's confidence in its predictions. The point that is then added to the active set is the one at which the predictive variance is the highest. The process is repeated until the desired number of data points has been selected.

The greedy MES selects data points with the highest predictive variance, as these data points are the most informative and can reduce the model's uncertainty the most. This is immediately visible from 3.3.2, which we also easily get by setting $D = 1$ in 3.3.1. The entropy of a single Gaussian random variable is a strictly increasing function of its variance. Therefore, the Gaussian random variable with the highest entropy is the one where the variance, or uncertainty, is the highest. In short, the inclusion criteria is:

$$\Delta_j = \frac{1}{2}\log(2\pi\sigma^2) \tag{3.30}$$

The variance will be highest in areas that are sparsely sampled up to that point in the greedy algorithm. We can conclude this in a similar manner that was already argued in 2.3.2 regarding Figure 2.6.

Lemma 3.3.2 (**Entropy of a Gaussian Random Variable**). *The entropy of a Gaussian random variable X with mean μ and variance σ^2 is $\frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2}$.*

$$\begin{aligned} H(x) &= -\mathbb{E}[\log(\mathcal{N}(x;\mu,\sigma^2))] = \\ &= -\mathbb{E}\left[\log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)\right)\right] = \\ &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\mathbb{E}[(x-\mu)^2] = \\ &= \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sigma^2 = \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2} \end{aligned}$$

The units of entropy depend on the logarithm's base. If we define it with a logarithm with base 2 then the units are called bits. If the logarithm is the natural logarithm, with e as the base, than the units are called nats, or natural units od information.

3.3.3.2 Informative Vector Machine

The MES sampling as presented in the previous section is quite similar to another proposed sparse method, the so-called Informative Vector Machine (IVM) proposed in [18]. The author suggests to greedily add points to the active set such that the inclusion of a location maximizes the differential entropy score:

$$\Delta_j = H(p^{inactive}(f_j)) - H(p^{active}(f_j)) \tag{3.31}$$

where $H(p^{inactive}(f_j))$ is the entropy at the location j and $H(p^{active}(f_j))$ is the entropy at this location after the training location j is included in the active set. As we will now show, this approach is equivalent to MES, or equivalent to choosing a location with maximal variance.

First, we define $\mathbf{y}_I$ as the observations at locations $\mathbf{X}_I$, which is the current active set. Similarly, y_j is the observation and f_j the latent function value at the candidate location X_j that we are evaluating for inclusion into the active set. Let's denote the posterior variance at location j before inclusion to the active set as $\sigma_{j,inactive}$. As we know that:

$$\begin{aligned} p(f_j|\mathbf{y_I}, y_j) &= \frac{p(f_j, \mathbf{y_I}, y_j)}{p(\mathbf{y_I}, y_j)} = \frac{p(y_j|f_j, \mathbf{y_I})p(f_j|\mathbf{y_I})p(\mathbf{y_I})}{p(\mathbf{y_I}, y_j)} \\ &= \underbrace{\frac{p(\mathbf{y_I})}{p(\mathbf{y_I}, y_j)}}_{\substack{\text{normalization} \\ \text{constant}}} p(y_j|f_j, \mathbf{y_I})p(f_j|\mathbf{y_I}) \\ &\propto p(y_j|f_j)p(f_j|\mathbf{y_I}) = \mathcal{N}(y_j|f_j, \sigma_\epsilon^2)p(f_j|\mathbf{y_I}) \end{aligned} \tag{3.32}$$

where we used the chain rule add the fact that y_j and $\mathbf{y_I}$ are statistically independent given f_j, meaning $p(y_j|f_j, \mathbf{y_I}) = p(y_j|f_j)$. Now we can find the solution for the posterior variance once the location j has been added to the active set as:

$$\frac{1}{\sigma^2_{j,active}} = \frac{1}{\sigma^2_{j,inactive}} + \frac{1}{\sigma^2_\epsilon} \tag{3.33}$$

Using the result given by lemma 3.3.2 we finally get for the inclusion criteria:

$$\Delta_j = \frac{1}{2}\log(1 + \frac{\sigma^2_{j,inactive}}{\sigma^2}) \tag{3.34}$$

Again, this inclusion criterion is a strictly increasing function of $\sigma^2_{j,inactive}$ so it is equivalent to choosing the location with maximum variance out of the inactive set, or equivalent to the already proposed MES algorithm.

3.3.4 Variational learning of inducing inputs

We have so far explored active set selection strategies that rely on choosing a subset of the given training locations. If we would, however, wish to introduce continuous

optimization over all possible inducing input locations, we would have to stray away from the previous selection strategies. One approach that allows for continuous optimization of inducing inputs over the whole area of interest, not just on the observed locations, is presented in [15]. The author does propose an approximate greedy algorithm too, recognizing that continuous optimization might be too complex of a solution for many applications. If we are dealing with high-dimensional spaces, in cases where the active set cardinality M is large, gradient-based continuous optimization might be challenging. Also, the optimization might suffer from multiple local minima, in which case restarting the procedure with different initial inducing locations is necessary. The rigorous selection procedure of pseudo-inputs proposed in this work is in spite of its drawbacks an interesting method. It can give us an insight into the best possible performance of a sparse model with a prespecified size of the active set M.

The most important difference between the method proposed in [15], and others available presented so far, is that the inducing inputs are now defined as variational parameters. They can therefore be optimized by minimizing the Kullback-Leibler (KL) divergence between the variational distribution and the exact posterior. In Section 3.2.2 a similar idea was proposed. To recapitulate, the Projected Process Approximation chooses the variational distribution $\phi(\mathbf{f}_M)$ such that the KL divergence, $\mathrm{KL}(q(\mathbf{f})||p(\mathbf{f}|\mathbf{y}))$, between the approximate distribution $q(\mathbf{f})$ and the true posterior $p(\mathbf{f}|\mathbf{y})$ is minimized. In the Section 3.2.2 we already argued that in the PP method, the inducing inputs play the role of extra hyperparameters. As such, they can also be treated as model hyperparameters to be optimized via the log marginal likelihood procedure. This procedure was described in 2.3.2 for the kernel hyperparameters $\boldsymbol{\theta}$. However, because the prior of the model changes in the PP method, continuous optimization over $\mathbf{X}_M$ of the log marginal likelihood might not reliably approximate the full original GP model. If we turn again to Section 3.2.2, we will see that this optimization over $\mathbf{X}_M$ does not systematically reduce the distance between the modified and the true GP model. Instead, it specifies the distribution that minimizes the mentioned KL divergence for a given set of inducing inputs.

Furthermore, the inclusion of M additional model hyperparameters might lead to overfitting. Overfitting is a common problem in machine learning. It occurs when a model performs exceptionally well on the training data but fails to generalize well to unseen test data. In other words, the model becomes too closely fitted to the training data and loses its ability to make accurate predictions on new, unseen examples. Introducing too many hyperparameters in a regression model can lead to overfitting. In general, when a model has a large number of hyperparameters, it becomes very flexible and can capture even the smallest details and noise in the training data. As a result, the model ends up memorizing the training examples instead of learning the underlying patterns and relationships. With an excessive number of parameters, the model can fit the noise or outliers in the training data, which may not be representative of the true underlying latent process. This results in an overly complex model that fails to generalize well to new data. Essentially, the model becomes too specific to the training set, losing its ability to capture the broader patterns and trends.

To combat the issues described above, instead of modifying the GP prior and maximizing the log marginal likelihood of the modified model as proposed in PP, the variational learning method minimizes the KL divergence between the exact posterior $p(\mathbf{f}, \mathbf{f}_M|\mathbf{y})$ and a variational approximation $q(\mathbf{f}, \mathbf{f}_M)$, given by:

$$\mathrm{KL}(q(\mathbf{f}, \mathbf{f}_M)||p(\mathbf{f}, \mathbf{f}_M|\mathbf{y})) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} q(\mathbf{f}, \mathbf{f}_M) \log \frac{q(\mathbf{f}, \mathbf{f}_M)}{p(\mathbf{f}, \mathbf{f}_M|\mathbf{y})} d\mathbf{f} d\mathbf{f}_M \tag{3.35}$$

The inducing inputs $\mathbf{X}_M$ become variational parameters that are continuously optimized so as to minimize 3.35. The probability distribution $q(\mathbf{f}, \mathbf{f}_M)$ is not a completely free variational distribution. We assume that the latent function values at the inducing inputs $\mathbf{f}_M$ are a sufficient statistic of $\mathbf{f}$. In Section 3.2.2 we have argued this point in some detail. Among other things, this implies that the exact posterior factorizes as:

$$p(\mathbf{f}, \mathbf{f}_M|\mathbf{y}) = p(\mathbf{f}|\mathbf{f}_M)p(\mathbf{f}_M|\mathbf{y}) \tag{3.36}$$

where we used 3.13. The variational distribution should behave in the same way and factorize as:

$$q(\mathbf{f}, \mathbf{f}_M) = p(\mathbf{f}|\mathbf{f}_M)\phi(\mathbf{f}_M) \tag{3.37}$$

where $p(\mathbf{f}|\mathbf{f}_M)$ is the conditional GP prior. The distribution $\phi(\mathbf{f}_M)$ is now a true unconstrained variational distribution. Take notice that this result is the same as what we already saw in the PP Approximation, specifically in 3.17.

The minimization of the KL divergence 3.35 is equivalent to the maximization of the variational lower bound on the true log marginal likelihood. In variational Bayesian methods, this substitution of minimizing the KL divergence with maximizing the so-called Evidence Lower Bound (ELBO), or the variational lower bound, is commonly used. [26] In our case, we will denote the ELBO as F, to comply with the notation in [15]. We can now derive the ELBO, or the true log marginal likelihood lower bound as:

$$\begin{aligned}
-\mathcal{L}_{\text{var}} = \log p(\mathbf{y}) \geq F(\mathbf{X}_M, \phi(\mathbf{X}_M)) &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} q(\mathbf{f}, \mathbf{f}_M) \log \frac{p(\mathbf{f}, \mathbf{f}_M, \mathbf{y})}{q(\mathbf{f}, \mathbf{f}_M)} d\mathbf{f} d\mathbf{f}_M = \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} q(\mathbf{f}, \mathbf{f}_M) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{f}_M)p(\mathbf{f}_M)}{q(\mathbf{f}, \mathbf{f}_M)} d\mathbf{f} d\mathbf{f}_M = \\
&= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} p(\mathbf{f}|\mathbf{f}_M)\phi(\mathbf{f}_M) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{f}_M)p(\mathbf{f}_M)}{p(\mathbf{f}|\mathbf{f}_M)\phi(\mathbf{f}_M)} d\mathbf{f} d\mathbf{f}_M = \\
&= \int_{-\infty}^{\infty} \phi(\mathbf{f}_M) \left[\underbrace{\int_{-\infty}^{\infty} p(\mathbf{f}|\mathbf{f}_M) \log p(\mathbf{y}|\mathbf{f})) d\mathbf{f}}_{\log G(\mathbf{f}_M, \mathbf{y})} + \log \frac{p(\mathbf{f}_M)}{\phi(\mathbf{f}_M)} \right] d\mathbf{f}_M = \\
&= \int_{-\infty}^{\infty} \phi(\mathbf{f}_M) \log \frac{G(\mathbf{f}_M, \mathbf{y})p(\mathbf{f}_M)}{\phi(\mathbf{f}_M)} d\mathbf{f}_M
\end{aligned} \tag{3.38}$$

with a helper function introduced as:

$$\log G(\mathbf{f}_M, \mathbf{y}) = \log \mathcal{N}(\mathbf{y}; \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{f}_M, \sigma_\epsilon^2 \mathbf{I}_N) - \frac{1}{2\sigma_\epsilon^2}\operatorname{Tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN}) \tag{3.39}$$

and $\mathbf{Q}_{NN} = \mathbf{K}_{NM}\mathbf{K}_{MM}^{-1}\mathbf{K}_{MN}$ as has already been abbreviated.

To find the optimal distribution $\phi_{\text{opt}}(\mathbf{f}_M)$ we would have to maximize the lower bound given by 3.38. However, by using Jensen's inequality we can conveniently avoid the dependency on $\phi(\mathbf{f}_M)$ altogether. Jensen's inequality states that, for any random variable Y and convex function g, we have:

$$\mathbb{E}[g(Y)] \geq g(\mathbb{E}[Y]) \tag{3.40}$$

Since the logarithm is a convex function, we can move it outside of the integral using Jensen's inequality as:

$$\begin{aligned} F(\mathbf{X}_M) &= \int_{-\infty}^{\infty} \phi(\mathbf{f}_M) \log \frac{G(\mathbf{f}_M, \mathbf{y}) p(\mathbf{f}_M)}{\phi(\mathbf{f}_M)} d\mathbf{f}_M \geq \\ &\geq \log \int_{-\infty}^{\infty} \phi(\mathbf{f}_M) \frac{G(\mathbf{f}_M, \mathbf{y}) p(\mathbf{f}_M)}{\phi(\mathbf{f}_M)} d\mathbf{f}_M = \\ &= \log \int_{-\infty}^{\infty} G(\mathbf{f}_M, \mathbf{y}) p(\mathbf{f}_M) d\mathbf{f}_M = \\ &= \log \mathcal{N}(\mathbf{y}; \mathbf{0}, \mathbf{Q}_{NN} + \sigma_\epsilon^2 \mathbf{I}_N) - \frac{1}{2\sigma_\epsilon^2}\operatorname{Tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN}) \end{aligned} \tag{3.41}$$

As we can see, this lower bound can be calculated without specifying the exact optimal variational distribution. We do, however, need this distribution for prediction, which we can find by optimizing 3.38. Unsurprisingly, the result of this optimization in [15] and [27], gives the same predictive distribution as in the Projected Process (PP) Approximation. This is unsurprising because both of these methods rely on the same assumptions about the statistics of the inducing inputs. If both models were used with the same active set they will and should result in the same predictive distribution.

