Energy-Efficient Navigation of Electric Vehicles using Gaussian Processes

A Contextual Multi-Armed Bandit Approach for Online Learning

Master's thesis in Engineering Mathematics and Computational Science

Jack Sandberg

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Energy-Efficient Navigation of Electric Vehicles using Gaussian Processes

A Contextual Multi-Armed Bandit Approach for Online Learning

Jack Sandberg

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Energy-Efficient Navigation of Electric Vehicles using Gaussian Processes
A Contextual Multi-Armed Bandit Approach for Online Learning
Jack Sandberg

Supervisor: Morteza Haghir Chehreghani, Department of Computer Science and Engineering
Co-supervisor: Niklas Åkerblom, Department of Computer Science and Engineering
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Explored routes in a simulation of the road network of Monaco.

Typeset in LaTeX
Gothenburg, Sweden 2023

Energy-Efficient Navigation of Electric Vehicles using Gaussian Processes
A Contextual Multi-Armed Bandit Approach for Online Learning
Jack Sandberg
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Navigation software that prioritizes energy efficiency could, through simple means, help extend the effective range and adoption rate of electric vehicles (EV).

This thesis extends a previously studied online learning framework using Bayesian inference to find energy-efficient routes. In the extended framework, the characteristics of the road segments are combined with observed energy consumption data to provide probabilistic energy consumption estimates using Gaussian processes (GP). The GP uses a graph Matérn kernel extended from [1] and a feature kernel to model the correlation of energy consumption on separate road segments.

The framework is applied to a simple synthetic road network and real-world road networks in the traffic simulator SUMO. The results demonstrate that the GP learns more efficiently in the networks considered than in the Bayesian inference method. Furthermore, we investigate how the GP method is impacted by the number of inducing points, heteroskedastic noise modeling, an informative prior, and the choice of bandit algorithm.

Keywords: Gaussian processes, Electric vehicles, Online learning, Multi-armed bandits, Energy efficient navigation.

Acknowledgements

I would like to express my gratitude to my supervisor Morteza Haghir Chehreghani and co-supervisor Niklas Åkerblom for their guidance and advice during the thesis. I would also like to thank my family for their support during the thesis and throughout my education.

Jack Sandberg, Gothenburg, 2023-06-09

Contents

List of Figures

List of Tables

1

Introduction

In this chapter, the motivation and problem formulation of the thesis is introduced and an outline of the thesis is presented.

1.1 Background

Road transports were responsible for 16.2% of global greenhouse gas emissions during 2016 [2]. Battery electric vehicles (BEV) are, and will be, essential to reduce the emission of carbon dioxide in the transportation sector in the coming years. According to estimates from the International Energy Agency [3], BEVs could have a market share of new car sales of 25-35% in 2030 and would be one of the leading contributors to the growth in electricity demand. Compared to internal combustion engine vehicles (ICEVs), BEVs tend to have a lower range and access to fewer charging stations. Drivers often refer to the low range as a reason not to purchase, or lease, a BEV as they fear it would limit where they can travel, a phenomenon referred to as *range anxiety* [4].

Increasing the battery capacity and expanding charging networks are important methods for increasing the range of BEVs. An additional method is to increase the efficiency of the vehicle by choosing energy-efficient routes and thereby extending its effective range. Navigation tools that effectively increase the energy efficiency of BEVs and function for any vehicle in any city could remarkably affect BEV's adoption rate and decrease global energy consumption.

Previous work on efficient navigation of electric vehicles has focused on combining powertrain simulations with large-scale datasets [5]–[7] or assumed the energy consumption is known from a static model [8], [9]. In [10], Åkerblom *et al.* introduced an online learning framework for energy-efficient navigation. In an online setting, the goal is to learn efficient routes using as few data points as possible. In comparison, for an offline environment, a fixed dataset is typically provided, and the goal is to maximize relevant metrics using all available data.

Applying a data-inefficient offline algorithm to a new city potentially requires collecting large amounts of new data to ensure good predictions. In comparison, an efficient online learning algorithm could be deployed to road networks with no currently available data and would adapt quickly to find efficient routes.

The online learning framework presented in [10] treats each road segment in a network independently, leading to a simple and effective algorithm. However, the assumption of independence necessitates collecting additional data as the model cannot confidently estimate the energy consumption of road segments without driving over them. By instead assuming that the energy consumption of the segments is correlated and depends on the characteristics of the road, is it possible to more effectively use the collected data to find energy-efficient routes?

This thesis seeks to answer the above question using Gaussian processes (GP). In this thesis, GPs provide a probabilistic method of estimating energy consumption and correlation between edges. GPs use kernel functions to measure similarities and correlations between inputs. The kernel function's design impacts the GP's properties and must be selected carefully. GPs allow us to embed prior information into the model before observing any data, allowing the model to make more accurate estimates when using a small dataset.

In most GP applications, the estimated quantity is assumed to have constant variance across the input space, or rather the noise is said to be homoskedastic. For energy consumption, it is natural that the variance should differ across the network, which motivates heteroskedastic modeling of the noise. However, heteroskedastic modeling adds additional complexity with no guarantee of improving the estimates.

A fundamental problem of using GPs for large datasets is the cubic complexity of computing exact estimates. Approximate GPs provide faster but less accurate estimates. The most common approximate GP method, SVGP [11], uses *inducing points* to summarize the dataset. The predictive quality of the SVGP is dependent on the number of inducing points.

To select the data collection routes, [10] compares two bandit algorithms, Thompson sampling and Upper Confidence Bounds. While [10] finds a small benefit for Thompson sampling, this does not necessarily hold for GP models.

1.2 Problem formulation

The aim of the thesis is to investigate how GP methods can be used to increase the data efficiency of the online learning framework for energy-efficient navigation of electric vehicles introduced in [10].

To understand how to design the GP for this problem, we will further investigate the following questions:

Q1. What is an effective kernel function for the GP?

Q2. What is the impact of the number of inducing points on the GP?

Q3. What is the impact of homoskedastic versus heteroskedastic modeling of the noise?

Q4. What is the impact of informative priors on the GPs' performance and exploration?

Q5. Does the choice of bandit algorithm impact the performance of the GP?

To answer the questions above, the online learning framework of [10] is extended to use contextual information about the road segments. The new framework is then applied to synthetic and simulated road networks.

1.3 Outline of thesis

Chapter 2 introduces the energy model, bandit problems and GPs, and how they are used to build an online learning framework.

Chapter 3 describes the implementation of GP models, the road networks, and general methodology.

Chapter 4 presents and visualizes the results from the experiments.

Chapter 5 discusses the results and methodology, and provides a final conclusion for the thesis.

2

Theory

In this chapter, we introduce the theoretical background required to understand the thesis. In Section 2.1, we formulate the energy model and frame the online shortest path problem as a contextual and combinatorial bandit problem. Then, in Section 2.2 we introduce the theory of Gaussian processes.

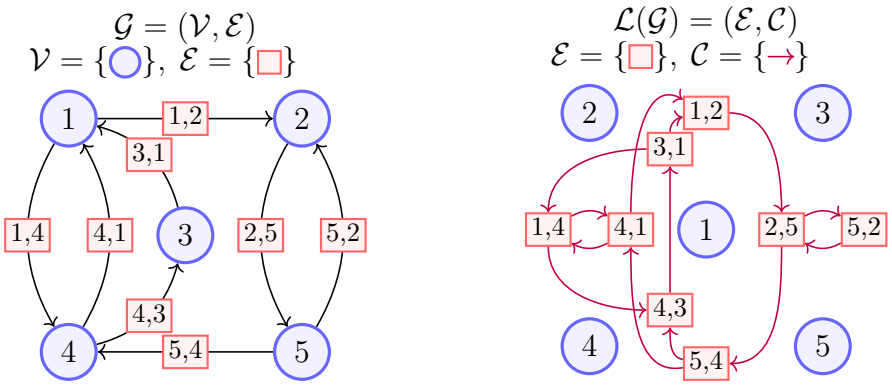
This chapter assumes basic familiarity with Bayesian inference and the multivariate normal distribution. Readers unfamiliar with these two topics are encouraged to read Appendix A for a brief introduction.

2.1 Energy model and learning shortest paths as a bandit problem

In this section, we introduce an energy model for electric vehicles, an online framework for learning the energy consumption and bandit problems.

2.1.1 Energy model

The road network in which the vehicle navigates has two representations, visualized in Figure 2.1. The first representation is a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{w})$ where the vertices $\mathcal{V}$ denote intersections of road segments and the edges $e = (u_1, u_2) \in \mathcal{E}$



(a) The directed graph $\mathcal{G}$.

(b) The line graph $\mathcal{L}(\mathcal{G})$.

Figure 2.1: Two representations of a road network.

denote the travel from intersection u_1 to intersection u_2. Each edge has an associated weight w_e representing the energy consumption when traveling along edge $e \in \mathcal{E}$. In addition to the weights $\boldsymbol{w}$, we associate a set of features $\mathcal{X}_e$ with each edge $e \in \mathcal{E}$. The purpose of the features in $\mathcal{X}_e$ is to aid in determining the energy consumption w_e.

The second representation is the line graph $\mathcal{L}(\mathcal{G}) = (\mathcal{E}, \mathcal{C}, \boldsymbol{w}_{\mathcal{L}})$ where we additionally introduce the set of connections $\mathcal{C}$ between directed edges. The connections $\mathcal{C}$ determine which turns are legal in the road network: $\mathcal{C} \subseteq \{(e_1, e_2) | e_1 = (u, v) \in \mathcal{E}, e_2 = (v, w) \in \mathcal{E}\}$. The weight $w_{\mathcal{L},e_1,e_2}$ in the line graph we choose to be the energy consumption when traveling along edge e_1, thus it is equivalent to w_{e_1}. This choice means that the shortest path from $e = (u, v) \in \mathcal{E}$ to $\tilde{e} = (\tilde{u}, \tilde{v}) \in \mathcal{E}$ in $\mathcal{L}(\mathcal{G})$ is roughly equivalent[1] to the shortest path from u to $\tilde{u}$ in $\mathcal{G}$.