The main difference lies exactly in the lower bound expression 3.41, which in the variational method has an additional trace term. It is precisely the sum of the log marginal likelihood from the PP plus this additional term. This variational approach, in conclusion, will try to maximize the log marginal likelihood of the Projected Process while also minimizing the trace term. The trace term corresponds to the error of predicting the latent function values at training locations $\mathbf{f}$ from the latent function variables at only the inducing locations $\mathbf{f}_M$. If we disregard the PP log marginal likelihood term, by optimizing over the inducing inputs X_M, we are finding the M points such that $\mathbf{f}_M$ most closely approximates a sufficient statistic of $\mathbf{f}$. Additionally, the inducing inputs X_m are now variational parameters instead of additional kernel parameters, which might mitigate the issue of overfitting discussed in Section 3.2.2.

3.3.4.1 Implementation - Derivation of the SGPR equations

If coded naively the computational complexity of the SGPR variational model is still $\mathcal{O}(N^3)$ in complexity. However, because of the form of the augmented covariance matrix and by using some well-known matrix properties, we can reduce the complexity to $\mathcal{O}(NM^2)$. This chapter will deal with the details of the numerical implementation of the variational SGPR model, following [28].

First, we should state some matrix identities used in the following derivations. Matrix dimensions that all the matrices should conform to in order for the statements to hold will be denoted in the subscripts. All matrices that are inverted are assumed to be invertible.

$$\mathrm{Tr}(\mathbf{A}_{M\times N}\mathbf{B}_{N\times M}) = \mathrm{Tr}(\mathbf{BA}) \tag{3.42}$$

$$(\mathbf{A}_{N\times N}^T)^{-1} = (\mathbf{A}^{-1})^T = \mathbf{A}^{-T} \tag{3.43}$$

$$(\mathbf{A}_{N\times N}\mathbf{B}_{N\times N})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1} \tag{3.44}$$

Lemma 3.3.3 (**Matrix determinant lemma**).

$$\det(\mathbf{A}_{N\times N} + \mathbf{U}_{N\times M}\mathbf{W}_{M\times N}\mathbf{V}_{M\times N}) = \det(\mathbf{W}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})\det(\mathbf{W})\det(\mathbf{A})$$

Lemma 3.3.4 (**Woodbury identity or Matrix inversion lemma**).

$$(\mathbf{A}_{N\times N} + \mathbf{U}_{N\times M}\mathbf{W}_{M\times M}\mathbf{V}_{M\times N})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{W}^{-1} + \mathbf{V}\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}\mathbf{A}^{-1}$$

Using the Woodbury identity, instead of calculating the inverse of an N-by-N matrix on the left, we can only calculate the inverse of an M-by-M matrix on the right. As we have so far, we suppose M is much smaller than N. We start with the log marginal likelihood lower bound, alreeady given by 3.41 as:

$$\begin{aligned} F(\mathbf{X}_M) &= \log\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N) - \frac{1}{2}\sigma_\epsilon^{-2}\,\mathrm{Tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN}) = \\ &= -\frac{N}{2}\log 2\pi - \frac{1}{2}\log\det(\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N) - \frac{1}{2}\mathbf{y}^T[\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N]^{-1}\mathbf{y} - \\ &\quad - \frac{1}{2}\sigma_\epsilon^{-2}\,\mathrm{Tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN}) \end{aligned} \tag{3.45}$$

Because the matrix $\mathbf{K}_{MM}$ is a kernel matrix, it is Hermitian, positive-definite, and real so we can use the Cholesky decomposition $\mathbf{K}_{MM} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is an M-by-M real lower triangular matrix with positive diagonal entries. First, we deal with the determinant factor by using lemma 3.3.3:

$$\begin{aligned} \det(\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N) &= \det(\mathbf{K}_{MM} + \sigma_\epsilon^{-2}\mathbf{K}_{MN}\mathbf{K}_{NM})\det(\mathbf{K}_{MM}^{-1})\det(\sigma_\epsilon^2\mathbf{I}_N) = \\ &= \det(\mathbf{L}\mathbf{L}^T + \sigma_\epsilon^{-2}\mathbf{K}_{MN}\mathbf{K}_{NM})\det(\mathbf{L}^{-T})\det(\mathbf{L}^{-1})\det(\sigma_\epsilon^2\mathbf{I}_N) = \\ &= \det(\mathbf{I}_M + \sigma_\epsilon^{-2}\mathbf{L}^{-1}\mathbf{K}_{MN}\mathbf{K}_{NM}\mathbf{L}^{-T})\det(\sigma_\epsilon^2\mathbf{I}_N) \end{aligned} \tag{3.46}$$

For the sake of brevity we define two matrices as $\mathbf{U} = \sigma_\epsilon^{-1}\mathbf{L}^{-1}\mathbf{K}_{MN}$ and $\mathbf{W} = \mathbf{I}_M + \mathbf{U}\mathbf{U}^T$. This results in the following expression of the effective covariance matrix determinant:

$$\det(\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N) = \det(\mathbf{W})\det(\sigma_\epsilon^2\mathbf{I}_N) = \sigma_\epsilon^{2N}\det(\mathbf{W}) \tag{3.47}$$

Secondly, we deal with the N-by-N matrix inverse term by using 3.3.4:

$$[\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N]^{-1} = \sigma_\epsilon^{-2}\mathbf{I}_N - \sigma_\epsilon^{-4}\mathbf{K}_{NM}[\mathbf{K}_{MM} + \sigma_\epsilon^{-2}\mathbf{K}_{MN}\mathbf{K}_{NM}]^{-1}\mathbf{K}_{MN} \tag{3.48}$$

To obtain a better conditioned matrix for inversion we will use a trick provided below in blue.

$$\begin{aligned}
[\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N]^{-1} &= \sigma_\epsilon^{-2}\mathbf{I}_N - \sigma_\epsilon^{-4}\mathbf{K}_{NM}\mathbf{L}^{-1}\mathbf{L}[\mathbf{K}_{MM} + \sigma_\epsilon^{-2}\mathbf{K}_{MN}\mathbf{K}_{NM}]^{-1}\mathbf{L}^T\mathbf{L}^{-T}\mathbf{K}_{MN} = \\
&= \sigma_\epsilon^{-2}\mathbf{I}_N - \sigma_\epsilon^{-4}\mathbf{K}_{NM}\mathbf{L}^{-1}[\mathbf{L}^{-1}(\mathbf{K}_{MM} + \sigma_\epsilon^{-2}\mathbf{K}_{MN}\mathbf{K}_{nm})\mathbf{L}^{-T}]^{-1}\mathbf{L}^{-T}\mathbf{K}_{MN} = \\
&= \sigma_\epsilon^{-2}\mathbf{I}_N - \sigma_\epsilon^{-4}\mathbf{K}_{NM}\mathbf{L}^{-1}[\mathbf{I}_M + \sigma_\epsilon^{-2}\mathbf{L}^{-1}\mathbf{K}_{MN}\mathbf{K}_{NM}\mathbf{L}^{-T}]^{-1}\mathbf{L}^{-T}\mathbf{K}_{MN} = \\
&= \sigma_\epsilon^{-2}\mathbf{I}_N + \sigma_\epsilon^{-2}\mathbf{U}^T\mathbf{W}^{-1}\mathbf{U}
\end{aligned} \tag{3.49}$$

This matrix is better conditioned because, for many kernels, it has eigenvalues bounded both above and below. A longer discussion on this topic is given in Section 3.4.3 of [3]. In short, if the eigenvalues of a matrix are very close to zero, numerical issues might occur when we try to invert it. This is because small changes in the input values or errors in the computations can cause large changes in the output values. When we invert a matrix, we are dividing by its determinant, which is the product of its eigenvalues. If one or more of the eigenvalues are close to zero, then the determinant is close to zero, and the inverse of the matrix can be very large. When the eigenvalues of a matrix are close to zero, the matrix is said to be ill-conditioned, which is what we are trying to avoid.

With expressions 3.47 and 3.49 we can expand the lower bound as:

$$\begin{aligned}
2F(\mathbf{X}_M) &= -N\log 2\pi - \log\det(\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N) - \mathbf{y}^T[\mathbf{Q}_{NN} + \sigma_\epsilon^2\mathbf{I}_N]^{-1}\mathbf{y} \\
&\qquad - \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{K}_{NN} - \mathbf{Q}_{NN}) = \\
&= -N\log 2\pi - \log(\sigma_\epsilon^{2N}\det\mathbf{W}) - \mathbf{y}^T\sigma_\epsilon^{-2}\mathbf{I}_n\mathbf{y} - \sigma_\epsilon^{-2}\mathbf{y}^T\mathbf{U}^T\mathbf{W}^{-1}\mathbf{U}^{-1}\mathbf{y} \\
&\qquad - \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{K}_{NN}) - \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{Q}_{NN}) = \\
&= -N\log 2\pi - \log\det\mathbf{W} - N\log\sigma_\epsilon^2 - \sigma_\epsilon^{-2}\mathbf{y}^T\mathbf{y} - \sigma_\epsilon^{-2}\mathbf{y}^T\mathbf{U}^T\mathbf{W}^{-1}\mathbf{U}^{-1}\mathbf{y} - \\
&\qquad - \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{K}_{NN}) - \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{U}\mathbf{U}^T)
\end{aligned} \tag{3.50}$$

Where we used the fact that $\sigma^{-2}\operatorname{Tr}(\mathbf{Q}_{nn}) = \operatorname{Tr}(\mathbf{U}\mathbf{U}^T)$. This is shown as:

$$\begin{aligned}
\operatorname{Tr}(\mathbf{U}\mathbf{U}^T) &= \operatorname{Tr}(\sigma_\epsilon^{-1}\mathbf{L}^{-1}\mathbf{K}_{MN}(\sigma_\epsilon^{-1}\mathbf{L}^{-1}\mathbf{K}_{MN})^T) = \\
&= \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{L}^{-1}\mathbf{K}_{MN}\mathbf{K}_{NM}\mathbf{L}^{-T}) = \\
&= \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{K}_{NM}\mathbf{L}^{-T}\mathbf{L}^{-1}\mathbf{K}_{MN}) = \\
&= \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{K}_{NM}(\mathbf{L}\mathbf{L}^T)^{-1}\mathbf{K}_{MN}) = \sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{Q}_{NN})
\end{aligned} \tag{3.51}$$

Again for the sake of brevity, we define $\mathbf{c} = \sigma_\epsilon^{-1}\mathbf{L}_W^{-1}\mathbf{U}\mathbf{y}$, where $\mathbf{L}_W$ comes similarly from the Cholesky decomposition of the matrix $W = \mathbf{L}_W\mathbf{L}_W^T$. Finally, we reach a numerically stable implementation of the lower bound of the log marginal likelihood:

$$F(\mathbf{X}_M) = -\frac{N}{2}\log 2\pi - \frac{1}{2}\log\det\mathbf{W} - \frac{N}{2}\log\sigma_\epsilon^2 - \frac{1}{2}\sigma_\epsilon^{-2}\mathbf{y}^T\mathbf{y} - \frac{1}{2}\mathbf{c}^T\mathbf{c} - \frac{1}{2}\sigma_\epsilon^{-2}\operatorname{Tr}(\mathbf{K}_{NN}) - \frac{1}{2}\operatorname{Tr}(\mathbf{U}\mathbf{U}^T) \tag{3.52}$$

We should briefly remember that the variational optimization of inducing inputs relies on the maximization of this lower bound $F(\mathbf{X}_M)$, which is equivalent to minimizing the negative lower bound on the log marginal likelihood $-F(\mathbf{X}_M)$. Additionally, we should note that, even though we omitted it in the previous derivations, this lower bound depends on the kernel hyperparameters too. We can also jointly optimize for both inducing inputs $\mathbf{X}_M$ and kernel hyperparameters $\boldsymbol{\theta}$, in which case it would be more accurate to write $F(\mathbf{X}_M, \boldsymbol{\theta})$.

3.3.5 Optimization of inducing inputs

With the derived numerically stable form of the lower bound on the log marginal likelihood $F(\mathbf{X}_M, \boldsymbol{\theta})$ in 3.52 we can now implement and visualize the optimization procedure. We will show the procedure for one-dimensional input locations. We chose some deterministic function and added white Gaussian noise to create a training set. The initial values for the inducing inputs and the initial values of hyperparameters are intentionally chosen to be unfavorable. The resulting procedure is shown in Figure 3.1. Initial inducing inputs are placed in the center of the interval of interest, and during the optimization they disperse throughout it. From Figure 3.1 it is clear that even after only a couple of optimization steps, for example 20, the achieved mean predictor fits the real latent function well. This might be an indication that the longer continuous optimization is unnecessary if we can tolerate certain error levels.

3.4 Comparison of different strategies

As the last point of this chapter, we will compare all mentioned SGPR methods. For the purpose of this comparison, we sampled a two-dimensional map from a GP prior with an isotropic kernel. This map, or target latent function, is shown in Figure 3.2. We generate a training data set by scattering an abundance of spatially uniformly sampled points over the area of interest, and add a negligible amount of noise to the latent function. Adding a small amount of noise is favorable for numerical stability, as it adds a small positive value to the main diagonal of covariance matrices that are to be inverted. Additionally, we suppose during the SGPR procedure that we already know the true hyperparameters. As already mentioned, SGPR models rely on choosing some active set, which could be either a subset of training data or pseudo-inputs. We would only like to compare different active set selection strategies. For this reason, we remove the issue of hyperparameter optimization to keep the active set as the only remaining variable. The comparison

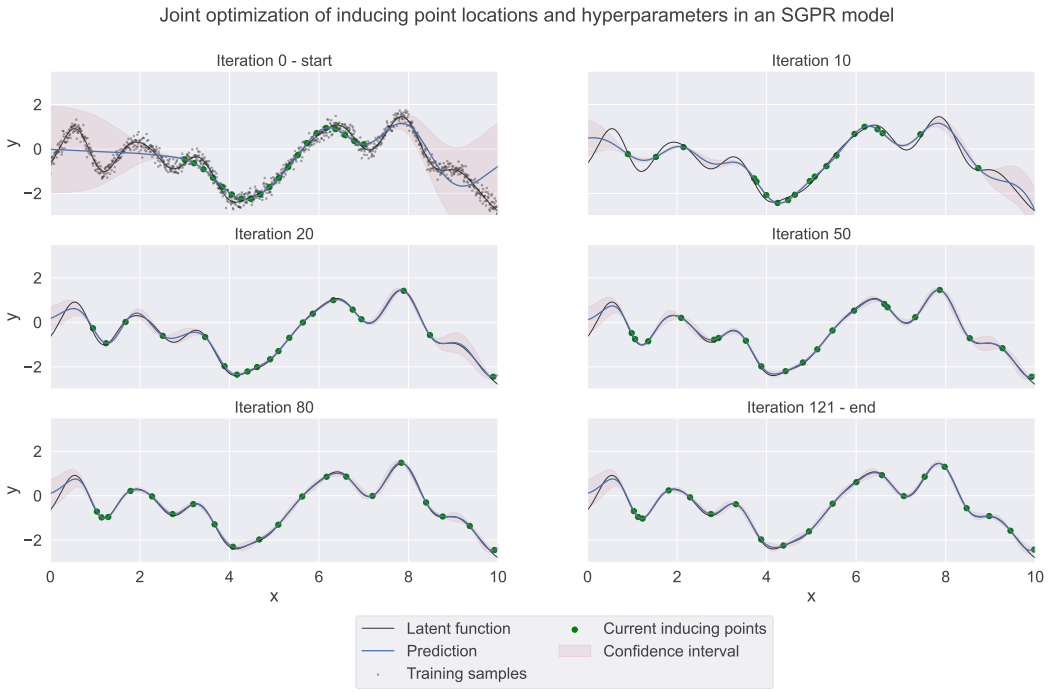


Figure 3.1: Joint optimization of inducing inputs and hyperparameters using variational learning

strategy is simple, for different values of M we simulate each strategy 10 times and calculate the average Mean Squared Error (MSE) of 10 runs. This averaging is carried out because all strategies require random initialization, so we would like to estimate the average performance of multiple runs. The results of this simulation are shown in Figure 3.3.

There are several conclusions we can draw from the presented results. Firstly, it is evident that random subsampling methods demonstrate the poorest performance, which aligns with our intuition. The Projected Process Approximations outperform the simpler Subset of Data (SD) method, as it does not disregard the remaining inactive set. As anticipated, the variational learning approach exhibits the best performance due to the continuous optimization process discussed earlier. Nevertheless, as the value of M increases, the performance gap between the variational approach and other methods narrows. Ultimately, for sufficiently large M values, all the described methods achieve similar and minimal MSE. This emphasizes the significance of the number of inducing inputs M as a parameter, as with the appropriate selection of M, we can achieve good performance regardless of the chosen SGPR procedure. Naturally, the computational complexity of the sparse model also scales with M, resulting in increased computational demands.

Another noteworthy finding is that the MES strategy does not yield a significant perfor-

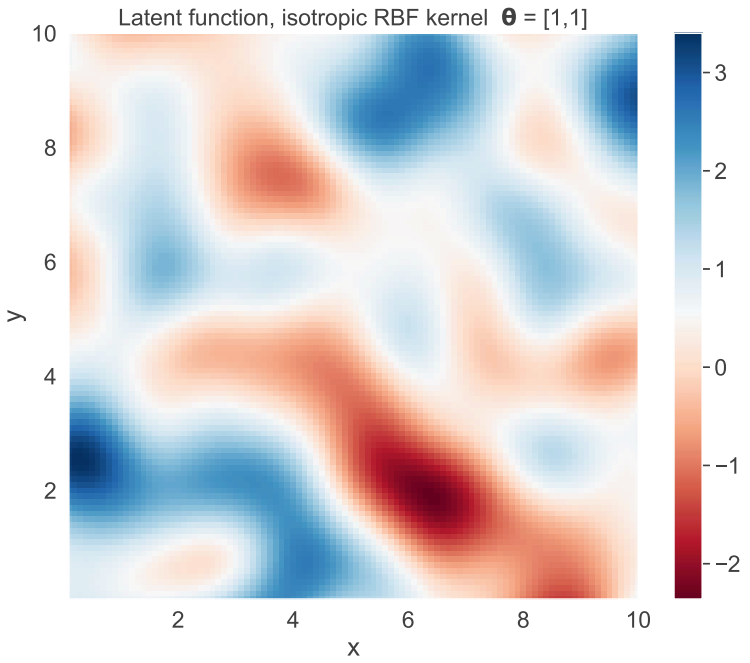


Figure 3.2: Latent function map

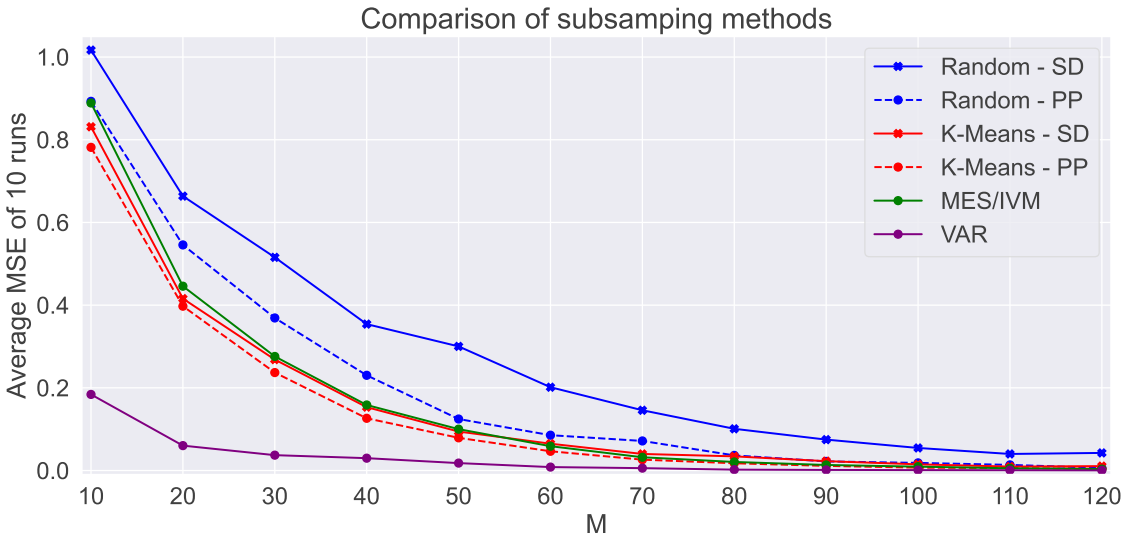


Figure 3.3: Comparison of different SGPR strategies

mance improvement compared to clustering. In fact, for small M values, it even performs slightly worse. This observation can be explained geometrically. Referring back to Figure 2.6 and the subsequent discussion, when the data set is sparsely distributed in space, the GP predictive mean reduces to the shape of the kernel placed at each data point. In the

interstitial region, the GP predictive mean defaults to the prior mean, set to zero. In the same regions, the GP predictive variance is at its highest. For 2D location space, the RBF kernel takes the form shown in Figure 2.5, projecting as a disk shape onto the plane. Thus, for the MES sampling strategy, when M is small, we essentially place disjoint disks over the area of interest. This is similar to what the K-Means clustering algorithm does. Since K-Means clusters points based on the Euclidean distance, it divides the input space into approximately disjoint disks represented by cluster centers and radii. In the case of MES, the greedy algorithm tends to choose points close to the edge of the area of interest, because it pushes the inducing inputs towards sparsely sampled areas. However, K-Means distributes the cluster centers more evenly than MES, leading to a slight performance advantage for smaller M values. It is only when we cover the entire area with disks, achieving a certain density of inducing points, that the MES strategy starts to yield benefits. This analysis suggests that K-Means would serve as a good initialization strategy for the MES SGPR model, eliminating the need for a lengthy greedy algorithm in which we select the active set point by point.

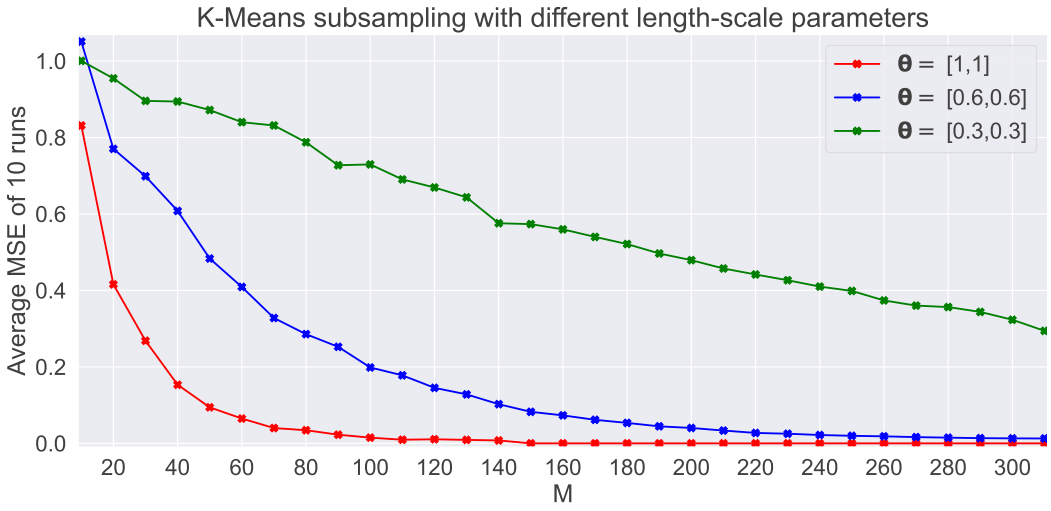


Figure 3.4: K-Means subsampling with different length-scale parameters

Thus far, we have observed that all discussed SGPR models converge in performance as the size of the active set M increases. As the last point, we would like to investigate what the active set size M, after which the methods converge, depends on. We posit that it will depend on the appropriate length-scale parameter, relative to the size of the area of interest. If the length-scale parameter remains the same but the size of the area of interest increases, the needed number of inducing inputs increases. If, on the other hand, the area of interest remains the same but the length-scale parameter becomes smaller, the needed active set size again increases. To showcase this effect, we present the results of another experiment similar to the one shown in 3.3. This time, however, we will use the same Sparse Gaussian Process Regression model, namely SD with K-Means, but vary the hyperparameters. The area of interest remains the same size as in the previous

experiment. In Figure 3.4 we see the results for different values of an isotropic 2D RBF kernel. The simulation results confirm our assumption, revealing that as the length-scale decreases, a higher number of inducing inputs is necessary to achieve the same level of MSE. This finding underscores the relevance of estimating the length-scale parameter prior to the GP training process. Approaches like the one proposed in 2.3.3 or leveraging prior knowledge about the data can be employed for this purpose.

CHAPTER 4

Gaussian Processes in Wireless Communication

4.1 Gaussian Processes in Wireless Networks

After diving deep into GP models on a theoretical level, we would now like to present some use-cases specifically in the field of Telecommunications. As it turns out, there are plenty of ways GPs can be exploited in this area. To demonstrate the wide-ranging utility of GPs in wireless communication, we put forward a review of some applications presented in the current literature.

4.1.1 Traffic Prediction

In order to plan and manage a communication network in the most efficient possible way, comprehensive understanding of the traffic load, it's temporal evolution, intensity and structure, is needed. Traffic prediction and modeling could help network planners optimize network resources, ensure Quality of Service (QoS), and deliver the best possible user experience while minimizing costs. This is especially important in wireless networks as they have limited resources, so accurate traffic prediction and modeling could allow for effective and efficient resource allocation. With accurate traffic models network planners can determine the capacity requirements of the network, optimize QoS parameters, the network topology, configuration, and routing to minimize congestion and improve performance. Additionally, network managers can, by analyzing and predicting traffic patterns, identify areas of high demand that require additional network infrastructure investment and areas of low demand that can be serviced with lower-cost infrastructure. Overall, traffic prediction and modeling play a critical role in ensuring the effective and efficient management of wireless networks. For this reason, accurate traffic modelling and prediction has a long-standing appeal in the research area of communication networks.

A GP model was proposed in [29] and [30] using the Alternating Direction Method of Multipliers (ADMM) for distributed hyper-parameter optimization to predict 4G wireless traffic, which was shown to outperform some Deep Neural Networks (DNN) models, such as Long Short-Term Memory (LSTM). A scheme combining GP and LSTM was proposed in [31] to generate accurate cellular traffic load prediction. The proposed scheme achieved state-of-the-art performance when compared to benchmark schemes.

In [32], an adaptive load balancing algorithm was proposed using an online GP to estimate path status and allocate traffic load to each path properly in multimedia multipath systems. This GP-based algorithm can provide higher reliability and stability by utilizing various communication media and paths.

4.1.2 Localization

Due to the increasing demand for location-based applications, wireless localization has become a cornerstone of modern life. Some commercially interesting applications of localization are in indoor navigation systems, which can be used to help people navigate large buildings such as shopping malls, airports, and hospitals. Another example is in asset tracking, where localization can be used to track the location of goods and equipment in warehouses, distribution centers, and factories. In addition, localization is becoming increasingly important in Internet of Things (IoT), where it can be used to track the location of devices and sensors. Localization research in wireless networks has significant commercial potential, with applications in a wide range of industries and sectors.