The line graph provides a more accurate model for how the vehicle can navigate and will be useful for defining a GP over the edges of $\mathcal{G}$. Out of convenience, we will generally refer to $\mathcal{G}$ as the road network but keep in mind that $\mathcal{L}(\mathcal{G})$ is the more accurate representation.

The energy consumption of driving along a road segment is stochastic in practice. As in [10], we model it using a Bayesian approach in order to incorporate both the prior knowledge and the observations from the environment. The prior knowledge consists of a simplified deterministic model for the energy consumption. The deterministic model assumes that the vehicle drives along an edge $e \in \mathcal{E}$ of length l_e, with inclination α_e and at the constant speed v_e:

$$E_e^{\text{det}} := \frac{mgl_e \sin(\alpha_e) + mgC_r l_e \cos(\alpha_e) + 0.5 C_d A \rho l_e v_e^2}{3600\eta}. \tag{2.1}$$

The deterministic energy consumption E_e^{det} in Equation (2.1) is given in Watt-hours and depends on the vehicle-specific parameters: mass m, rolling resistance C_r, front surface area A, air drag coefficient C_d and powertrain efficiency η. The gravitational acceleration g and air density ρ also determine E_e^{det}.

A benefit of electric vehicles is their ability to recuperate energy by using *regenerative braking*. With regenerative braking, the energy consumption of an electric vehicle can be negative along individual road segments which presents some challenges when we wish to find the most energy-efficient path. The most common shortest path algorithm, Djikstra's algorithm [12], does not permit negative edge weights. Whilst alternative shortest path algorithms, such as Bellman-Ford [13]–[15], allow negative edge weights, they are significantly slower and do not return a path if the graph has a reachable negative cycle. In fact, for graphs with negative cycles, the problem of finding the shortest simple path is NP-hard [16]. To avoid the complexity associated with negative weights, we use the rectified normal distribution (see Appendix A.3) to get non-negative energy consumption estimates.

Algorithm 1 Online learning for energy-efficient navigation using contextual information

Require: $(\boldsymbol{\mu}_0, \boldsymbol{\varsigma}_0, \boldsymbol{\sigma}_0) \in \mathbb{R}^{|\mathcal{E}|} \times \mathbb{R}_+^{|\mathcal{E}|}$ - prior mean and standard deviations for the energy consumption along the edges $\mathcal{E}$, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{|\mathcal{E}| \times d}$ - matrix of edge features $\mathcal{X}_e$, $e \in \mathcal{E}$.

1: **for** $t \leftarrow 1, \ldots, T$ **do**
2: $\quad \boldsymbol{w}_t \leftarrow \text{GetEdgeWeights}(t, \boldsymbol{\mu}_{t-1}, \boldsymbol{\varsigma}_{t-1}, \boldsymbol{\sigma}_{t-1})$
3: $\quad \boldsymbol{a}_t \leftarrow \text{SolveOptimizationToFindAction}(\boldsymbol{w}_t)$
4: $\quad \boldsymbol{r}_t \leftarrow \text{ApplyActionAndObserveReward}(\boldsymbol{a}_t)$
5: $\quad \boldsymbol{\mu}_t, \boldsymbol{\varsigma}_t, \boldsymbol{\sigma}_t \leftarrow \text{UpdateParameters}(\boldsymbol{a}_{1:t}, \boldsymbol{r}_{1:t}, \boldsymbol{\mu}_{1:t-1}, \boldsymbol{\varsigma}_{1:t-1}, \boldsymbol{\sigma}_{1:t-1}, \boldsymbol{\mathcal{X}})$
6: **end for**

2.1.2 Online learning of the energy model

In [10], the authors introduce an online learning framework that balances choosing the optimal route with respect to the current posterior distribution of the weights $\boldsymbol{w}$ and exploring routes with high uncertainty. Algorithm 1 provides a high-level overview of the framework in [10] with slight modifications.

The idea is that we first compute or sample a set of weights $\boldsymbol{w}_t$ given the current posterior distribution parameters $(\boldsymbol{\mu}_{t-1}, \boldsymbol{\varsigma}_{t-1}, \boldsymbol{\sigma}_{t-1})$. The vectors $\boldsymbol{\mu}_{t-1}$ and $\boldsymbol{\varsigma}_{t-1}$ denote the mean and standard deviation of the posterior of $\boldsymbol{\mu} := \mathbb{E}[\boldsymbol{w}]$ whilst the vector $\boldsymbol{\sigma}_{t-1}$ denotes the posterior mean of the standard deviation $\boldsymbol{\sigma} := \sqrt{\text{Var}[\boldsymbol{w}]}$. Second, we find the optimal route $\boldsymbol{a}_t$ (or action) in $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{w}_t)$ using Djikstra's algorithm with $\boldsymbol{w}_t$ as the weights. Third, we drive along the selected route $\boldsymbol{a}_t$ to observe the *reward* $\boldsymbol{r}_t$ where the reward is the negation of the energy consumption. With this definition, minimizing the energy consumption is equivalent to maximizing the reward. Finally, we update the distribution parameters based on all observations and repeat the process.

In the original online learning framework of [10], the posterior mean of the standard deviation $\boldsymbol{\sigma}_t$ is fixed to the initial values $\boldsymbol{\sigma}_0$. Additionally, the UpdateParameters function accepts only the current action $\boldsymbol{a}_t$, reward $\boldsymbol{r}_t$ and distribution parameters $(\boldsymbol{\mu}_{t-1}, \boldsymbol{\varsigma}_{t-1})$ and no edge features are utilized. Our modification of UpdateParameters gives a GP model (or other possible models) access to the complete history of actions $\boldsymbol{a}_{1:t}$, rewards $\boldsymbol{r}_{1:t}$ and all the edge features $\boldsymbol{\mathcal{X}}$.

To find effective solutions to implement GetEdgeWeights, we will formulate the online learning framework as a bandit problem in Section 2.1.3.1.

2.1.2.1 Bayesian inference for online learning of energy consumption

In this section, we introduce the Bayesian inference method used in [10] to learn the distribution of the energy consumption in each road segment. The key assumption is that the energy consumption, $E_e \in \mathbb{R}$, of an electric vehicle driving along the road

[1] Under the reasonable assumptions that we can only make legal turns and will continue driving on $\tilde{e}$.

Algorithm 2 Gaussian parameter update of the energy model

procedure UPDATEPARAMETERS($\boldsymbol{a}_{1:t}, \boldsymbol{r}_{1:t}, \boldsymbol{\mu}_{1:t-1}, \boldsymbol{\varsigma}_{1:t-1}, \boldsymbol{\sigma}_{1:t-1}, \boldsymbol{\mathcal{X}}$)
  for each edge $e \in \boldsymbol{a}_t$ **do**
    $\varsigma^2_{e,t} \leftarrow \left(\varsigma^{-2}_{e,t-1} + \sigma^{-2}_{e,t-1}\right)^{-1}$
    $\mu_{e,t} \leftarrow \varsigma^2_{e,t}\left(\mu_{e,t-1}\varsigma^{-2}_{e,t-1} + r_e\sigma^{-2}_{e,t-1}\right)$
  end for
  $\boldsymbol{\sigma}_t \leftarrow \boldsymbol{\sigma}_{t-1}$
  return $\boldsymbol{\mu}_t$, $\boldsymbol{\varsigma}_t$, $\boldsymbol{\sigma}_t$
end procedure

segment $e \in \mathcal{E}$ is stochastic and follows a Gaussian distribution with unknown mean θ^*_e and known variance σ^2_e. Additionally, we assume that the energy consumption along different edges are independent, $E_e \perp\!\!\!\perp E_{e'}\ \forall e' \in \mathcal{E} \setminus \{e\}$.

The likelihood function of the Bayesian inference model is given by

$$p(E_e|\theta^*_e, \sigma^2_e) := \mathcal{N}(E_e|\theta^*_e, \sigma^2_e). \tag{2.2}$$

Similarly, we use a Gaussian conjugate prior for the mean parameter θ^*_e:

$$p(\theta^*_e|\mu_{e,0}, \varsigma^2_{e,0}) = \mathcal{N}(\theta^*_e|\mu_{e,0}, \varsigma^2_{e,0}) \tag{2.3}$$

where $\mu_{e,0}$ and $\varsigma^2_{e,0}$ is the prior mean and variance respectively. Section 3.4 describes the initialization of the prior parameters.

The conjugate prior provides simple update equations for the posterior parameters $\mu_{e,t}$ and $\varsigma^2_{e,t}$, see Algorithm 2. As discussed in Section 2.1.1, feasibly computing the shortest path requires non-negative edge weights w_e. Therefore, the edge weights w_e are set to $\mathbb{E}[z_e]$ where z_e is distributed as the rectified Gaussian $\mathcal{N}^R(\theta^*, \sigma^2_e)$, see Appendix A.3 for additional information about the rectified Gaussian distribution.

2.1.3 Bandit problems

The K-armed bandit problem, or multi-armed bandit problem, is a classical problem in which an agent repeatedly has to choose between K available actions. After selecting an action, the agent receives a reward from an (to the agent) unknown probability distribution associated with the chosen action. The goal of the agent is to maximize its expected total reward over a certain time horizon - either finite or infinite [17], [18].

An effective agent must balance acting greedy with trying new actions. If the agent only chooses the action it currently believes to be best (i.e. acts *greedily*), it may fail to discover better actions. Similarly, if the agent only acts in a stochastic manner it will fail to achieve a high total reward. This is often referred to as the *exploration-exploitation trade-off*.

The multi-armed bandit (MAB) problem has several extensions that deal with more complicated scenarios. Two extensions of interest are *combinatorial* and *contextual*

extensions. In a combinatorial MAB problem [19], [20], the agent must select a set of actions in each time step instead of a single action. In a contextual MAB problem [21], the agent is provided additional context for each arm that has some correlation with the expected reward. By utilizing the context, an agent can more efficiently learn the expected reward. In the online shortest path problem, the context of an arm (edge) is a feature vector of static and (potentially) dynamic information about the corresponding edge.

In [10], the authors formulate the online shortest path problem as a combinatorial semi-bandit problem where the term *semi* denotes that we observe the reward for each arm in the set of selected arms. In the following section we extend their formulation to a combinatorial and contextual semi-bandit problem. The extended formulation is an instance of the C3-MAB problem introduced in [22].

2.1.3.1 Online shortest path as bandit problem

In this section, we formalize the problem of choosing the most energy-efficient route in the road network $\mathcal{G}(\mathcal{V}, \mathcal{E}, \boldsymbol{w})$ as a bandit problem.

Assume that we are provided a start vertex $u_1 \in \mathcal{V}$ and a goal vertex $u_n \in \mathcal{V}$. Let $\mathcal{P}$ denote the set of legal and simple paths from u_1 to u_n. A path $\boldsymbol{p} = \langle u_1, u_2, \ldots, u_n \rangle$ is legal if all the connections are legal, i.e. $((u_{i-1}, u_i), (u_i, u_{i+1})) \in \mathcal{C}$, $\forall i \in \{2, \ldots, n-1\}$, and $\boldsymbol{p}$ is simple if every vertex is visited at most once.

At each time step t, we initially observe the context $\mathcal{X}_{e,t} \in \mathbb{R}^d$, $\forall e \in \mathcal{E}$. Based on the context we select an action $\boldsymbol{a}_t \in \mathcal{P}$ and observe the corresponding reward (or negated energy consumption) for each edge in the path:

$$\boldsymbol{r}_t = \sum_{e \in \boldsymbol{p}} r_e. \tag{2.4}$$

Each reward r_e is sampled from a distribution with unknown mean and variance. Importantly, we assume that edges with similar context have similar reward distributions. The objective in our bandit problem is to maximize the sum of rewards obtained from our chosen routes $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots\}$.

An important metric to compare algorithms in bandit problems is the *cumulative regret*. Cumulative regret measures the expected difference in reward between the chosen actions and the optimal action. If the expected energy consumption is known and non-negative along each edge, then the optimal route $\boldsymbol{a}^*$ and corresponding expected reward $\boldsymbol{r}^*$ can be computed using a shortest path algorithm. The cumulative regret is then given by

$$\text{Regret(T)} = \sum_{t=1}^{T} \boldsymbol{r}^* - \mathbb{E}[\boldsymbol{r}_t]. \tag{2.5}$$

We choose to formulate the bandit problem with potentially time-varying context because certain features, such as congestion in the network, are time-dependent in real-world scenarios. In simulated experiments, providing the current network congestion is simple but could be complicated in a real-world scenario. Therefore,

Algorithm 3 Thompson sampling

procedure GETEDGEWEIGHTS($t, \boldsymbol{\mu}_{t-1}, \boldsymbol{\varsigma}_{t-1}, \boldsymbol{\sigma}_{t-1}$)
  for each edge $e \in \mathcal{E}$ **do**
    $\tilde{\theta}_e \leftarrow$ Sample from posterior $\mathcal{N}(\mu_{e,t-1}, \varsigma^2_{e,t-1})$
    $w_{e,t} \leftarrow \mathbb{E}[z_e]$ where $z_e \sim \mathcal{N}^R(\tilde{\theta}_e, \sigma^2_{e,t-1})$
  end for
  return $\boldsymbol{w}_t$
end procedure

Algorithm 4 Bayesian Upper Confidence Bounds

procedure GETEDGEWEIGHTS($t, \boldsymbol{\mu}_{t-1}, \boldsymbol{\varsigma}_{t-1}, \boldsymbol{\sigma}_{t-1}$)
  for each edge $e \in \mathcal{E}$ **do**
    $\tilde{\theta}_e \leftarrow Q(\frac{1}{t}, \mathcal{N}(\mu_{e,t-1}, \varsigma^2_{e,t-1}))$
    $w_{e,t} \leftarrow \mathbb{E}[z_e]$ where $z_e \sim \mathcal{N}^R(\tilde{\theta}_e, \sigma^2_{e,t-1})$
  end for
  return $\boldsymbol{w}_t$
end procedure

we restrict ourselves to static features of the network and drop the t in $\mathcal{X}_{e,t}$ in our notation.

2.1.4 Algorithms for bandit problems

In the following sections, we introduce two commonly used bandit algorithms. The methods balance exploration and exploitation in different ways, and we use them to select the weights $\boldsymbol{w}_t$ in Algorithm 1.

2.1.4.1 Thompson sampling

The first bandit algorithm is Thompson sampling [23], [24]. In Thompson sampling, the key idea is to sample from the posterior and select the action based on the samples. If the posterior has a large uncertainty, the chosen actions will essentially be random. As we collect more samples, the posterior variance shrinks, and the chosen actions will become increasingly deterministic. Thus, Thompson sampling can always explore but will reduce its exploration over time if the posterior converges.

In Algorithm 3, we present an adapted version of Thompson sampling, introduced in [10]. Since we require our weights $\boldsymbol{w}_t$ to be non-negative, we let the weight equal the mean of a rectified Gaussian.

2.1.4.2 Bayesian Upper Confidence Bounds

The second bandit algorithm is called Upper Confidence Bound (UCB). The idea of UCB is to overestimate the expected rewards and reduce the overestimation as the predictive variance shrinks or as its often referred to: *optimism in the face of*

uncertainty. UCB algorithms have been applied to a wide variety of problems, and there are a lot of different variants.

In the context of Bayesian optimization with GPs, a variant of UCB is used to select the next evaluation point (or *action*) of an unknown function $f(x)$ [25]. Using the notation of multi-armed bandits, the selected evaluation point is given by

$$\boldsymbol{a}_t = \arg\max_{\boldsymbol{a}} \boldsymbol{\mu}_{t-1}(\boldsymbol{a}) + \sqrt{\beta_t}\boldsymbol{\sigma}_{t-1}(\boldsymbol{a}) \tag{2.6}$$

where $\boldsymbol{\mu}_{t-1}(\boldsymbol{a})$ and $\boldsymbol{\sigma}_{t-1}(\boldsymbol{a})$ is the posterior mean and standard deviation of action $\boldsymbol{a}$ at time $t-1$ whilst β_t is a parameter increasing with t. From the expression, it is clear that actions where either the posterior mean or standard deviation is large will be selected. Unlike Thompson sampling, UCB is deterministic which can be favorable if a predictable algorithm is desired.

In this thesis, we consider a similar variant of UCB called Bayes-UCB [26]. Let $Q(\beta, \lambda)$ denote the quantile function of the distribution λ where Q is defined such that $P(X \leq Q(\beta, \lambda)) = \beta$ for $X \sim \lambda$ and $\beta \in [0, 1]$. If we assume that our posterior distribution of the rewards are univariate Gaussians, then the selected action in Bayes-UCB is given by

$$\boldsymbol{a}_t = \arg\max_{\boldsymbol{a}} Q\left(1 - \frac{1}{t}, \mathcal{N}(\boldsymbol{\mu}_{t-1}(\boldsymbol{a}), \boldsymbol{\sigma}_{t-1}(\boldsymbol{a}))\right). \tag{2.7}$$

In Equation (2.7), we select the action whose posterior $1-1/t$ quantile is the highest.

To apply Bayes-UCB to our online learning framework, we use the lower quantile $1/t$ since our goal is to minimize energy consumption. Additionally, we again use the rectified normal distribution to ensure the computed quantiles are positive. See Algorithm 4 for the implementation in the online shortest path problem.

2.2 Gaussian processes

A Gaussian process (GP) is a generalization of the multivariate Gaussian distribution from a finite collection of random variables to an infinite set of variables [27]. Using GPs, we can sample normally distributed random variables over arbitrary sets. The most common scenario is to consider samples on $\mathbb{R}$. The obtained samples are then one-dimensional functions instead of n-dimensional vectors. In a sense, GPs can be viewed as an infinite-dimensional Gaussian distribution.

The most common application of GPs is to estimate a function that is only known at a few datapoints. The probabilistic nature of GPs allows us to quantify where an estimate is uncertain and where it is certain. The drawback of applying GP methods is their cubic complexity, but many techniques have been devised to address this issue [11], [27], [28].

In the following sections, we will formally define GPs, examine properties of their kernel functions, discuss how to estimate functions with GPs, and discuss advanced methods for scaling GPs to large datasets and complicated problems.

2.2.1 Definition of Gaussian processes

Formally, a GP is a set of random variables, $\mathcal{X}$, such that any finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathcal{X}$ has a joint Gaussian distribution

$$p(\mathbf{f}|\mathbf{X}) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \mathbf{K}) \tag{2.8}$$

where $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$, $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^\top$, $\boldsymbol{\mu} = [m(\mathbf{x}_1), \ldots, m(\mathbf{x}_N)]^\top$ and $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$.

GPs are fully specified by their mean function $m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and covariance function (or kernel function) $k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}')]$. Following the notation of [27], we denote a GP as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{2.9}$$

The kernel function must be symmetric, i.e. $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$, and positive semi-definite. The kernel function is positive semi-definite if for any finite subset $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathcal{X}$ the corresponding covariance matrix $\mathbf{K}$ is positive semi-definite.

The choice of covariance function is an important hyperparameter that imposes regularity conditions on f. For example, we can require the function to be sufficiently smooth or periodic. We illustrate this in Figure 2.2 with the following kernel functions:

$$k_{RBF}(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right), \tag{2.10}$$

$$k_M(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right)^\nu K_\nu\left(\frac{\sqrt{2\nu}\|\mathbf{x} - \mathbf{x}'\|}{\ell}\right), \tag{2.11}$$

$$k_P(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_i \frac{\sin^2\left(\frac{\pi}{p}(x_i - x_i')\right)}{\ell}\right). \tag{2.12}$$

The first kernel, k_{RBF}, is commonly referred to as the *radial basis function* or *squared exponential kernel* and imposes that $f(\mathbf{x})$ is infinitely differentiable ($f \in C^\infty$). The lengthscale parameter $\ell > 0$ controls how quickly the covariance between two points shrinks and consequently the flatness of f. The Matérn kernel, k_M, has an additional parameter $\nu > 0$ that regulates the smoothness of f. If $\nu > k$, then f is k-times differentiable ($f \in C^k$). The functions $\Gamma(\nu)$ and K_ν denote the gamma function and a modified Bessel function respectively [27]. The periodic kernel, k_P, has a period length parameter p that regulates the periodicity of f [29], [30].

In Figure 2.2, we can observe the properties that these kernels impose. The sampled functions of k_{RBF} change smoothly whilst the sampled functions of k_M with $\nu = 5/2$ look more jagged in comparison. The periodic kernel generates functions that are both smooth and periodic.