Localization also has the potential to improve the QoS, optimize network performance, enhance security and privacy, and enable new technologies and applications. Wireless applications and services often require location information, and accurate localization can improve the quality of these services, leading to better user experiences and increased revenue opportunities. Localization can also be used to optimize network performance by enabling location-based traffic engineering and load balancing. In addition, localization can enhance security and privacy by enabling location-based access control and intrusion detection, and it can be used in disaster management to improve emergency response.

GPs possess exceptional abilities for nonlinear regression and uncertainty modeling, which makes them ideal for indoor wireless localization tasks such as wireless tracking. In [33], a distributed recursive GP framework was proposed to create multiple local Received Signal Strength (RSS) maps, reducing computational complexity on big data from large-scale sensor networks. A global map is subsequently generated by merging all local RSS maps. The proposed framework demonstrates excellent positioning accuracy in both static fingerprinting and mobile target tracking scenarios.

In [34], a novel scheme combining crowdsourcing and GP regression is proposed to adapt radio maps to environmental dynamics in an online manner. The method recursively fuses crowdsourced fingerprints with an existing offline radio map, and it has distinct advantages in efficiency and scalability.

In order to compensate for frequency-dependent shadowing effects and multipaths in RSS observations, a GP model was proposed in [35] for calibrating multichannel RSS observations to improve localization accuracy over a large space. The application of the GP model enables more effective combination of multichannel RSS observations.

WiFi localization is an accurate method of indoor and outdoor localization and can be used in location-aware applications, but it requires a training set of signal strength readings labeled against a ground truth location map which is difficult to maintain. A new technique using Gaussian Process Latent Variable Model (GP-LVM) is proposed in [36] to solve the WiFi Simultaneous Localization and Mapping (SLAM) problem and reconstruct a topological connectivity graph from a signal strength sequence for efficient localization.

4.1.3 UAV Trajectory Planning

Wireless communication is strongly linked to the emerging technology of Unmanned Aerial Vehicles (UAVs). UAVs have a wide range of commercial and useful applications, including aerial photography and videography, delivery and logistics, agriculture, search and rescue, environmental monitoring, and infrastructure inspection. They have the potential to increase efficiency, improve safety, and reduce costs in many industries. Trajectory planning for UAVs involves determining a safe and optimal path for the UAV to follow during flight. It is done to reduce energy consumption, avoid obstacles, and ensure that it can reach its destination efficiently and safely with reduced risk of collisions. UAV trajectory planning is an interesting research area in wireless communication because it involves designing communication protocols that can handle the unique challenges posed by UAVs, such as their mobility, limited battery life, and a complex and time-variant communication environment. By integrating trajectory planning with wireless communication, innovative solutions for optimizing UAV flight paths, enhancing situational awareness, and improving overall performance and safety of UAVs can be developed.

UAV trajectory planning has recently gained attention in the wireless communication field. Reinforcement Learning (RL) and DNN methods have been considered for wireless trajectory planning. However, GP-based approaches have unique features such as uncertainty representation and spatial consistency that make them interesting for wireless trajectory planning. In [37] a fast online planning, learning, and recovery approach for safe autonomous operations under unknown runtime disturbances is presented. GPR theory is used to continuously train and adapt a model using data collected during the autonomous operation, providing safe plans at runtime under previously unseen disturbances. A recovery procedure is triggered to guarantee safety and enable learning and replanning whenever a safety constraint is violated.

Gaussian Process Model Predictive Control (GP-MPC) scheme for path planning is presented in [38]. The scheme deals with the time-varying network delay, non-linearity,

and time-sensitive characteristics of multirotor-type UAVs. It increases the accuracy of path planning and state estimation for multirotor-type UAVs thorugh GPs.

In [39] the authors propose an obstacle-aware adaptive path planning algorithm for searching targets in cluttered environments for UAVs. Their method adopts a layered planning approach that employs a GP-based model of target occupancy to generate informative paths in continuous 3D space. Within this framework, an adaptive replanning scheme is introduced, such that information gain, field coverage, sensor performance, and collision avoidance are balanced for efficient target detection.

4.1.4 Signalling Reduction

Channel Quality Indicator (CQI) feedback is a mechanism used in mobile networks to provide information about the quality of the radio channel between the User Equipment (UE) and the Base Station (BS). The UE sends CQI feedback to the base station to inform it of the current channel conditions, which can be used to adjust various parameters in the network, such as modulation and coding schemes, transmission power, and resource allocation. CQI feedback is important because it enables the network to optimize its performance by adapting to changing channel conditions, thus improving the overall user experience. For example, if the channel quality is poor, the network can adjust the transmission parameters to improve the signal quality and reduce the risk of packet loss. On the other hand, if the channel quality is good, the network can increase the data rate to improve the throughput.

Reducing the CQI feedback signaling overhead is beneficial for several reasons. Firstly, it reduces the overall network load, which improves network performance and user experience. Secondly, it conserves battery life on mobile devices since signaling messages require significant power to transmit. Lastly, reducing CQI feedback overhead improves the scalability and efficiency of mobile networks, allowing them to handle more users and traffic without compromising network performance. CQI feedback is known to consume a significant portion of uplink resources in wireless networks. This can pose a challenge, especially in situations with high traffic load, such as music festivals, sporting events, or any other event gathering large crowds, which put extra pressure on wireless networks. In such scenarios, channel estimation strategies must be implemented to overcome the issue of signaling overhead while maintaining acceptable performance levels. Reducing CQI feedback signaling overhead is interesting to a Mobile Network Operator to maintain a competitive edge in the market and provide better services to their customers.

In [40], the authors propose a limited feedback selection scheme that allows the BS to obtain CQI feedback from a subset of users, while estimating the channel for the remaining users. They leverage the theory of GPR to estimate the CQI by exploiting the correlation property of the macroscopic shadow fading, which implies that users close to each other have a correlated channel. Their results show that the proposed approach achieves a significant reduction in feedback while maintaining an acceptable Block Error Ratio (BLER). The same idea is again explored in [41], where authors compare the performance

of the described GP-based CQI reduction scheme with a theoretical Cramer-Rao lower bound.

In [42], another GP-based method for CQI estimation is suggested. Here, the authors additionally take into account the temporal evolution of the users' CQI values. They propose a channel quality estimation method using GPR to predict users' channel states based on their mobility profiles. Simulation results show an extreme reduction in signalling overhead with still low packet loss rates.

4.1.5 Performance Evaluation

Several machine learning techniques can be used also for performance prediction problems in wireless networks. Performance prediction problems frequently entail using historical measurement data to anticipate network performance where direct measurements are unfeasible or unavailable. Some examples of performance estimation are the prediction of expected performance in the future for a specific mobile network user given historical data. These temporal prediction issues might, for example, use mobility profiles of users to anticipate where the user will be located in the next time instance. Another type of problem is a spatial prediction of some network parameter, such as for example received signal power, at locations where no measurements are available. Such problems are crucial for extending the results of drive tests to the adjacent areas not covered during the test. Although any regression approach could potentially be applied for spatial prediction, methods inspired by interpolation techniques tend to yield the most promising outcomes. Among them, GPR stands out by allowing direct estimation of essentially all relevant parameters from the data. GPR models the correlation between different locations and predicts the received signal strength at a new location based on its distance and similarity to nearby locations.

Spatial prediction problems are mostly focused on building so-called radio coverage, or performance maps. Radio performance maps that accurately predict the received signal strength at different locations based on the network topology and environment are a useful tool for planning and optimizing network coverage. Not only can they help in estimating the network coverage, but also the expected interference from neighboring Base Stations. In mobile networks, interference is the main performance limitation especially in dense urban environments. Therefore, minimizing and managing interference is crucial for achieving the most efficient mobile network.

In [43] the authors propose a GP-based model that predicts the local shadow fading while simultaneously fitting Path Loss (PL) to the data. The GP achieves spatial consistency and provides uncertainty estimates to the data. The authors validate the model on a data set from an indoor corridor environment. The GP approach outperforms some common PL models such as for example the log-distance and K-Nearest Neighbors (KNN) model.

In [44] the authors utilize GPs in several performance prediction scenarios. One in the aforementioned drive test extension, for Received Signal Strength Indicator (RSSI) but

also Carrier-to-Interference Ratio (CIR) and Data Rate measurements, and another for the temporal evolution of the same KPIs for a single moving user.

The propagation environment in dense urban areas is heavily influenced by tunneling effects and building blockages, which are not easily modeled by interpolation techniques such as GPR. To address this, in [45] the authors introduce a propagation-aware GPR that includes spatial information through the diffusion kernel. This graph-based method offers great flexibility in modeling various environments and as such not limited to a specific street grid layout.

CHAPTER 5

Railway Connectivity and Gaussian Processes

In this chapter, we will diverge from the discussion of GPs and explore a different topic related to railway connectivity. Railway communication scenarios possess unique characteristics that distinguish them from more common mobile cellular systems, where users are typically located outdoors on city streets or indoors within buildings. Consequently, specific connectivity solutions have been developed to cater to the needs of train passengers. Our exploration begins with an examination of railway connectivity aspects, highlighting the distinctive features and solutions tailored to this environment. Subsequently, we will delve into a measurement campaign conducted on an operational LTE network onboard a train. Detailed discussions will cover the process of data collection and preprocessing, with a particular emphasis on measurements of Reference Signal Receive Power (RSRP).

To gain a comprehensive understanding of the observed data, we will explore general concepts related to wireless propagation effects. This exploration will help us identify anticipated phenomena and analyze various propagation models commonly used in the field of wireless communication. Many of these common models are developed for certain types of subscriber surroundings. For example, they might use different parameters for users located in a dense city center and users in rural areas. To account for different propagation environments surrounding the train tracks, we will classify the collected data into two distinct classes, namely Urban and Rural, and present observed differences and similarities in received power measurements between these environments.

Furthermore, we will leverage GPR as an interpolation technique to extract valuable insights from the data. Specifically, we will utilize GPR to learn essential model parameters such as the spatial correlation properties of the observed power levels. As we conclude the chapter, we will touch upon the potential simplifications in complexity, drawing connections to the previously discussed SGPR models.

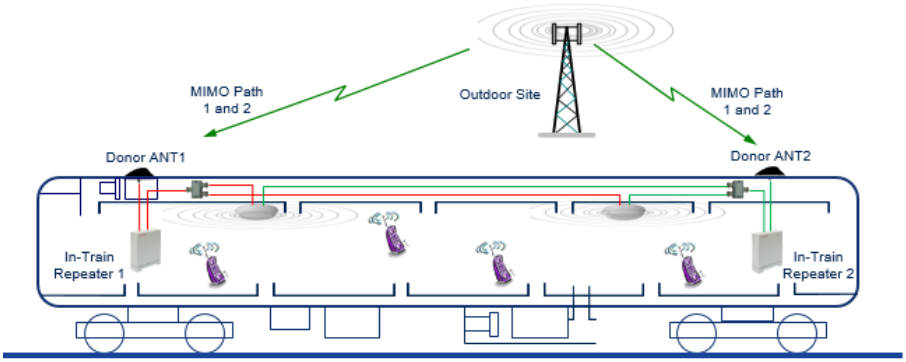
5.1 Achieving connectivity onboard trains

In a train, the windows of the cabins are often coated with a thin metal layer. The purpose of this coating is to reduce heat dissipation through the window panels so the inside heating or cooling system can efficiently maintain temperature and ensure a pleasant traveling experience. However, this metal coating, along with the metal exterior of the train cabin, creates an almost Faraday cage, causing high Vehicle Penetration Loss (VPL) [46]. These effects limit the amount of electromagnetic radiation that can penetrate the inside of the train cabin restricting wireless connectivity. As for passengers today, connectivity during hours of traveling is almost mandatory, railway companies have to get around this problem. One widely deployed solution is an Amplify-and-Forward (AAF) repeater, to which an outside antenna is mounted. The outside antenna is usually positioned on the roof such that the reception is not limited by the surrounding train cabin. The signal is then amplified and dissipated throughout the cabin with a leaky cable. This solution has some disadvantages. An active element such as a repeater introduces additional noise to the system, it requires maintenance and consumes power. Also, some changes in the technology used by the operators, such as for example introducing new frequency bands, require changes that have to be implemented in all train repeaters in a railway company fleet. Repeaters that are commercially available to railway companies are not designed to support a wide range of frequencies, but instead specifically tailored to the customer's current needs. Therefore, if a new frequency band has to be introduced, substantial capital expenditures occur for the railway operator.

In Figure 5.1a such a repeater system is pictured. On the Figure 5.1a, however, there are two roof antennas, repeaters, and leaky cables. Such a setup is beneficial because it preserves a Multiple Input Multiple Output (MIMO) channel. On the other hand, it is more expensive to install and operate, so it is sometimes not used. In the case of a single roof antenna, MIMO properties of the channel on the outside of the cabin are lost. The repeater setup creates a so-called keyhole channel, effectively turning the channel to a Single Input Multiple Output (SIMO) channel from the perspective of the users inside the cabin. This is another potential disadvantage of the repeater system, especially since, as we are moving towards 5G and 6G mobile networks, MIMO remains one of the key features for achieving high-capacity channels.

Another potential solution to the problem of limited wireless connectivity inside train cabins is the use of specialized window panes with a modified metal coating that allows wireless signals to pass through. These coatings have a laser-cut grid-like pattern that maintains the thermal insulation properties of the window while also enabling wireless communication. Several companies offer this technology, which may be a cost-effective and low-maintenance alternative to using active elements like AAF repeaters. In Figure 5.1b, one such window grid design is shown. As we can see from the close-up photo, this grid is a very simple rectangular one. The measurements performed in this thesis were done inside a cabin with such a simple design. There are, however, other ones on the market that have a more involved grid design, promising better characteristics and increased coverage inside the cabin.