The lengthscale ℓ used in the three kernels above can be ill-suited if the dimensions of the input $\mathbf{x} \in \mathbb{R}^d$ either have different scales. By replacing $\|\mathbf{x} - \mathbf{x}'\|/\ell$ with $(\mathbf{x} - \mathbf{x}')^\top \operatorname{diag}(\boldsymbol{\ell})^{-1}(\mathbf{x} - \mathbf{x}')$ in Equations (2.10) to (2.12) for $\boldsymbol{\ell} \in \mathbb{R}^d_+$, we obtain the ability

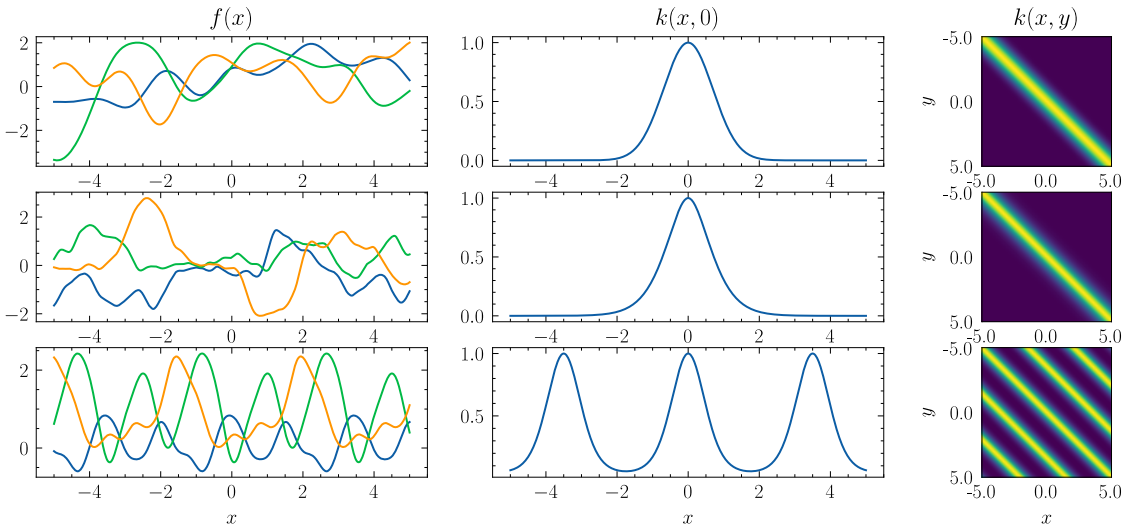


Figure 2.2: Samples drawn from three Gaussian processes (left column) with $m(x) = 0$ but different kernels (middle and right column). From top to bottom, the kernels are a squared exponential kernel, a Matérn kernel, and a periodic kernel.

to scale the dimensions independently. As we will discuss later, the parameters of the kernels can be optimized automatically and using separate lengthscales $\boldsymbol{\ell}$ is often referred to as *automatic relevance determination* (ARD) [27], [31]. Using ARD and kernel parameter optimization, the kernel can learn which dimensions or *features* are relevant.

Another tunable property of the kernels that we have omitted in Equations (2.10) to (2.12) is the output scale σ_f. By multiplying the kernels with an output scale parameter, the magnitude of the covariance can be increased or decreased.

2.2.2 Gaussian process regression

In this section, we explain how GPs can be used to estimate an unknown function $f(\mathbf{x})$.

Let $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$ be N evaluation samples where $\mathbf{x}_i \in \mathcal{X}$ is the sample location and $y_i \in \mathbb{R}$ is a (noisy) evaluation of the unknown function f at the input location $\mathbf{x}_i$. From the samples, we wish to learn the conditional distribution $p(f(\mathbf{x}) \,|\, \mathbf{X}, \mathbf{y})$ where the matrix $\mathbf{X} = (\mathbf{x}_i)_{i=1}^N$ and the vector $\mathbf{y} = (y_i)_{i=1}^N$. We assume that the unknown function $f(\mathbf{x})$ is sampled from a Gaussian process $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$ with $m(\mathbf{x}) \equiv 0$ and that the noise is sampled independently from a Gaussian distribution, i.e. $\mathbf{y} = f(\mathbf{x}) + \boldsymbol{\epsilon}$ where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_y^2 \mathbf{I})$. Additionally, let $\mathbf{f}$ denote the vector of function values $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$.

To obtain a predictive distribution at M new points $\mathbf{X}^* = (\mathbf{x}_i^*)_{i=1}^M$, we consider the joint distribution of $\mathbf{y}$ and $\mathbf{f}^* = [f(\mathbf{x}_1^*), \ldots, f(\mathbf{x}_M^*)]^\top$:

$$\begin{pmatrix} \mathbf{y} \\ \mathbf{f}^* \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{pmatrix} \mathbf{K} + \sigma_y^2 \mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^\top & \mathbf{K}_{**} \end{pmatrix}\right) \tag{2.13}$$

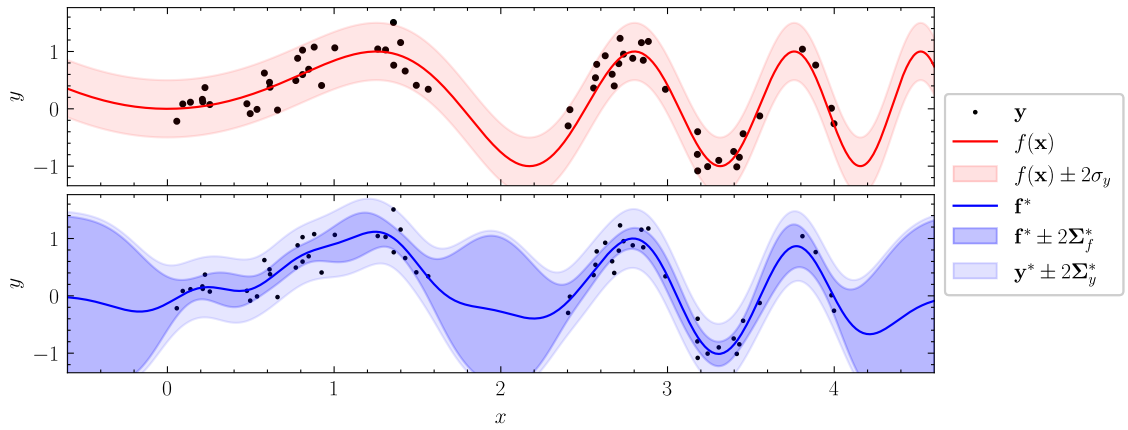


Figure 2.3: Estimation of the function $f(x) = \sin(x^2)$ using Gaussian process regression from 60 noisy observations.

where the covariance matrices are given by $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_* = k(\mathbf{X}, \mathbf{X}^*)$ and $\mathbf{K}_{**} = k(\mathbf{X}^*, \mathbf{X}^*)$. The notation $k(\mathbf{A}, \mathbf{B})$ for $\mathbf{A} \in \mathbb{R}^{N \times P}$, $\mathbf{B} \in \mathbb{R}^{M \times P}$ denotes the application of the covariance function k to all combination of row vectors from $\mathbf{A}$ and $\mathbf{B}$, resulting in a $N \times M$ matrix.

Using Theorem 1, we obtain the posterior predictive distribution $p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{y})$

$$p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{y}) = \mathcal{N}(\mathbf{f}^*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*), \tag{2.14}$$

$$\boldsymbol{\mu}_* = \mathbf{K}_*^\top \mathbf{K}_y^{-1} \mathbf{y}, \tag{2.15}$$

$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^\top \mathbf{K}_y^{-1} \mathbf{K}_* \tag{2.16}$$

where the additional covariance matrix is given by $\mathbf{K}_y = \mathbf{K} + \sigma_y^2 \mathbf{I}$. Note that the predictive distribution requires computing the inverse of a $N \times N$ matrix which has a complexity of $\mathcal{O}(N^3)$. The cubic complexity of matrix inversion makes GP regression infeasible for large datasets. In Section 2.2.3, we explore how to modify the model to avoid computing an expensive inverse.

As we saw in Section 2.2.1, the kernel function k is often specified with a set of hyperparameters $\boldsymbol{\theta}$. A method of estimating the hyperparameters $\boldsymbol{\theta}$ and the variance σ_y^2 is to maximize the log likelihood using gradient ascent algorithms

$$\log p(\mathbf{y}) = \log \left[\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma_y^2 \mathbf{I}) \right]. \tag{2.17}$$

In Figure 2.3, we visualize the results of GP regression based on 60 noisy observations in two separate regions using the squared exponential kernel. In the upper panel, we visualize the underlying function $\sin(x^2)$ along with a 95% confidence region. In the bottom panel, we visualize the predictive mean $\boldsymbol{\mu}^*$ and two 95% confidence regions corresponding to the distributions $p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{y})$ (blue) and $p(\mathbf{y}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{y})$ (lighter blue) respectively. We can note that the confidence region shrinks around the observations but increases outside two regions with observations. The mean prediction generally resembles the underlying function but the noise adds some inaccuracy. Outside the observed intervals, the mean function tends to zero due to our assumption that $m(\mathbf{x}) \equiv 0$.

2.2.3 Sparse Gaussian process regression

As we noted in Section 2.2.2, exact GP regression has a complexity of $\mathcal{O}(N^3)$ which hinders its application as the number of observed datapoints increases. In Sparse GP regression, we use m inducing variables to approximate the exact GP. The idea of the inducing variables is to let a small number of points summarize the data without explicitly choosing the exact placement.

In addition to optimizing the kernel hyperparameters $\boldsymbol{\theta}$ and noise variance σ_y^2, Sparse GPs also optimize the placement of the inducing variables by maximizing a lower bound of the log likelihood $\log p(\mathbf{y})$. In this section, we introduce and explain the variational learning method for Sparse GP introduced by Titsias in [28].

The posterior predictive distribution $p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{y})$ from Equation (2.16) can be rewritten by marginalization using $\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N)]^\top$:

$$p(\mathbf{f}^*|\mathbf{X}^*, \mathbf{X}, \mathbf{y}) = p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}^*|\mathbf{f})p(\mathbf{f}|\mathbf{y})\mathrm{d}\mathbf{f} \tag{2.18}$$

where we now implicitly condition on the inputs $\mathbf{X}$ and $\mathbf{X}^*$ to simplify the notation. Inside the integral, $p(\mathbf{f}^*|\mathbf{f})$ is the posterior of the predictions $\mathbf{f}^*$ conditioned on the latent training variables $\mathbf{f}$ while $p(\mathbf{f}|\mathbf{y})$ is the posterior of the latent variables $\mathbf{f}$ conditioned on the observed values $\mathbf{y}$.

Let $\mathbf{u}$ denote m inducing variables evaluated at the pseudo-inputs $\mathbf{Z} = (\mathbf{z}_i)_{i=1}^m$ and once again rewrite $p(\mathbf{f}^*|\mathbf{y})$ by conditioning on the inducing variables $\mathbf{u}$:

$$p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}^*|\mathbf{u}, \mathbf{f})p(\mathbf{f}|\mathbf{u}, \mathbf{y})p(\mathbf{u}|\mathbf{y})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u}. \tag{2.19}$$

If we assume that the inducing variables $\mathbf{u}$ summarize the data exactly, then $\mathbf{f}^*$ and $\mathbf{y}$ are independent of $\mathbf{f}$ given $\mathbf{u}$ and thus $p(\mathbf{f}^*|\mathbf{u}, \mathbf{f}) = p(\mathbf{f}^*|\mathbf{u})$ and $p(\mathbf{y}|\mathbf{u}, \mathbf{f}) = p(\mathbf{y}|\mathbf{u})$. The integral in Equation (2.19) can then be expressed as

$$p(\mathbf{f}^*|\mathbf{y}) = \int p(\mathbf{f}^*|\mathbf{u})p(\mathbf{f}|\mathbf{u})p(\mathbf{u}|\mathbf{y})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u} \tag{2.20}$$

$$= \int p(\mathbf{f}^*|\mathbf{u})p(\mathbf{u}|\mathbf{y}) \left(\int p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f} \right) \mathrm{d}\mathbf{u} \tag{2.21}$$

$$= \int p(\mathbf{f}^*|\mathbf{u})p(\mathbf{u}|\mathbf{y})\mathrm{d}\mathbf{u}. \tag{2.22}$$

Let $q(\mathbf{f}^*) = p(\mathbf{f}^*|\mathbf{y})$ and $\phi(\mathbf{u}) = p(\mathbf{u}|\mathbf{y})$ and rewrite the final expression with this new notation:

$$q(\mathbf{f}^*) = \int p(\mathbf{f}^*|\mathbf{u})\phi(\mathbf{u})\mathrm{d}\mathbf{u}. \tag{2.23}$$

The assumption that the inducing variables $\mathbf{u}$ summarize the data exactly will not hold in practical applications and $q(\mathbf{f}^*)$ is then only an approximation of the exact posterior $p(\mathbf{f}^*|\mathbf{y})$. To make $q(\mathbf{f}^*)$ a good approximation of $p(\mathbf{f}^*|\mathbf{y})$, Titsias proposes that we let $\phi(\mathbf{u})$ be a Gaussian distribution parameterized by mean vector $\mathbf{m}$ and covariance matrix $\mathbf{S}$ that we implicitly optimize by maximizing the following lower bound to the log likelihood:

$$\log p(\mathbf{y}) \geq \log \left[\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{Nm}^\top + \sigma_y^2\mathbf{I} \right] - \frac{1}{2\sigma_y^2} \operatorname{Tr}\left(\mathbf{K}_{NN} - \mathbf{Q}_{NN}\right) \tag{2.24}$$

where $\mathbf{Q}_{NN} = \mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{Nm}^{\top}$, $\mathbf{K}_{Nm} = k(\mathbf{X}, \mathbf{Z})$, $\mathbf{K}_{mm} = k(\mathbf{Z}, \mathbf{Z})$ and $\mathbf{K}_{NN} = k(\mathbf{X}, \mathbf{X})$. The lower bound above only requires computing the inverse $\mathbf{K}_{mm}^{-1}$ rather than $\mathbf{K}_{NN}^{-1}$ which is considerably faster since we assume that the data can be efficiently summarized (i.e. $m \ll N$).

The overall complexity of the Sparse GP method described above is $\mathcal{O}(Nm^2)$. As the dataset grows, it becomes increasingly expensive to add additional inducing points which hinders the application of Sparse GPs to giant datasets.

2.2.4 Stochastic Variational Gaussian process regression

As we discussed in Section 2.2.3, to get a GP model that can be applied to huge datasets we require that the complexity is independent of the number of datapoints. In [11], Hensman *et al.* introduced a technique called Stochastic Variational Inference for GPs (SVGP). The novelty of the SVGP is that the loss function can be expressed as a sum over the datapoints which permits the use of stochastic gradient ascent.

Unlike Titsias, Hensman *et al.* propose that we use the parameterization $\phi(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$ to derive a further lower bound:

$$\begin{aligned}\log p(\mathbf{y}) \geq &\log\left[\mathcal{N}\left(\mathbf{y}|\mathbf{K}_{mN}\mathbf{K}_{mm}^{-1}\mathbf{m}, \sigma_y^2\mathbf{I}\right)\right] \\ &- \frac{1}{2\sigma_y^2}\operatorname{Tr}\left(\mathbf{K}_{Nm}\mathbf{K}_{mm}^{-1}\mathbf{S}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mN}\right) \\ &- \frac{1}{2\sigma_y^2}\operatorname{Tr}\left(\mathbf{K}_{NN} - \mathbf{Q}_{NN}\right) - \mathrm{KL}\left(q(\mathbf{u})||p(\mathbf{u})\right) \end{aligned} \tag{2.25}$$

where $\mathrm{KL}(q||p)$ is Kullback-Leibler divergence [32], [33].

The expression above is quite daunting, but we note that we only need to invert the $m \times m$ matrix $\mathbf{K}_{mm}$. An additional property that Hensman *et al.* demonstrated is that the expression above can be separated into a sum of N terms - with each term corresponding to one datapoint $(\mathbf{x}_i, y_i)$. Expressing the loss as a sum in this way allows us to use stochastic gradient ascent [34] to estimate the gradient using only a small subset of the data.

In Figure 2.4, we visualize the results of using exact GP regression and SVGP regression to estimate the function $f(x) = \cos(x^2)$ from 400 noisy observations. In the top panel, we visualize the underlying function along with a 95% confidence region. In the middle, we visualize the results of the exact GP. And in the bottom, we visualize the results of the SVGP along with the optimized inducing points.

The inducing points have been optimized to be spread out evenly where the data is located, allowing the SVGP model to approximately summarize the data using only 10 locations. Note that the confidence region $\mathbf{f}^* \pm 2\boldsymbol{\Sigma}_f^*$ for the SVGP widens between the inducing points on the left - indicating that SVGP is not able to perfectly summarize the data. However, the predictive mean $\mathbf{f}^*$ between the two models does not differ significantly at a glance.

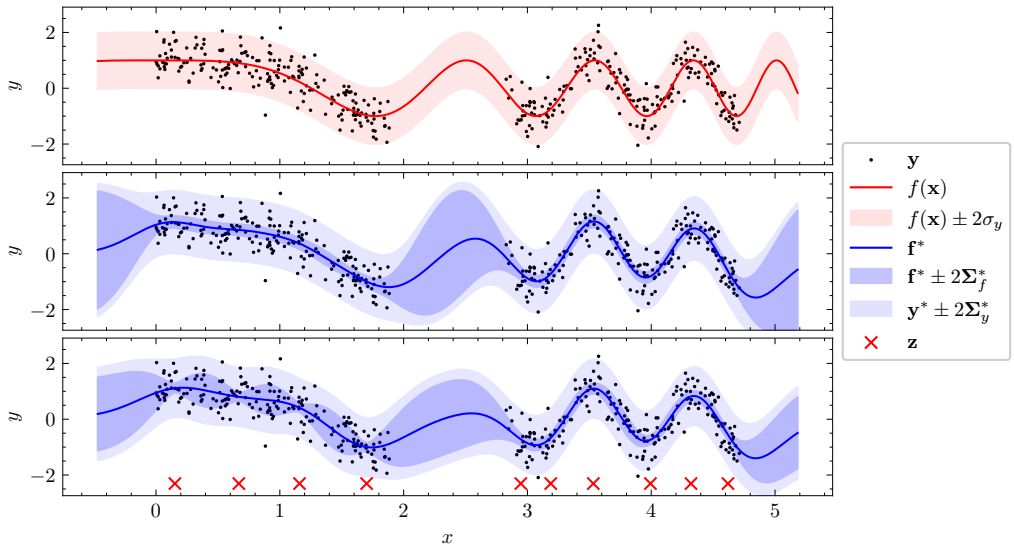


Figure 2.4: Estimation of the function $f(x) = \cos(x^2)$ (top) from 400 noisy observations using an exact GP (middle) and Stochastic Variational GP (bottom) with 10 inducing points.

2.2.5 Heteroskedastic noise modeling

In the preceding sections, we made the natural assumption that the observational noise (σ_y^2) equals an unknown constant (homoskedastic noise). In this section, we describe how to model the scenario where the observational noise varies across the sampling space (heteroskedastic noise).

In Bayesian inference, if the noise parameter σ_y^2 is set too high, then the posterior converges slower because the samples are deemed overly noisy. Conversely, if the noise parameter σ_y^2 is too low, the posterior will overemphasize the sampled data and may have difficulty converging smoothly. If a heteroskedastic GP can model the noise accurately, then the model should be able to assess the noise level of each sample and update the posterior appropriately.

There is no unique method of modeling heteroskedastic noise, but in this thesis we make the following assumptions:

$$f_1 \sim \mathcal{GP}(0, k_1), \quad f_2 \sim \mathcal{GP}(0, k_2) \tag{2.26}$$

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)), \text{ where } \boldsymbol{\mu} = f_1, \boldsymbol{\sigma} = \log(1 + \exp(f_2)), \tag{2.27}$$

where k_1 and k_2 are two kernel functions and $\text{diag}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{e}_i^\top \mathbf{x} \mathbf{e}_i \mathbf{e}_i^\top$ for $\mathbf{x} \in \mathbb{R}^n$. The idea is that one GP generates the mean function μ while another GP generates the standard deviation σ. Note that the softplus function $s(x) = \log(1 + \exp(x))$ ensures that the standard deviation is positive.