Modified windows also have their drawbacks, as the grid design might be frequency, polarization, or Angle of Arrival (AoA) selective. Companies that produce modified window panes can easily perform tests in a laboratory environment to prove or disprove certain properties of the panes. But the quality of the solution can only be properly evaluated with tests in an actual train cabin, as the VPL depends also on the size of the window panes, their position and number, and the train cabin as a whole. So, before implementing this solution, it is important to conduct thorough testing and evaluation to ensure that the modified window panes provide adequate wireless coverage and meet passenger needs. Large-scale measurement campaigns along many kilometers of the track and on an operational mobile network are seldom done. The measurement campaign performed for the purposes of this thesis is one such large-scale evaluation.



(a) An AAF repeater sketch, from [47]

(b) A simple grid design of a modified window, picture taken by [48]

Figure 5.1: Current solutions for wireless connectivity in trains

5.2 Data Acquisition and Preprocessing

The data used in this work was collected during a measurement campaign onboard an empty Railjet train operated by the Austrian Federal Railways, or Österreichische Bundesbahnen (ÖBB). The route taken was Vienna-Innsbruck-Vienna. The track distance for a one-way journey is approximately 500 km, which shows that the measurement campaign was extensive in scope. Our measurement devices were five Samsung Galaxy Note 4 SM-N910F phones with NEMO measurement software from Keysight [49]. Five User Equipment (UE) devices were distributed across the cabin with Modified Permeable Windows. The devices used are pictured in 5.2. All measurements shown in this work were performed on band 3, meaning the downlink frequency of 1800 MHz, of an operational LTE network. Although we had available measurements of several Key Performance Indicators (KPIs), such as Channel Quality Indicator (CQI), download speed, call setup time, and so on, we are focusing specifically and only on one KPI of interest, namely Reference Signal Receive Power (RSRP).

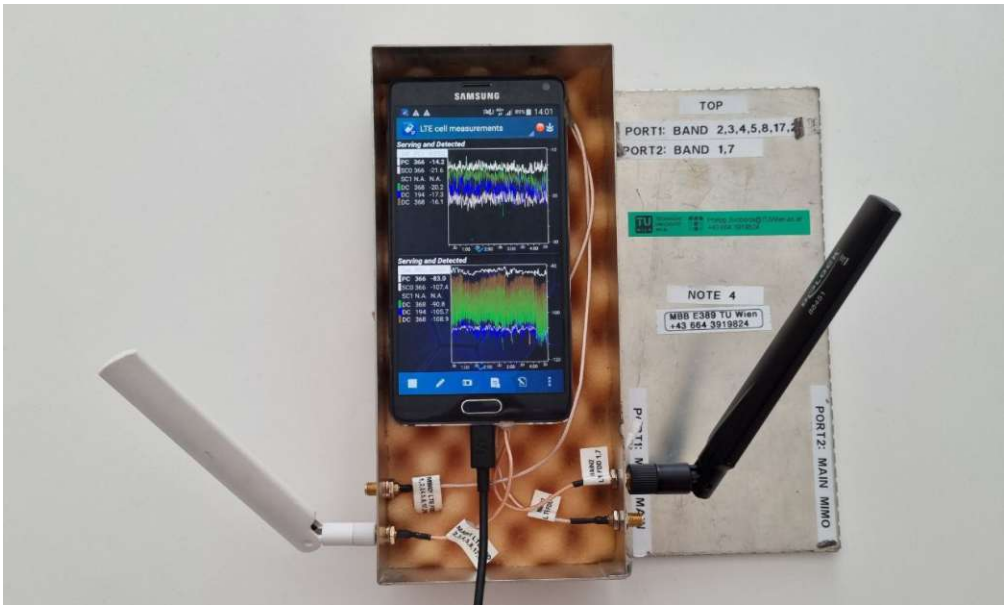


Figure 5.2: Measurement device - a Samsung phone with NEMO software

5.2.1 Localization

As a first step in the data processing phase, we handle the geolocation of our measurements. For the purposes of localization during the campaign, we used a standard GPS system. Reported GPS coordinates have unavoidable uncertainty. However, because trains cannot deviate from a specific track, GPS samples can be aligned to the known track positions. Luckily, we have available a comprehensive map of the entire Austrian railroad system from OpenStreetMap (OSM). [50] OSM is a collaborative project in which volunteers

create and update a freely accessible and editable map of the world. Contributors use GPS devices and aerial imagery to collect and modify data, making it freely available for download and use. It serves as a thorough resource, providing detailed information on various geographic features like roads, buildings, landmarks, and in this case railroad tracks. In Figure 5.3 we can see the traveled path retrieved from OSM.

Figure 5.3: Railway track connecting Vienna and Innsbruck retrieved from OSM

Raw GPS samples obtained in this measurement campaign contain a triplet of time, longitude, and latitude. The longitude and latitude components are translated to a (x, y) coordinate system in meters using the Austria Lambert coordinate projection system. The Austria Lambert coordinate system is a specific map projection used to represent geographic data in a two-dimensional format. It is based on the Lambert Conformal Conic projection, which is commonly used in cartography. The primary reason for using the Austria Lambert coordinate system instead of latitude and longitude directly from a GPS is to have a standardized and consistent spatial reference system specific to Austria. While latitude and longitude expressed in decimal degrees provide a global coordinate system that is widely used, the Austria Lambert coordinates are tailored to accurately represent the geographic features of Austria. The Lambert Conformal Conic projection minimizes distortions in shape, distance, and area within each zone, providing a more precise representation of the Austrian territory compared to latitude and longitude. As mentioned, the translated GPS coordinates may not precisely match the exact coordinates of the train tracks due to inherent imperfections in the GPS system. In such scenarios, the measured coordinates are transformed into their corrected counterparts through an orthogonal projection onto the track. This projection ensures that the samples align with the track point that has the minimum Euclidean distance from the raw measurement.

Secondly, we can calculate the distance along the track from the start to the mapped (x, y) point and convert a 2D coordinate into a single one, named d. This new coordinate d denotes the distance from the starting point of the track, in our case chosen as Vienna Main Station, measured along the train track. In this way, we translated a triplet of time, latitude, and longitude into an ordered pair (t, d) without any loss of information.

Still, as RSRP and other measurements conducted during this campaign are not synchronized to the measurements of the GPS system we need a way to infer the spatial coordinate of samples that were measured in the interstitial period between two GPS samples. Also, positioning information is missing in areas where the GPS system failed, such as tunnels, for which interpolation is again necessary. The phone fleet used in this study was connected to a centralized GPS system, causing the entire train cabin to be considered as a single moving point. This introduces an unavoidable localization error and makes it impossible to localize the measurement devices independently. The cabin length of a Railjet train is approximately 20 meters, which results in at least 10 meters of error if we assume that we are able to perfectly localize the center of the moving cabin at each time instance. We enforce a smooth functional mapping between time and distance by interpolating the collected ordered pair samples (t, d). The resulting interpolated distance at the time instance t_x, denoted as d_x is calculated as:

$$d_x = d_1 + (d_2 - d_1)\frac{t_x - t_1}{t_2 - t_1} \tag{5.1}$$

where (t_1, d_1) and (t_2, d_2) are the closest measured coordinates to t_x in time, and such that $t_1 < t_x < t_2$. Despite all the issues discussed, it's still reasonable to assert that location uncertainty is less of a problem in our railway data set compared to urban crowdsourced scenarios. In urban settings, dealing with localization errors can be a significant headache, particularly because it is difficult to separate indoor and outdoor users based on GPS alone. In our railway data, it is relatively less of a concern as we are aware of the exact scenario in which our users are situated.

5.3 Data analysis

In this section, we will first discuss the basics of wireless propagation, to understand what phenomena we expect to observe in the data and which parameters we would like to extract from it. Then we will explain some specifics of the processing needed to obtain the classification of the train surroundings, namely the Urban and Rural categories. Lastly, we will present the results of the measurement campaign and some conclusions we draw from them.

5.3.1 Propagation effects in wireless communication

In wireless communication various propagation effects can impact the quality and reliability of received signals. However, most accepted models of wireless propagation rely on three main effects. These three common propagation effects are Path Loss (PL), Shadow Fading, and Small-Scale Fading. [51] Path Loss is the overall reduction in signal strength as it propagates through space, primarily influenced by the distance between the transmitter and receiver. This distance dependence measured in dB is most commonly approximated by models that have a linear slope in the logarithmic domain, such as:

$$\mathrm{PL}(d) = \mathrm{PL}_0 + 10n \log_{10} \frac{d}{d_0} \tag{5.2}$$

where n is the path loss exponent and $\mathrm{PL}_0 = PL(d_0)$ is the intercept of the linear model, meaning the Path Loss at some starting distance d_0. This form, for example, appears when we consider the free space path loss model, with the Friis' power transmission formula for free space. The Friis' formula additionally depends on the transmitter and receiver antenna gain and signal frequency. However, most other more complex empirical models also retain the same linear slope in the logarithmic domain form, with added parameters such as average street width, average building height, receiver and transmitter heights, and so on, as for example in [52]. Another common criteria for model selection is whether the subscriber is in Line-of-Sight (LOS) or Non-Line-of-Sight (NLOS) conditions. These additional parameters are meant to adjust the path loss exponent n, which varies with terrain and environment.

Shadow Fading, on the other hand, refers to the signal attenuation caused by obstacles, such as buildings or natural terrain, resulting in variations in signal strength. It reflects changes in propagation conditions as one, for example, turns a corner, or moves behind a large building or hill. Because shadowing is mostly caused by large obstacles it is therefore correlated over fairly large distances. Variations caused by Shadow Fading are shown to follow a log-normal distribution. That means that, when shadowing levels are measured in logarithmic units, they follow a Gaussian distribution. Consequently, shadowing effects are usually modeled by the addition of a zero-mean Gaussian random variable, with some standard deviation σ_{SF}, to the overall Path Loss. The standard deviation σ_{SF} of this random variable is usually empirically determined and might vary depending on the frequency, type of propagation environment, such as Urban or Rural, and heights of the BS and UE. Many measurement campaigns have already been conducted to investigate the statistical properties of shadow fading in different scenarios.

Lastly, Small-Scale Fading refers to the rapid fluctuations in signal strength caused by multipath propagation, when the radio waves take multiple paths due to reflections, diffraction, and scattering, leading to constructive and destructive interference and signal variations over short distances. Small-Scale Fading is often handled in a wireless system with diversity schemes. Contrary to Large-Scale Fading, which is usually only modeled as log-normal, Small-Scale fading has several commonly accepted statistical models. Some of them are the Rayleigh and Rice distributions, but also Nakagami-m and even Gaussian in some cases. Small-Scale fading is difficult to predict and model on a large scale, such as in our railway scenario. Therefore, we will only treat it as inherent random measurement noise. In order to avoid Small-Scale fading distorting our conclusions, we subsample instances where the train was standing still. In these cases the fluctuations in the measured RSRP are due only, or mostly, to Small-Scale fading.

5.3.2 Base Station Mapping

To estimate the Path Loss (PL) we first find the 3D distance to the serving BS. We have available a map of all Base Stations of the Austrian MNO on whose network we measured during the campaign. To align the measurement to a serving BS we use the Physical Cell Identifier (PCI) to filter only BS locations with that cell identifier, and

then pronounce the closest one as the serving BS for our measurement. In a 4G mobile network, the PCI is a unique identifier assigned to each cell within a base station, used for cell identification. The BS map we had available is however imperfect, as we have some incomplete or outdated information. Because of the imperfect data, we sometimes get false distance mappings of the serving BS. Another issue with the distance mappings are repeater chains, that are fairly common in Mobile Network deployments along railway tracks and highways. Repeater chains in a mobile network refer to a series of strategically placed signal repeaters or amplifiers along railway tracks and highways. They are used to enhance mobile network coverage, improve signal strength, and minimize frequent handovers in high-speed scenarios, as well as address signal loss challenges in tunnels. We have no information on hand as to where the repeater heads are placed, therefore it is much more difficult to estimate the distance to the true transmitter. In order to deal with this false or missing information and repeater chains we had to find a workaround. There is another way to check if the distance mapping is plausible, and that is by looking into the Timing Advance (TA) reported by the .

Timing Advance (TA) in 4G mobile networks is a parameter used to synchronize uplink transmissions between the User Equipment (UE) and the Base Station (BS). It ensures proper timing alignment by compensating for the propagation delay between the UE and the BS. The primary purpose of TA is to prevent interference and collisions between uplink transmissions from different UEs. TA is measured in discrete units. Each unit represents the duration of one chip in the network's spreading sequence. The value of TA indicates the number of chip units that the UE needs to advance its uplink transmission timing. By adjusting the TA value, the BS instructs UEs to transmit their signals slightly earlier, compensating for the propagation delay and aligning the arrival of signals at the BS.

While TA itself is not directly used for localization or estimating the distance between the UE and the BS, it can provide an indication of the round-trip delay or the propagation distance between them. By considering factors such as the speed of light and the network's time synchronization, TA can be used as a rough estimate of the user's proximity to the BS. One chip duration equals 78.125 meters of propagated distance if we assume a direct LOS propagation. We can assume that if the TA indicates a smaller distance than the calculated one the BS mapping is false, as the signal can't propagate faster than the speed of light on a direct path between the UE and BS. This result can be seen in 5.4. The different colors represent different PCI values, i.e. different measured cells. The shaded region indicates the BS mappings that are impossible based on the reported TA, with a slight margin as the TA reports are not perfectly synchronized with the RSRP measurements. The main diagonal on the Figure 5.4, represented in red, is the perfect LOS scenario where the actual distance and the reported TA values agree. In the Figure 5.4, we can also easily see the influence of repeater chains. These are the vertical zig-zag lines appearing on the right. In this case the distance to the BS is increasing (or decreasing) as we are moving away from the BS (or towards it), but the TA indicates only a small movement. This misalignment demonstrates that the actual distance the

signal travels is much shorter as there are periodically placed repeater heads along the traveled path. In conclusion, these impossible cases and repeater chains are excluded when estimating the PL, which will be described in the following section.

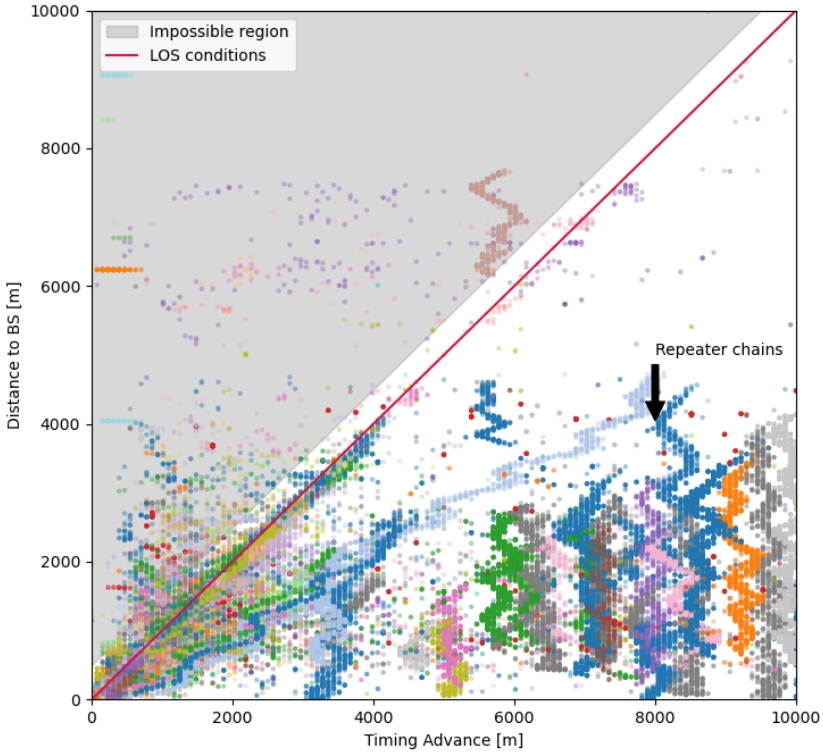


Figure 5.4: Mapped BS distance over corresponding reported TAs

5.3.3 Classification

As we have mentioned previously, when discussing wireless propagation effects, most common models suppose the surroundings of a subscriber will effect the received signal levels. Some notable characteristics in the surroundings that might effect wireless propagation are tall buildings, hills, foliage and so on. Usually, the differences in the surroundings are classified coarsely into only several classes. There are several accepted land use and terrain classifications that researchers have considered distinct. However, typical scenarios fall broadly into three categories, namely Urban, Rural, and Suburban, as in for example [52]. We will now focus on the classification of our data into measurements obtained in Urban and in Rural surroundings.

Corine Land Cover (CLC) is a project initiated by the EU Commission for the unified

classification of major land cover types. Digital satellite images from EU member states have been collected and evaluated, with a focus on changes in land use and their environmental impact. The project uses five main categories, which are further divided into a total of 44 European-wide land use classes. In Figure 5.5 we see the main five category types plotted for Austria. This freely available data set dates from the year 2018, which is the newest version available. It is possible some changes in the landscape occurred on smaller track segments since. Since the CLC project is not aimed at classifying the land use classes for the purpose of wireless network planning, the classes are not perfectly fitted to our needs. For this reason, we will keep the classification coarse, mapping the Artificial Surfaces class to Urban and all others to Rural scenarios. Classification into Rural, Urban, and Suburban areas for the purpose of wireless network planning is in some cases difficult even for professionals when an abundance of data and images is available. The information available from CLC is insufficient for any finer classification than what we present here.

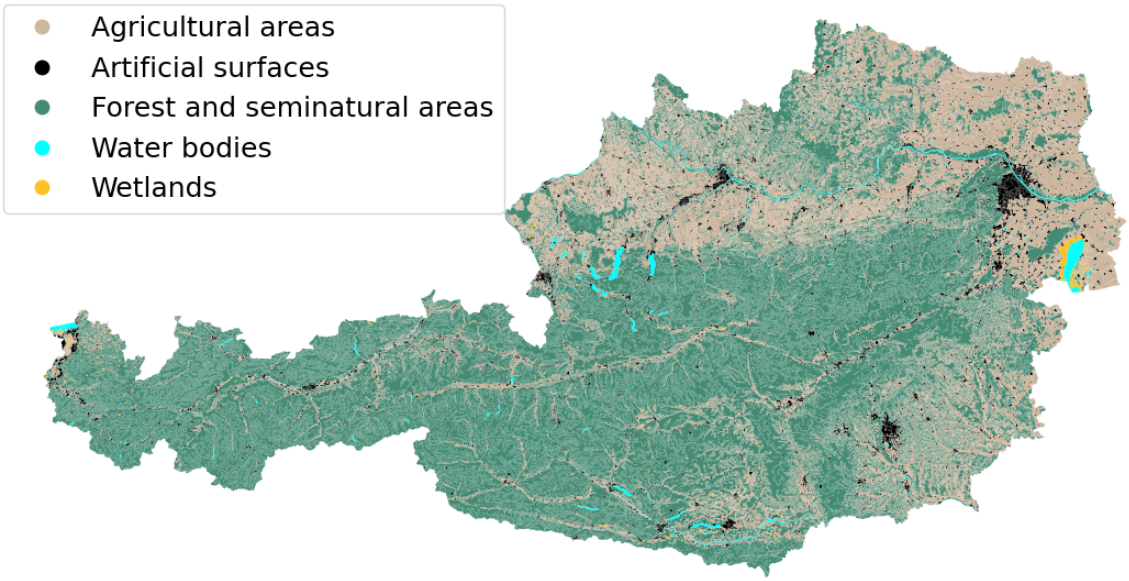


Figure 5.5: CORINE Land use Classes Map

First step in this stage of data processing was splitting the track obtained from OSM into segments of equal length, namely 100 meters. The 100-meter-long track segments are evaluated based on the percentage of Urban, i.e. Artificial Surfaces, surroundings. The surroundings are defined as an area in which every point is at most 100 meters away from the given track segment. A detail of the result of such a classification is shown in Figure 5.6. All RSRP measurements are classified in the same class as the corresponding 100-meter segment they fall into.

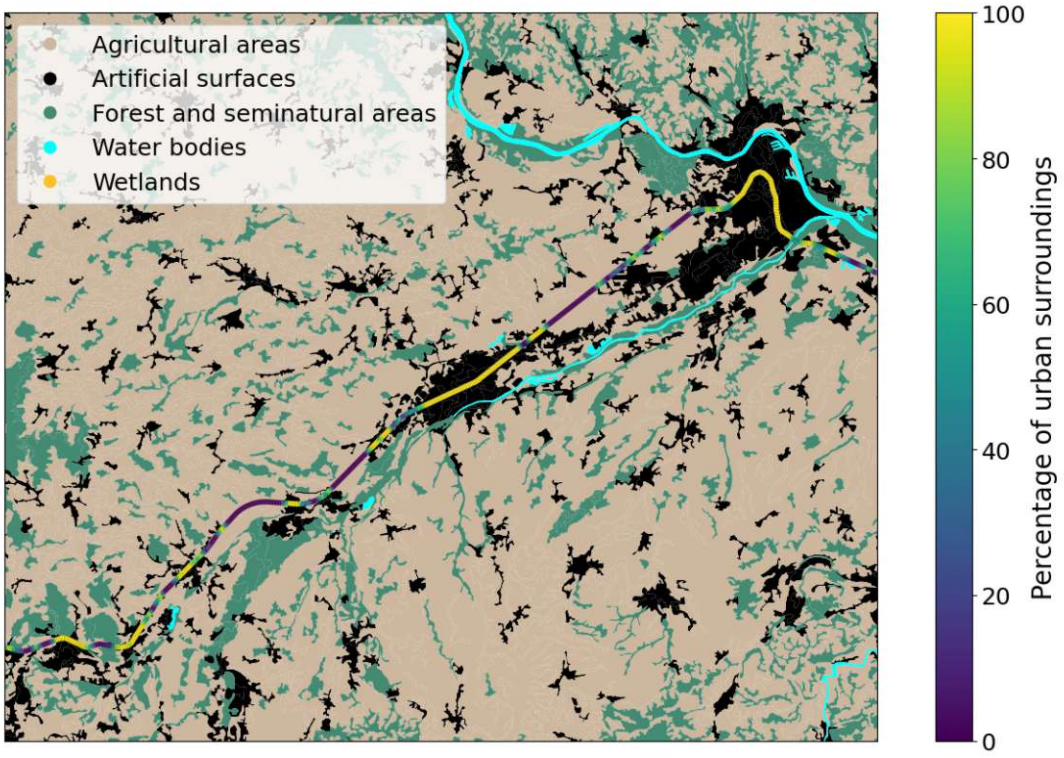


Figure 5.6: Land use classification of track segments detail

5.3.4 Path Loss estimation

In order to fit a Path Loss model to our measurements, we first calculated the 3D distance between the UE and the serving Base Station. However, because of the centralized GPS system, we are only able to localize the train cabin as a single moving point along the track. To combat the resulting location uncertainty, we round the obtained distances to the closest 20-meter multiple. The resulting data points are shown in Figure 5.7.

The most suitable Path Loss model, which aligns well with our observed data, is depicted in Figure 5.7. We have used a two-slope model, with one slope applicable to close proximity to the Base Station and another for greater distances from it. The transition between these two models occurs at a distance of 150 meters, as illustrated in Figure 5.7. The resulting Path Loss exponents and intercepts for the models are summarized in Table 5.1.

An interesting observation emerges in the Urban scenario, where the fitted model in close proximity to the BS appears as a relatively flat line. We believe that this phenomenon can be explained by the specific characteristics of the train cabin environment. In urban settings, BS antennas are typically positioned at elevated locations, such as building roofs. Consequently, when the train travels near a BS, the transmitting antenna predominantly radiates onto the train cabin roof. The train cabin roof acts as a shield,

obstructing the majority of the signal directed towards the subscriber positions. Only with increasing distance does the geometry change, causing the signal from the Base Station to deviate from being predominantly directed at the train roof. This hypothesis is further strengthened by the fact that this effect is less noticeable in rural scenarios. In rural environments, the proximity to the BS follows a different geometry, as the BS antennas are more frequently mounted at lower positions. Along highways and train tracks in rural areas, Base Stations are often strategically located to serve these transportation routes, resulting in lower elevations compared to typical Urban scenarios.

In the second part of the model, where the distance to the BS exceeds 150 meters, we observe similar behavior in both Urban and Rural cases. This might indicate that the train cabin and train track environment are specific enough on their own as a propagation environment, that the differences in the outside terrain or blockage density are overshadowed.

Figure 5.8 presents histograms of the observed RSRP values for the Urban and Rural scenarios. The blue histograms represent the original observed values, while the orange histograms represent the Shadow Fading values obtained by removing the mean Path Loss function from the RSRP values. Accompanying Gaussian distributions are depicted as solid lines, with the mean and standard deviation derived directly from the measurements and depicted in accompanying boxes.

Focusing on the original RSRP values, one might be inclined to conclude that the Urban scenario is more favorable due to its higher overall mean RSRP value. However, as we have observed from the Path Loss models, the overall path loss is fairly similar in Urban and Rural environments at distances larger than 150 meters. The discrepancy in mean values primarily arises from the density of deployed Base Stations. In Urban scenarios, the BS density is higher, resulting in shorter average distances. Consequently, the Path Loss is smaller for these shorter distances. Shifting our attention to the Shadow Fading values, we observe several noteworthy aspects. As anticipated, the Shadow Fading values conform to a zero mean Gaussian distribution. Although the observed mean of the Shadow Fading values is not precisely zero, it is very close, indicating that our Path Loss model is suitable for the observed measurements. Furthermore, the standard deviation is decreased compared to the raw measurements, reflecting reduced uncertainty in our observations due to knowledge of the average Path Loss model. Lastly, we observe that the standard deviation is again quite similar in both scenarios. This aligns with our earlier discussion regarding the PL model, suggesting that the unique characteristics of the train cabin and train track environment overshadow the effects of outside terrain or blockage density variations.