In Figure 2.5, we demonstrate the difference between homoskedastic and heteroskedastic modeling when $k_1 = k_{RBF}$ with $\ell = \frac{1}{2}$ and $k_2 = 2 \cdot k_{RBF}$ with $\ell = 1$. The heteroskedastic model is able to accurately predict how the noise varies, while the homoskedastic model finds a middle ground, leading to large under- and overestimations in some regions.

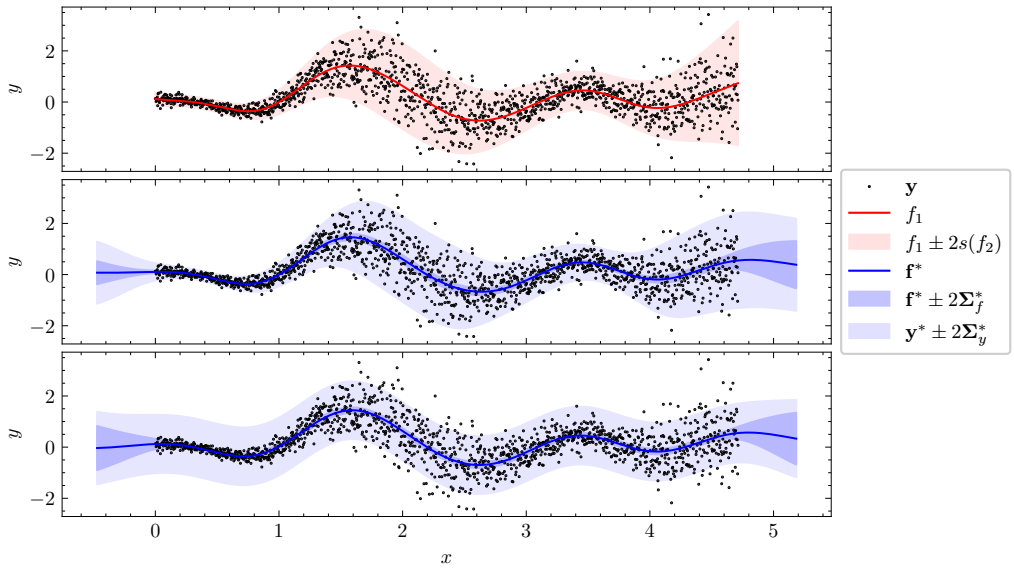


Figure 2.5: Gaussian process regression of a randomly sampled function (top) using heteroskedastic modeling (middle) and homoskedastic modeling (bottom).

2.2.6 Graph Gaussian processes

Since the navigation problem involves estimating the cost of traveling along each edge in a graph, it seems natural to incorporate the graph structure into the kernel of our GP. In [1], Borovitskiy *et al.* introduce the Graph Matérn GP (GMGP) by extending their earlier work on Riemann manifolds [35]. Borovitskiy *et al.* combine the stochastic partial differential equation (SPDE) definition of Matérn GPs in $\mathbb{R}^d$ with the graph Laplacian to create a GP on the nodes of a weighted and undirected graph. The GMGP is applied to traffic congestion estimation and classification in a scientific citation network.

To apply the GMGP to estimate energy consumption on directed edges of a weighted graph, the GMGP must be extended to directed graphs which requires us to define a positive semi-definite graph Laplacian. But first, we introduce the GMGP in more detail.

2.2.6.1 Graph Matérn Gaussian Process

Whittle [36] showed that GPs in $\mathbb{R}^d$ using the Matérn kernel (recall Equation (2.11)) satisfy the following SPDE

$$\left(\frac{2\nu}{\kappa^2} - \Delta\right)^{\frac{\nu}{2}+\frac{d}{4}} f = \mathcal{W} \tag{2.28}$$

where $\nu < \infty$, Δ is the Laplacian and $\mathcal{W}$ is Gaussian white noise. The details of SPDEs are not relevant to this thesis, but for further details and references see [1]. From Equation (2.28), we see that defining a Matérn GP on a graph requires analogs to the Laplacian Δ and Gaussian white noise $\mathcal{W}$ in the graph domain.

Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ be a weighted and undirected graph with nodes $\mathcal{V}$, edges $\mathcal{E}$ and

weight matrix $\mathbf{W}$. If $\mathbf{D}$ is the diagonal degree matrix such that $D_{ii} = \sum_{j=1}^{|\mathcal{V}|} W_{ij}$, then we define the graph Laplacian as

$$\boldsymbol{\Delta} = \mathbf{D} - \mathbf{W}. \tag{2.29}$$

Since $\boldsymbol{\Delta}$ is symmetric and positive semi-definite [37], $\boldsymbol{\Delta}$ permits an eigendecomposition $\boldsymbol{\Delta} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$. Next, let $\Phi : \mathbb{R} \to \mathbb{R}$ be a real function and $\Phi(\boldsymbol{\Lambda})$ be the function applying Φ to the diagonal elements of a diagonal matrix $\boldsymbol{\Lambda}$. Let $\Phi(\boldsymbol{\Delta}) = \mathbf{U}\Phi(\boldsymbol{\Lambda})\mathbf{U}^\top$ for the diagonalizable matrix $\boldsymbol{\Delta}$. If we now let $\Phi(\lambda) = \left(\frac{2\nu}{\kappa^2} + \lambda\right)^{\frac{\nu}{2}}$, then $\Phi(\boldsymbol{\Delta})$ is the graph equivalent operator in the left-hand side of Equation (2.28). Finally, by introducing a standard Gaussian $\boldsymbol{\mathcal{W}} \sim \mathcal{N}(0, \mathbf{I})$ we obtain the equivalent equation

$$\mathbf{U}\left(\frac{2\nu}{\kappa^2}\mathbf{I} + \boldsymbol{\Lambda}\right)^{\frac{\nu}{2}} \mathbf{U}^\top \mathbf{f} = \boldsymbol{\mathcal{W}}. \tag{2.30}$$

where the vector $\mathbf{f}$ defines the graph Gaussian process $\mathbf{f} : \mathcal{V} \to \mathbb{R}$. By applying Corollary 1, we obtain the Graph Matérn Gaussian process

$$\mathbf{f} \sim \mathcal{N}\left(0, \mathbf{U}\left(\frac{2\nu}{\kappa^2}\mathbf{I} + \boldsymbol{\Lambda}\right)^{-\nu} \mathbf{U}^\top\right). \tag{2.31}$$

An observant reader might notice that we inverted the sign of the Laplacian and dropped the $d/4$ term in the exponent. As explained by Borovitskiy *et al.*, the former is due to different sign conventions for graph and ordinary Laplacians whilst the latter is due to not needing $d/4$ to ensure regularity.

2.2.6.2 Extending Graph Matérn Gaussian Process to edges of directed graphs

To extend the Graph Matérn GP to the edges of a directed graph, we utilize the weighted line graph $\mathcal{L}(G) = (\mathcal{E}, \mathcal{C}, \mathbf{W}_\mathcal{L})$ introduced in Section 2.1.1 where $\mathcal{E}$ is the set of directed edges in G, $\mathcal{C}$ is the connections between directed edges and $\mathbf{W}_\mathcal{L}$ is the weight matrix of the directed graph. A connection $c = (e_1, e_2)$ exists only if $e_1 = (u, v)$ and $e_2 = (v, w)$ for nodes u, v and w in G.

The weight $W_{\mathcal{L},e_1,e_2}$ is set to $\bar{\ell}/\ell_{e_1}$ where $\bar{\ell}$ is the average length of all edges and ℓ_{e_1} is the length of edge e_1. As a consequence, $\mathbf{W}_\mathcal{L}$ and $\boldsymbol{\Delta}$ are not necessarily positive semi-definite. We replace the ordinary graph Laplacian with the incidence Laplacian

$$\boldsymbol{\Delta}_I = \mathbf{B}\mathbf{B}^\top \tag{2.32}$$

where the incidence matrix $\mathbf{B} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{C}|}$ has entries

$$B_{e,c} = \begin{cases} -W_{\mathcal{L},e_1,e_2} & \text{if } e = e_1, \\ W_{\mathcal{L},e_1,e_2} & \text{if } e = e_2, \\ 0 & \text{otherwise} \end{cases} \qquad \forall e \in \mathcal{E}, c = (e_1, e_2) \in \mathcal{C}. \tag{2.33}$$

Let $\boldsymbol{\Delta}_I = \mathbf{U}_I\boldsymbol{\Lambda}_I\mathbf{U}_I^\top$ denote the eigendecomposition of $\boldsymbol{\Delta}_I$, then the graph Matérn GP of the edges is

$$\mathbf{f} \sim \mathcal{N}\left(0, \mathbf{U}_I\left(\frac{2\nu}{\kappa^2}\mathbf{I} + \boldsymbol{\Lambda}_I\right)^{-\nu} \mathbf{U}_I^\top\right). \tag{2.34}$$

2.3 Related work

In this section, we discuss related work in energy-efficient navigation, GPs for bandit problems, and GPs on graphs.

2.3.1 Energy-efficient navigation

In [5], Holden *et al.* describe the RouteE model and package for accurate energy consumption estimation for various vehicle types. The package contains models that are pretrained using a large dataset of drive cycle data and energy-consumption data from FASTSim [6], a powertrain simulator. Their model achieves low estimation errors by using a random forest regression model. The RoutE model has been incorporated into Google Maps to provide a trade-off between time and fuel usage [7].

Abousleiman and Rawashdeh combine an electric vehicle model with the Bellman-Ford algorithm to compute energy-efficient routes in a graph [9]. Abousleiman and Rawashdeh observe that the most energy-efficient route is not necessarily the shortest nor fastest route. Additionally, the Bellman-Ford approach is demonstrated to be ill-suited for large graphs as the computational complexity is too high.

2.3.2 Gaussian processes for bandit problems

In [22], Nika *et al.* consider the contextual combinatorial multi-armed bandits problem with changing action sets (C3-MAB). The C3-MAB is formulated in a general manner to encompass a broad set of problems, including the online shortest path problem considered in this thesis. The authors introduce the O'CLOCK-UCB algorithm that combines GPs and UCB to solve C3-MAB and apply the algorithm to movie recommendation and crowdsourcing problems.

2.3.3 Gaussian processes on graphs

Most of the proposed GPs for graph problems in the literature consider undirected or unweighted graphs [38]–[41]. Applying these GPs to the navigation problem requires us to extend them, as done in Section 2.2.6.2. This section reviews similar works and discusses their usefulness to our problem.

Li *et al.* introduce Stochastic Deep GP over Graphs (SGPG) to learn mappings between input and output signals in a graph [38]. The signals are vectors on each node in the graph, and the model can be applied to traffic flow prediction. In the navigation problem, we receive signals on the subset of edges that we travel along - it is unclear if the SGPG can be applied in this setting or if it needs to be generalized.

In [39], Ng *et al.* consider semi-supervised classification problems on graphs and demonstrate that their Graph GP (GGP) model performs well on both small and large datasets. In addition, Ng *et al.* also apply their model in an active learning context and demonstrates that it can learn effectively. The GGP model assumes the

graph is undirected and unweighted and would need to be generalized to directed and weighted graphs.

Opolka *et al.* introduce a graph GP model that incorporates the graph structure using spectral graph wavelets in [40]. The graph wavelet GP has similarities to the graph Matérn GP in [1]. Notably, Opolka *et al.* propose alternative functions $\Phi(\lambda)$ that use multiple length-scales to capture signals on different frequencies.

3

Implementation and methods

In this chapter, we describe the implementation details of the GPs and the experimental setup.

Initially, Section 3.1 discusses the GP implementation in detail. Then, Section 3.2 describes the generation of a synthetic road network, and Section 3.3 describes the setup for the simulation of realistic road networks. Finally, Section 3.4 describes the priors, Section 3.5 specifies the default parameter values, and Section 3.6 introduces the evaluation metrics used.

3.1 Implementation of GP

To define, create and optimize the GPs throughout this thesis, we use the GPyTorch library [30] in Python. GPyTorch is an efficient and modular library for GPs built on top of PyTorch [42], allowing it to utilize GPU acceleration.

GPyTorch does not provide out-of-the-box complete implementations for GPs. Instead, GPyTorch provides building blocks for designing GPs. In this section, we describe how our GP is constructed and optimized.

Our GP implementation builds upon the *Stochastic Variational GP Regression*[1] and *Using Natural Gradient Descent with Variational Models*[2] examples available in the documentation of version 1.10. The model has four key components: the mean function $m(\mathbf{x})$, the kernel function $k(\mathbf{x}, \mathbf{x}')$, the variational distribution and the variational strategy. The two former components describe the mathematical properties of the GP whilst the latter two components describe how GPyTorch should optimize the variational parameters.

The input vector $\mathbf{x} \in \mathcal{E} \times \mathbb{R}^d$ and inducing points $\mathbf{z} \in \mathcal{E} \times \mathbb{R}^d$ of the GP consists of one edge dimension and d feature dimensions. Since $\mathcal{E}$ is discrete, the first dimension of $\mathbf{z}$ can not be optimized by continuous optimizers (such as gradient descent). To accommodate this, we extend the default `VariationalStrategy` class with a `VariationalStrategyWithIndex` class that only optimizes the d feature dimensions of the inducing points. Thus, the edge index of $\mathbf{z}$ must be set manually.

[1] See `https://docs.gpytorch.ai/en/v1.10/examples/04_Variational_and_Approximate_GPs/SVGP_Regression_CUDA.html`.

[2] See `https://docs.gpytorch.ai/en/v1.10/examples/04_Variational_and_Approximate_GPs/Natural_Gradient_Descent.html`.

Algorithm 5 SVGP Optimization Procedure

procedure UPDATEPARAMETERS($\boldsymbol{a}_{1:t-1}, \boldsymbol{r}_{1:t-1}, \boldsymbol{\mu}_{1:t-1}, \boldsymbol{\varsigma}_{1:t-1}, \boldsymbol{\sigma}_{1:t-1}, \boldsymbol{\mathcal{X}}$)
  $N \leftarrow$ The total traversed edges ($\sum_{i=1}^{t-1} \text{length}(\boldsymbol{a}_i)$).
  $\boldsymbol{a}_{1:N} \leftarrow$ Indices of traversed edges
  $\boldsymbol{r}_{1:N} \leftarrow$ Reward per traversed edge.
  Set inducing points $\mathbf{z}$ to top M most frequently visited edges.
  for $\tau \in \{1, \ldots,$ Number of gradient steps$\}$ **do**
    $\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{r}} \leftarrow$ Subsample $\boldsymbol{a}_{1:N}, \boldsymbol{r}_{1:N}$ of size B.
    $\mathbf{X} \in \mathbb{R}^{B \times (d+1)} \leftarrow$ Concatenate edge index $\tilde{a}_i$ and corresponding edge features into each row, $\forall i \in \tilde{\boldsymbol{a}}$.
    $\mathbf{y} \leftarrow -\tilde{\boldsymbol{r}}$
    $\mathcal{L} \leftarrow$ Compute Equation (2.25) with $\mathbf{X}$ and $\mathbf{y}$.
    Update variational parameters with natural gradient descent.
    Update kernel and likelihood parameters with Adam.
  end for
  Compute $\boldsymbol{\mu}_t, \boldsymbol{\varsigma}_t, \boldsymbol{\sigma}_t$ using the optimized GP parameters.
  Return $\boldsymbol{\mu}_t, \boldsymbol{\varsigma}_t, \boldsymbol{\sigma}_t$
end procedure

When training the model, we optimize the `VariationalELBO` loss function, see Equation (2.25), using two separate optimizers, the Adam [43] optimizer for the kernel parameters and the natural gradient descent (NGD) [44] optimizer for the inducing points. To use the NGD optimizer, our model uses the `NaturalVariationalDistribution` class. See Algorithm 5 for a high-level description of the complete procedure.

Notably in Algorithm 5, the inducing points $\mathbf{z}$ are set to equal the index and features of the top M most frequently visited edges before optimizing the parameters. This heuristic should enable the model to make accurate predictions on the relevant edges.

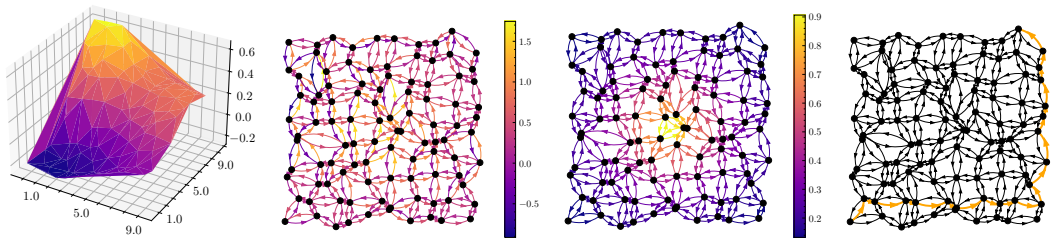
3.1.1 Mean function

The mean function $m(\mathbf{x}) = C$ is a constant function with an optimizable parameter C. The model can be provided an informative prior for each edge $\mathbf{p} \in \mathbb{R}^{|\mathcal{E}|}$ as an offset to the mean function: $m(\mathbf{x}) = C + p_e$ where e is the edge index in the input vector $\mathbf{x}$. If not otherwise specified, assume $\mathbf{p} = \mathbf{0}$.

3.1.2 Kernel function

To implement the Graph Matérn kernel in GPyTorch, we translate the original code[3] of [1] from the GPflow library [45] to GPyTorch. For large graphs, the kernel is approximated by the 500 smallest eigenpairs of the incidence Laplacian $\boldsymbol{\Delta}_I$. Let k_G denote the Graph Matérn kernel with optimizable parameters $\nu_G > 0$, $\kappa_G > 0$ and $\sigma_G > 0$. The input to the Graph kernel consists only of the edge index $x_e \in \mathcal{E}$.

[3]Available at `https://github.com/spbu-math-cs/Graph-Gaussian-Processes`.



(a) (b) (c) (d)

Figure 3.1: Synthetic road network with a) elevation, b) expected energy consumption, c) cluster energy consumption and d) most energy-efficient path from the bottom-left to the top-right.

Next, let k_f denote a feature kernel whose input $\mathbf{x}_f \in \mathbb{R}^d$ consists only of the features of an edge. We use an ordinary Matérn kernel with fixed $\nu = 5/2$ but tunable outputscale σ_f and lengthscales $\ell \in \mathbb{R}^d_+$ for each dimension (see ARD in Section 2.2.1).

In addition to the two basic kernels k_G and k_f, we define the following shorthand notation for composing the two kernels:

$$k_{G+f} = k_G + k_f, \tag{3.1}$$
$$k_{G \cdot f} = k_G \cdot k_f, \tag{3.2}$$
$$k_{G \cdot f + f} = k_G \cdot k_f + k'_f. \tag{3.3}$$

Note that for the final kernel, $k_{G \cdot f + f}$, the two feature kernels have separate parameters which allows for a more expressive model. For the multiplicative compositions $(k_G \cdot k_f)$, we assume one of the output scales, σ_G or σ_f, is fixed to 1.

3.1.3 Heteroskedastic noise GP

To add heteroskedasticity to an ordinary SVGP model (corresponding to $f_1 \sim \mathcal{GP}(0, k_1)$ in Equation (2.26)), we replace the usual likelihood class, `GaussianLikelihood`, with the class `_GaussianLikelihoodBase` with the parameter `noise_covar` set to an instance of the undocumented class `HeteroskedasticNoise`. The `HeteroskedasticNoise` class is provided a second SVGP model (corresponding to $f_2 \sim \mathcal{GP}(0, k_2)$ in Equation (2.26)) - which we refer to as the *noise GP*.

If nothing else is specified, assume the noise GP has the same covariance function as the ordinary GP. The parameters for the ordinary and noise GP are initialized identically but are optimized separately. When an informative prior $\mathbf{p}$ is used for the mean function of the ordinary GP, the noise GP is provided $\tilde{\mathbf{p}} = \text{abs}(\mathbf{p}) \cdot c$ as the informative prior where $\text{abs}(\cdot)$ is applied elementwise and $c > 0$ is a scaling factor.