5.3.5 Gaussian Processes in Railway Propagation Modeling

As a last point of this thesis, we will turn to Gaussian Processes once again. In the previous discussion, we derived some empirical properties of the propagation scenario encountered during railway travel. We have, however, also mentioned one statistical

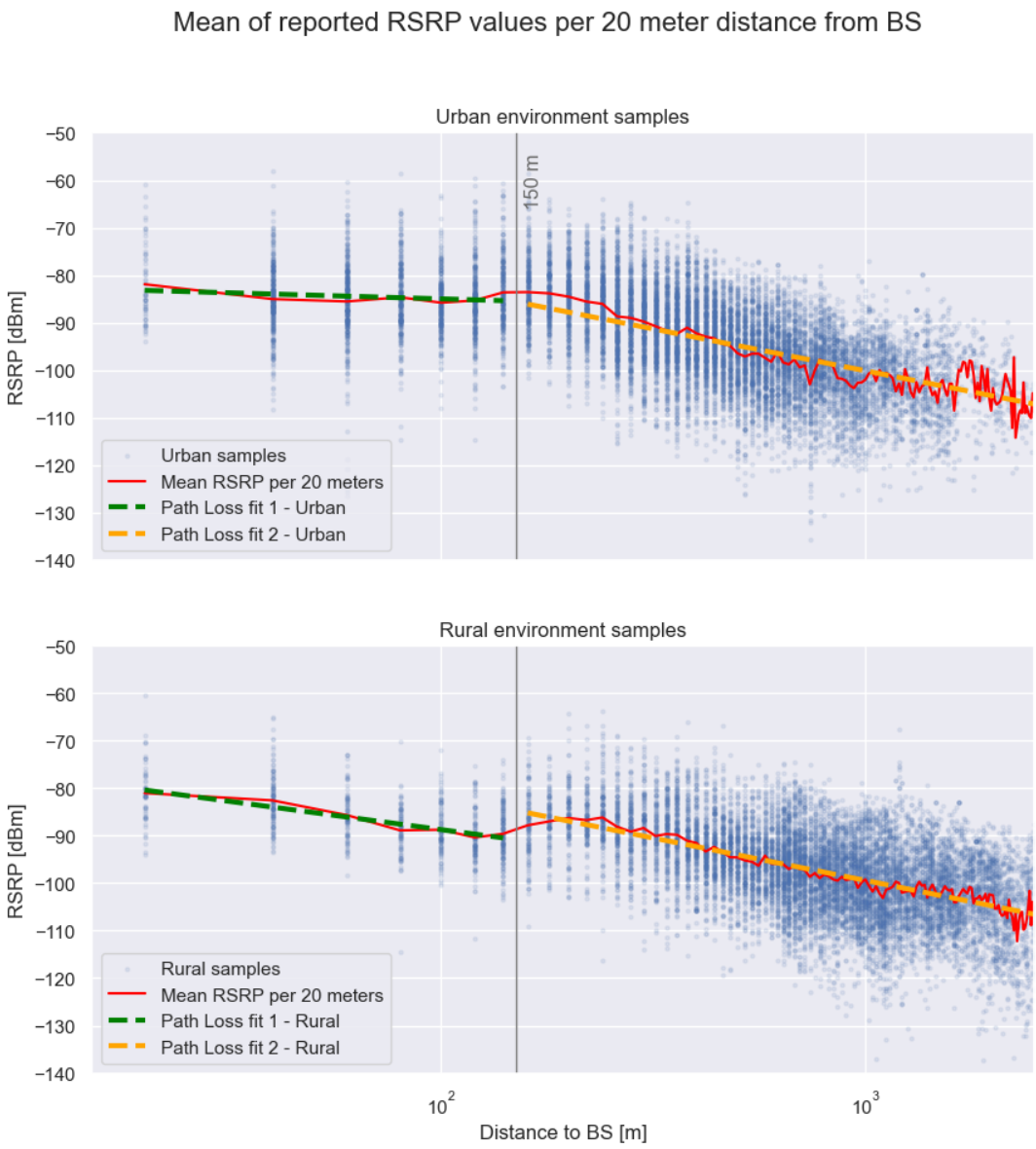


Figure 5.7: Path Loss extraction from the classified data

property of Shadow Fading that we have not tackled yet. That is the spatial correlation distance of the Shadow Fading component. In order to infer this quantity, we will leverage Gaussian Processes.

In chapters 2 and 3 of this thesis, we have discussed extensively the role of the length-scale parameter of a GP with an RBF kernel. We have understood that this parameter

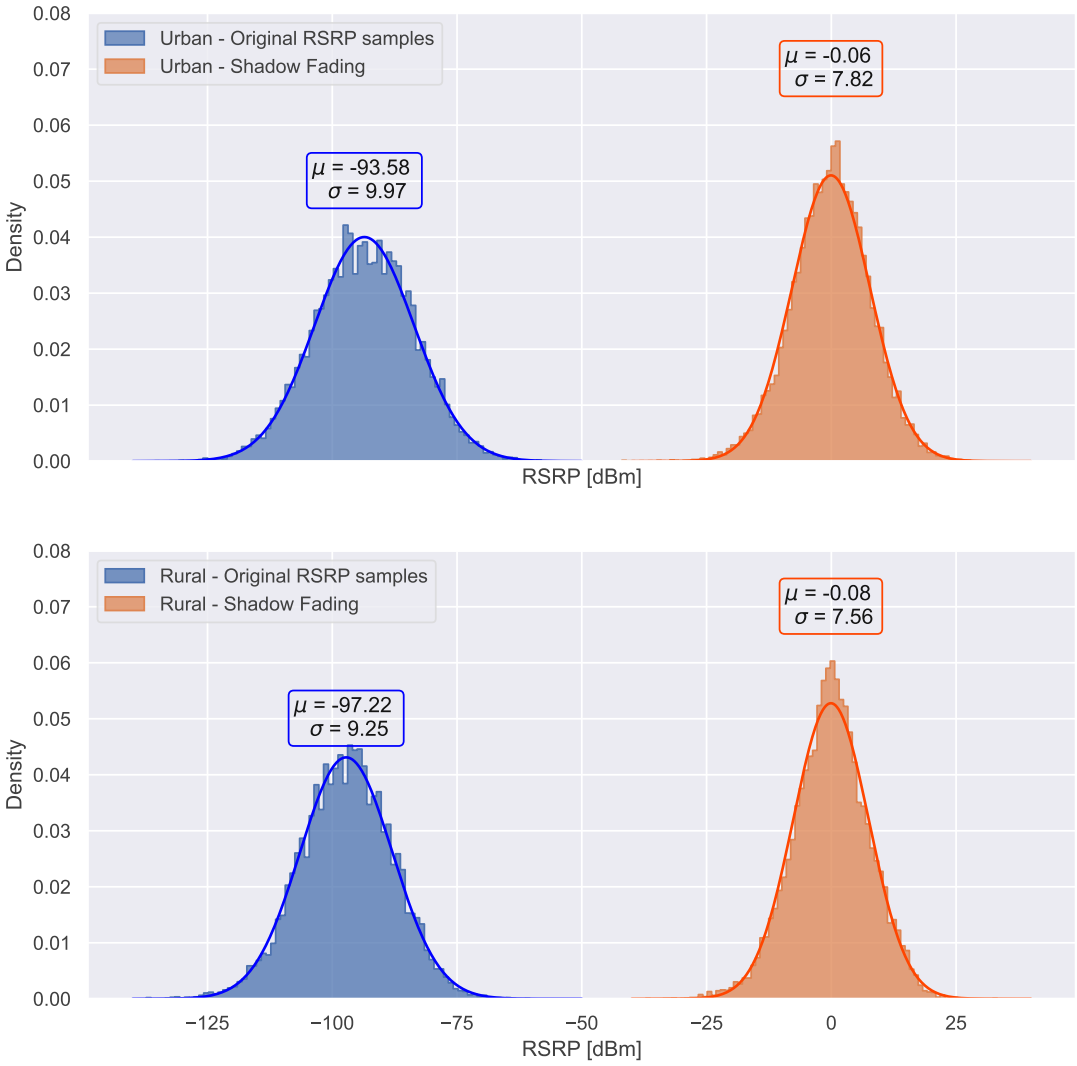


Figure 5.8: Histograms

	Path Loss fit 1		Path Loss fit 2	
	slope - $10n$	intercept - PL_0	slope - $10n$	intercept - PL_0
Urban	-1.88	-80.99	-17.56	-47.41
Rural	-11.95	-64.92	-17.99	-45.51

Table 5.1: Path Loss coefficients

models how quickly the correlation between measurement samples decreases with their increasing Euclidian distance. If we now fit a GP to our extracted Shadow Fading values, through the log marginal likelihood optimization procedure, we will learn the optimal decorrelation distances in our data. The GP fitting procedure was done as follows.

A separate GP was fitted to the data on a cell-by-cell case, meaning we got one GP model, with its learned characteristic length-scale, for each observed PCI. Cells were additionally classified as either Urban or Rural, similarly to the process explained in 5.3.3. The resulting length-scales, classified as either Urban or Rural, are then combined into corresponding Empirical Cumulative Distribution Function (ECDF) as shown in Figure 5.9.

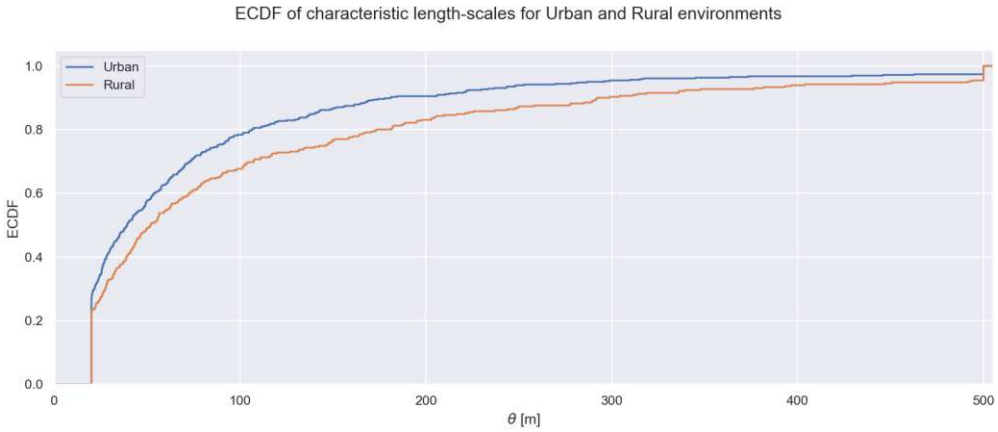


Figure 5.9: ECDF of characteristic length-scales for Urban and Rural environments

It is important to note that, during the training procedure, the length-scales were constrained to be between 20 and 500 meters. The lower bound, 20 meters, is necessary because of the inherent location uncertainty that stems from the measurement setup. If we go back to Section 5.2.1, we will remember we had a centralized GPS system, collapsing our train cabin into a single moving point. In addition, at high speeds, the measurement positions are rather dispersed in space. RSRP values were measured approximately every $500ms$. In areas where the train was moving at full speed, at over $200\frac{km}{h}$, which is around $55\frac{m}{s}$, we will only have a couple of measurement samples per 55 meter distance from our five phone fleet. If the correlation distances are much shorter than our sampling distance, we could not observe them anyway, similar to the discussion arising from Figure 2.6. For safe measure, we do not allow for a GP to learn correlation distances shorter than 20 meters, as in the opposite case it could potentially overfit to each measurement position. The upper bound, 500 meters, is intentionally chosen to be higher than the expected maximum decorrelation distance.

We have reached several conclusions concerning the statistics of the length-scale parameters based on the presented data. Firstly, we observe a tail extending to considerable decorrelation distances, up to 500 meters. However, it is plausible that most of these outcomes are not physically meaningful but rather a result of the GP fitting to the data as if it were purely noise. This suggests that the GPR model, or our data, fail to capture any significant correlations within the observed Shadow Fading component. A similar discussion was provided in chapter 2, pertaining to Figure 2.6. Conversely, the

majority of the data aligns with length-scale parameters within the range of 100 meters. Another intriguing finding is the higher characteristic decorrelation distances in the Rural environment. This aligns with expectations, as the Rural environment typically features more uniform terrain, encompassing large flat areas or extensive areas shadowed by hills and mountains.

Knowledge of these characteristic length scales can provide several advantages. Firstly, it allows for accurate modeling of the propagation environment in railway scenarios, enabling the extraction of statistical properties from the measured data. Secondly, it relates to the discussion on SGPR. As previously explored in chapter 3, the number of measurement samples required for reliable map prediction depends on the decorrelation distance of the underlying process. By understanding the typical decorrelation distances prevalent in different radio environments, we can estimate the number of data points needed for dependable radio map prediction using SGP. This approach is not restricted solely to railway scenarios but extends to various radio performance maps, such as those for city street grids, large airports, open fields, and more.

CHAPTER 6

Conclusion and Outlook

The primary focus of this thesis were Gaussian Processes (GPs), their various aspects and applications. In the first chapter, we provided a comprehensive overview of GPs, emphasizing the fundamental concepts and highlighting the benefits they offer for tasks such as classification, interpolation, and regression. Our attention then shifted toward regression analysis, specifically within the context of modeling spatial quantities. A significant portion of our work involved the development of a mathematical model for Gaussian Process Regression (GPR). We dedicated considerable discussion to the role of the covariance function and the kernel hyperparameters, with a specific emphasis on the widely-used Radial Basis Function kernel. By consistently employing the RBF kernel throughout the chapter, we aimed to develop an intuitive understanding of each component in the GPR model. This intuitive grasp of the model's components was one of the primary objectives, and we hope to have achieved it successfully.

In the second chapter of our thesis, we tackled one of the primary challenges associated with Gaussian processes, namely their computational complexity. To address this issue, we introduced a set of techniques known as sparse Gaussian processes, which aim to reduce the computational burden while maintaining reasonable predictive performance. We developed mathematical models for three main sparse methods: Subset of Data, Projected Process, and variational learning. All of these proposed models rely on selecting a subset of data points from the input location space, commonly referred to as the active set. To evaluate the effectiveness of each model, we conducted benchmarking experiments, aiming to understand how much the more complex models could enhance predictive capabilities. It became evident that the number of points included in the active set significantly influenced the accuracy of our predictions. Generally, when a sufficient number of points were included in the active set, most of the methods exhibited comparable and satisfactory performance. Furthermore, we recognized that the required number of inducing inputs in the active set depended on the ratio between the true length-scale parameter of the RBF

kernel and the area of interest. With smaller length-scale parameters, and faster varying underlying functions, a larger number of active points is needed for accurate predictions.

Subsequently, we explored the diverse possible applications of GPs in the domain of wireless networks. We introduced several key areas within the wireless communications domain, highlighting their importance both from an industry perspective and as possible subjects of further research. We presented an overview of the current state-of-the-art in these areas, specifically focusing on successful implementations of GPs. By examining existing literature and real-world applications, we aimed to showcase the potential of GPs in advancing wireless network technologies. Looking ahead, the intention is to delve deeper into the collected topics and leverage the knowledge and insights on GPs gained from this thesis.

In the final chapter of this thesis, we diverged from the topic of GPs to address a unique solution for wireless coverage onboard trains. Given that trains are predominantly constructed with metal materials, including metal-coated window panes to mitigate heat dissipation, the train cabin causes significant losses to impending electromagnetic radiation. To overcome this challenge, a novel solution is presented, one that involves Modified Window panes featuring laser-cut microgrids in the metal coating. This modified coating allows electromagnetic radiation to pass through while preserving its thermal insulation properties. In this final chapter, we documented a measurement campaign conducted within a train cabin equipped with these Modified Windows along the Vienna-Innsbruck route. Initially, we introduced fundamental concepts and effects related to wireless propagation. Subsequently, we discussed the findings derived from the measured received power levels in relation to the discussed propagation effects. We developed a simple Path Loss model and delved into the statistical properties of the extracted Shadow Fading values. Lastly, we presented an approach utilizing GPs to estimate spatial correlations between the obtained measurements. We discussed the potential of SGPs, enabling accurate predictions despite limited data points. Overall, this chapter expanded the scope of the thesis to explore a specific problem in wireless coverage onboard trains, while highlighting the potential of GPs in addressing the wireless coverage analysis.

List of Figures

2.1 A linear parametric model $a_{true} = 1.4$ $b_{true} = 2.7$ 7
2.2 Bayes' flow 8
2.3 Prior and posterior samples with different kernel choices 9
2.4 RBF kernel shape for different length-scales 15
2.5 RBF kernel shape for two dimensional inputs and $\theta = 1$ 15
2.6 Prediction with an RBF kernel with different length-scale parameters . . 17
2.7 Upcrossings count for two sample functions 18
2.8 Approximation error of the length-scale parameter over different sample lengths 20
2.9 Contour lines of samples generated with different length-scale parameters 21

3.1 Joint optimization of inducing inputs and hyperparameters using variational learning 41
3.2 Latent function map 42
3.3 Comparison of different SGPR strategies 42
3.4 K-Means subsampling with different length-scale parameters 43

5.1 Current solutions for wireless connectivity in trains 53
5.2 Measurement device - a Samsung phone with NEMO software 54
5.3 Railway track connecting Vienna and Innsbruck retrieved from OSM . . . 55
5.4 Mapped BS distance over corresponding reported TAs 59
5.5 CORINE Land use Classes Map 60
5.6 Land use classification of track segments detail 61
5.7 Path Loss extraction from the classified data 63
5.8 Histograms 64
5.9 ECDF of characteristic length-scales for Urban and Rural environments . 65

Acronyms

1D one-dimensional. 20

2D two-dimensional. 20, 43, 44

3D three-dimensional. 57, 61

AAF Amplify-and-Forward. 52

ADMM Alternating Direction Method of Multipliers. 46

AoA Angle of Arrival. 53

BCM Bayesian Committee Machine. 25

BLER Block Error Ratio. 48

BS Base Station. 48, 57, 58, 61, 62

CIR Carrier-to-Interference Ratio. 50

CLC Corine Land Cover. 59, 60

CQI Channel Quality Indicator. 48, 49, 54

DNN Deep Neural Networks. 46, 47

DTC Deterministic Training Conditional. 25

ECDF Empirical Cumulative Distribution Function. 65

ELBO Evidence Lower Bound. 36

GP Gaussian Process. 8–10, 13–16, 19, 20, 28, 35, 36, 40, 42–46, 49, 63–65

GP-LVM Gaussian Process Latent Variable Model. 47

GP-MPC Gaussian Process Model Predictive Control. 47

GPR Gaussian Process Regression. 1–3, 6, 9, 10, 13, 31, 33, 47–49, 51, 65, 67

GPS Global Positioning System. 54–56, 61, 65

GPs Gaussian Processes. 1–3, 6–8, 14, 16, 45, 51, 67, 68

IoT Internet of Things. 2, 46

IVM Informative Vector Machine. 28, 34

KL Kullback-Leibler. 27, 29, 35, 36

KNN K-Nearest Neighbors. 49

KPI Key Performance Indicator. 54

KPIs Key Performance Indicators. 54

LOS Line-of-Sight. 57, 58

LSTM Long Short-Term Memory. 46

LTE Long Term Evolution. 51, 54

MAP Maximum A Posteriori. 8

MES Maximum Entropy Sampling. 31, 33, 34, 41, 43

MIMO Multiple Input Multiple Output. 52

MMSE Minimum Mean Square Error. 8

MNO Mobile Network Operator. 48, 57

MSE Mean Squared Error. 8, 24, 41, 44

ND N-dimensional. 20

NLOS Non-Line-of-Sight. 57

NP Non-deterministic Polynomial-Time. 32

OSM OpenStreetMap. 54, 55, 60

PCI Physical Cell Identifier. 57, 58, 65

PL Path Loss. 49, 56, 57, 59, 62

PP Projected Process. 25, 27, 28, 33, 35–37, 41, 67

QoS Quality of Service. 45, 46

RBF Radial Basis Function. 9, 14, 15, 18, 43, 44, 63, 67

RL Reinforcement Learning. 47

RSRP Reference Signal Receive Power. 51, 54, 56, 57, 60, 62, 65

RSS Received Signal Strength. 46, 47

RSSI Received Signal Strength Indicator. 49

SD Subset of Data. 25, 41, 43, 67

SGP Sparse Gaussian Process. 23, 24, 29–31, 33, 66

SGPR Sparse Gaussian Process Regression. 25, 28, 29, 33, 38, 40, 41, 43, 51, 66

SGPs Sparse Gaussian Processes. 2, 68

SIMO Single Input Multiple Output. 52

SLAM Simultaneous Localization and Mapping. 47

SR Subset of Regressors. 25, 27

SVMs Support Vector Machines. 14

TA Timing Advance. 58

UAV Unmanned Aerial Vehicle. 2, 47

UAVs Unmanned Aerial Vehicles. 47, 48

UE User Equipment. 48, 54, 57, 58, 61

VPL Vehicle Penetration Loss. 52, 53

ÖBB Österreichische Bundesbahnen. 54

Bibliography

[1] D. G. Krige, *A Statistical Approach to some Mine Valuation and Allied Problems on the Witwatersrand*, 1951.

[2] D. J. MacKay *et al.*, "Introduction to Gaussian Processes", *NATO ASI series F computer and systems sciences*, vol. 168, pp. 133–166, 1998.

[3] C. K. Williams *et al.*, *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2006, vol. 2.

[4] J. Kaur *et al.*, "Machine Learning Techniques for 5G and Beyond", *IEEE Access*, vol. 9, pp. 23 472–23 488, 2021. DOI: 10.1109/ACCESS.2021.3051557.

[5] M. E. Morocho-Cayamcela *et al.*, "Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential, Limitations, and Future Directions", *IEEE Access*, vol. 7, pp. 137 184–137 206, 2019. DOI: 10.1109/ACCESS.2019.2942390.

[6] Ericsson, *Number of smartphone mobile network subscriptions worldwide from 2016 to 2022, with forecasts from 2023 to 2028 (in millions) [Graph]*, [Online; accessed June, 2023], 2022. [Online]. Available: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[7] Ericsson, *Average monthly smartphone traffic worldwide from 2012 to 2028 (in exabytes) [Graph]*, [Online; accessed June, 2023], 2022. [Online]. Available: https://www.statista.com/statistics/739002/worldwide-smartphones-monthly-data-traffic/.

[8] K. P. Murphy, *Machine Learning: a Probabilistic Perspective*. MIT press, 2012.

[9] R. J. Adler, *The Geometry of Random Fields*. SIAM, 2010.

[10] R. J. Adler, "Excursions Above a Fixed Level by N-dimensional Random Fields", *Journal of Applied Probability*, vol. 13, no. 2, pp. 276–289, 1976.

[11] J. Quinonero-Candela *et al.*, "A Unifying View of Sparse Approximate Gaussian Process Regression", *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[12] G. Wahba, *Spline Models for Observational Data*. SIAM, 1990.

[13] T. Poggio *et al.*, "Networks for Approximation and Learning", *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[14] V. Tresp, "A Bayesian Committee Machine", *Neural computation*, vol. 12, no. 11, pp. 2719–2741, 2000.

[15] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes", in *Artificial intelligence and statistics*, PMLR, 2009, pp. 567–574.

[16] M. Seeger, "Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations", University of Edinburgh, Tech. Rep., 2003.

[17] L. Csató *et al.*, "Sparse On-Line Gaussian Processes", *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[18] N. Lawrence *et al.*, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine", *Advances in Neural Information Processing Systems*, vol. 15, 2002.

[19] G. Gan *et al.*, *Data Clustering: Theory, Algorithms, and Applications*. SIAM, 2020.

[20] S. Tripkovic, "Construction of Mobile Performance Maps using Clustered Crowdsourced Measurements", Ph.D. dissertation, Wien, 2020.

[21] S. Lloyd, "Least Squares Quantization in PCM", *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[22] C. M. Bishop *et al.*, *Pattern Recognition and Machine Learning*. Springer, 2006, vol. 4.

[23] D. Arthur *et al.*, "K-means++ the Advantages of Careful Seeding", in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 1027–1035.

[24] M. C. Shewry *et al.*, "Maximum Entropy Sampling", *Journal of Applied Statistics*, vol. 14, no. 2, pp. 165–170, 1987.

[25] C.-W. Ko *et al.*, "An Exact Algorithm for Maximum Entropy Sampling", *Operations Research*, vol. 43, no. 4, pp. 684–691, 1995.

[26] D. M. Blei *et al.*, "Variational Inference: A Review for Statisticians", *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[27] M. K. Titsias, "Variational Model Selection for Sparse Gaussian Process Regression", *Report, University of Manchester, UK*, 2009.

[28] J. Hensman *et al.*, *Derivation of SGPR Equations*. [Online]. Available: `https://gpflow.readthedocs.io/en/v1.5.1-docs/notebooks/theory/SGPR_notes.html`.

[29] Y. Xu *et al.*, "Wireless Traffic Prediction with Scalable Gaussian Process: Framework, Algorithms, and Verification", *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1291–1306, 2019.

[30] Y. Xu *et al.*, "High-Accuracy Wireless Traffic Prediction: A GP-Based Machine Learning Approach", in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, 2017, pp. 1–6.

[31] W. Wang *et al.*, "Cellular Traffic Load Prediction with LSTM and Gaussian Process Regression", in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*, IEEE, 2020, pp. 1–6.

[32] J. Kim *et al.*, "Gaussian Process Regression-Based Traffic Load Balancing for Multimedia Multipath Systems", *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 1211–1223, 2019.

[33] F. Yin *et al.*, "Distributed Recursive Gaussian Processes for RSS Map Applied to Target Tracking", *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 3, pp. 492–503, 2017.

[34] Z. Xu *et al.*, "Online Radio Map Update Based on a Marginalized Particle Gaussian Process", in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 4624–4628.

[35] Z. He *et al.*, "Calibrating Multi-Channel RSS Observations for Localization Using Gaussian Process", *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1116–1119, 2019.

[36] B. Ferris *et al.*, "WiFi-SLAM Using Gaussian Process Latent Variable Models", in *IJCAI*, vol. 7, 2007, pp. 2480–2485.

[37] E. Yel *et al.*, "GP-Based Runtime Planning, Learning, and Recovery for Safe UAV Operations under Unforeseen Disturbances", in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 2173–2180.

[38] D. Jang *et al.*, "Networked Operation of a UAV Using Gaussian Process-Based Delay Compensation and Model Predictive Control", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9216–9222.

[39] A. A. Meera *et al.*, "Obstacle-Aware Adaptive Informative Path Planning for UAV-Based Target Search", in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 718–724.

[40] S. Homayouni *et al.*, "Gaussian Process Regression for Feedback Reduction in Wireless Multiuser Networks", in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, IEEE, 2019, pp. 1–5.

[41] S. Homayouni *et al.*, "Reducing CQI Feedback Overhead by Exploiting Spatial Correlation", in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, IEEE, 2018, pp. 1–5.

[42] A. Chiumento *et al.*, "Adaptive CSI and Feedback Estimation in LTE and Beyond: a Gaussian Process Regression Approach", *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, pp. 1–14, 2015.

[43] K. J. Jang *et al.*, "Path Loss Model Based on Machine Learning Using Multi-Dimensional Gaussian Process Regression", *IEEE Access*, vol. 10, pp. 115 061–115 073, 2022.

[44] J. Riihijarvi *et al.*, "Machine Learning for Performance Prediction in Mobile Cellular Networks", *IEEE Computational Intelligence Magazine*, vol. 13, no. 1, pp. 51–60, 2018.

[45] L. Eller *et al.*, "Propagation-Aware Gaussian Process Regression for Signal-Strength Interpolation along Street Canyons", in *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, IEEE, 2021, pp. 1–7.

[46] M. Lerch *et al.*, "Distributed Measurements of the Penetration Loss of Railroad Cars", in *2017 IEEE 86th VTC-Fall*, pp. 1–5.

[47] Prose Technologies India Pvt. Ltd., *LTE MIMO with In-Train Repeaters*, [Online; accessed June, 2023], ca. 2021. [Online]. Available: `https://www.prosetechnologies.com/solutionDetail.html?id=217`.

[48] Alain Herzog, *Laser engraved microscopic grid on the train cabin window metal coating*, [Online; accessed June, 2023], 2022. [Online]. Available: `https://www.techexplorist.com/making-window-panes-railcars-permeable-mobile-phone-signals/44583/`.

[49] Keysight Technologies, *Nemo Handy Handheld Measurement Solution*, `https://www.keysight.com/us/en/product/NTH00000B/nemo-handy-handheld-measurement-solution.html`, Accessed: 2022.

[50] OpenStreetMap contributors, *OpenStreetMap*, `https://www.openstreetmap.org`, 2017.

[51] A. F. Molisch, *Wireless Communications*. John Wiley & Sons, 2012.

[52] T. ETSI, "138 901 V14. 3.0: 5G; Study on channel model for frequencies from 0.5 to 100 GHz, 3GPP Std", 3GPP TR 38.901 version 14.3. 0 release 14, Tech. Rep., 2018.