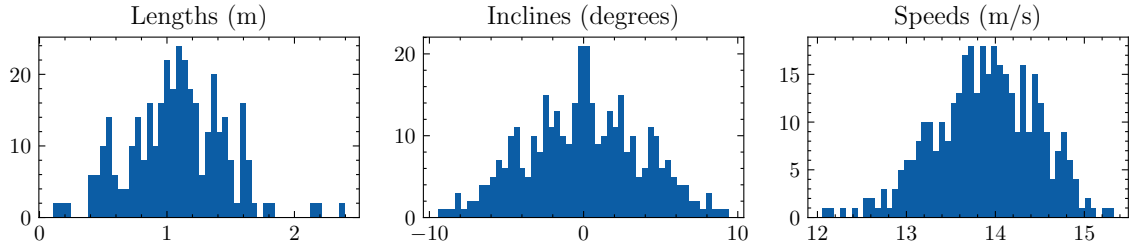


Figure 3.2: Features of the edges in the synthetic road network.

3.2 Synthetic road networks

We create a synthetic network to evaluate the GP model against a known ground-truth, visualized in Figure 3.1.

Let $G_s(\mathcal{V}_s, \mathcal{E}_s)$ denote a graph representing a synthetic road network with nodes $\mathcal{V}_s$ and edges $\mathcal{E}_s$. To construct the graph, the nodes are initially given x and y coordinates in a regular grid of size 10×10 with side-length of 1 m, thus $|\mathcal{V}_s| = 100$. Adjacent nodes are connected with bidirectional edges. The coordinates of each node is perturbed with Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.3^2)$ to make the network irregular and harder to traverse. To add elevation to each node, we sample the z-coordinates from a GP using the RBF-kernel with $\ell = 2$ and scaled by the factor 0.1. We visualize the elevation in Figure 3.1a. Using the x-,y- and z-coordinates, we compute the length and incline of each edge. The speed of each edge is set to $50 + \epsilon_e$ km/h where $\epsilon_e \sim \mathcal{N}(0, 2^2)$. In Figure 3.2, we visualize the distribution of features in the synthetic network.

Using the vehicle and environmental parameters in Table 3.2, we compute the energy consumption of the deterministic model as described by Equation (2.1).

To model pseudo-congestion in the network, we add an additional term to the energy consumption that we refer to as the cluster energy consumption. The cluster energy consumption is determined by the distance from the edge to the center of the graph:

$$E_{e,c} = \exp\left(-\sqrt{(x_e - x_g)^2 + (y_e - y_g)^2}/\,3\right) \tag{3.4}$$

where the edge coordinates x_e, y_e for edge $e = (u, v)$ corresponds to the edge mid-section. The graph center coordinates (x_g, y_g) are set as the average of all node coordinates. See Figure 3.1c for a visualization of the cluster energy consumption.

The true expected energy consumption is then $\theta_e^* = E_e^{\text{det}} + E_{e,c}$ and the true variance of the energy consumption is set to $(\sigma_e^*)^2 = (0.25 \cdot \theta_e^*)^2$. The most energy-efficient path from the start node to the end node is visualized in Figure 3.1d.

3.3 Simulated road networks

To simulate a real-world road network, we use the simulator SUMO [46] (v.1.16.0) combined with traffic scenarios and networks of two European cities: Monaco [47]

Table 3.1: The number of nodes, edges and connections in the three real-world road networks. The subcomponent is the largest strongly connected component.

Scenario	Complete			Subcomponent		
	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{C}\rvert$
Luxembourg	2247	5779	11 286	1940	5169	10 133
Monaco	1766	3613	5838	1659	3474	5660

and Luxembourg [48]. The road network of Luxembourg is combined with elevation data from [49] using QGIS [50] and the *netconvert* tool from SUMO. Adding the elevation data invalidates the position of certain bus stops and traffic detectors in the Luxembourg scenario. We recompute valid positions by considering the prior relative position of the object along the edge and rescale it based on the new length.

To conveniently be able to select edges to travel between, we restrict the road networks to the largest strongly connected component of the line graph $\mathcal{L}(G)$. Table 3.1 shows the number of nodes, edges, and connections for the complete and restricted networks. The restriction has a minor impact on Luxembourg, where many smaller residential roads lack U-turns, but this should not affect the navigational difficulty.

Due to unforeseen crashes in SUMO, we could not use the same simulation setup for Monaco and Luxembourg. For Luxembourg, the network was simulated for 24 hours, and the state of the simulation was saved every 100 seconds. When running Algorithm 1, the simulation was started using a randomly selected state at each time step t. For Monaco, the simulation was started with an empty network at a randomly selected time between 05:00 and 14:00 during every time step t. Although the Monaco simulation initializes vehicles, the traffic will be underestimated.

In Figure 3.3, the two road networks are visualized along with evaluation routes. The evaluation routes involve multiple regions of the respective networks, thus allowing for many alternative paths. Two evaluation routes are used for Luxembourg since it has a larger road network and more realistic traffic.

We also intended to simulate the significantly larger road network of Turin [51], but multiple reasons led us to exclude it. First, the Turin scenario is not as realistic compared to Luxembourg and Monaco since it uses a queue-based simulation and very high speeds on certain road segments. Second, the network gets incredibly congested when using later versions of SUMO. The energy consumption could not be extracted when using earlier versions of SUMO with accurate congestion levels. Finally, the scenario occasionally crashed when attempting to initialize the saved states of the simulation.

3.4 Informative priors

In this section, we introduce three methods of selecting the priors $\boldsymbol{\mu}_0, \boldsymbol{\varsigma}_0$ and $\boldsymbol{\sigma}_0$.

The first prior provides the most amount of information and we refer to as the *full*

(a) Luxembourg A

(b) Luxembourg B

(c) Monaco

Figure 3.3: The road networks of a-b) Luxembourg and c) Monaco with evaluation routes highlighted.

Table 3.2: Vehicle and environmental parameters for the energy model.

Variable	Value	Unit
Mass m	1830	kg
Rolling resistance coefficient C_r	0.01	
Front surface area A	2.6	m^2
Air drag coefficient C_d	0.35	
Power train efficiency η^+	0.98	
Recuperation efficiency η^-	0.96	
Gravitational acceleration g	9.82	m/s^2
Air density ρ	1.2	kg/m^3

prior. For the full prior, we let $\mu_{e,0} = E_e^{\text{det}}$, $\varsigma_{e,0}^2 = (0.25E_e^{\text{det}})^2$ and $\sigma_{e,0}^2 = (0.1E_e^{\text{det}})^2$ $\forall e \in \mathcal{E}$. The full prior is provided a good but simple estimation of the expected energy consumption and the prior standard deviation in both noise and mean are scaled proportionally to the prior mean.

In the second prior, the values are initialized *uniformly* which provides less information to the model overall but still gives the model the correct scale. Let $\overline{E^{\text{det}}} = \frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}} E_e^{\text{det}}$ and $\sigma^{\text{det}} = \sqrt{\frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}}(E_e^{\text{det}} - \overline{E^{\text{det}}})^2}$ denote the mean and standard deviation of the deterministic energy consumption. Then, the uniform prior assigns the following, $\mu_{e,0} = \overline{E^{\text{det}}}$, $\varsigma_{e,0}^2 = (0.25\sigma^{\text{det}})^2$ and $\sigma_{e,0}^2 = (0.1\sigma^{\text{det}})^2$ $\forall e \in \mathcal{E}$.

Finally, in the third prior we assign an *optimistic* prior mean and uniform prior variance: $\mu_{e,0} = 0$, $\varsigma_{e,0}^2 = (0.25\sigma^{\text{det}})^2$ and $\sigma_{e,0}^2 = (0.1\sigma^{\text{det}})^2$ $\forall e \in \mathcal{E}$.

The three priors described above are intended for Bayesian inference and are not compatible with the homo- and heteroskedastic GP models since they lack explicit parameters for $\boldsymbol{\varsigma}_0$. To ensure the GP is comparable to the Bayesian model, the kernel output scale parameters σ_f^2 and σ_G^2 of both GP models are initialized as $\boldsymbol{\varsigma}_0^\top \boldsymbol{\varsigma}_0/|\mathcal{E}|$. Additionally for the homoskedastic GP, the variance σ_y^2 of the likelihood is initialized as $\boldsymbol{\sigma}_0^\top \boldsymbol{\sigma}_0/|\mathcal{E}|$.

3.5 Parameters

In this section, we specify the default parameter values for the electric vehicle model and the GP model. If nothing else is specified in specific experiments, then the parameter values given in this section are used.

We use the default parameters for electric vehicles provided by SUMO [46] that describes a Kia Soul EV 2020, see Table 3.2.

We standardize the edge features of each network and set the initial ARD lengthscale to 1 for all features in the GP models. The graph kernel is initialized with parameters $\nu_G = 2$, $\kappa_G = 1$ and σ_G set according to the prior. The natural gradient descent learning rate is set to 0.1 whilst the Adam learning rate is set to 0.01. Both the GP

and Bayesian inference models use Thompson sampling and the uniform prior. The GP model uses homoskedastic noise modeling as standard.

In the synthetic network, the GP models use a batch size B of 2500 and 1 gradient step per optimization procedure in Algorithm 5. The number of inducing points equals the number of edges.

In the simulated networks, the GP models use a batch size B of 8000 and 5 gradient steps per optimization procedure. The simulated scenarios are CPU-bottlenecked, thus the added optimization steps does not add significantly to the run time. The number of inducing points is set to 1000.

3.6 Evaluation metrics

For the synthetic road network, cumulative regret is used to measure the difference between our online learning model and an oracle knowing the optimal route, see Section 2.1.3.1. The optimal expected reward $\boldsymbol{r}^*$ and the expected reward $\mathbb{E}[\boldsymbol{r}_t]$ are computed using the negation of the rectified weights:

$$w_e \leftarrow \mathbb{E}[z_e] \text{ where } z_e \sim \mathcal{N}^R(\theta_e^*, (\sigma_e^*)^2). \tag{3.5}$$

If the optimal path is given by $\mathcal{P}^*$ and the path selected at step t is given by $\mathcal{P}_t$, then $\boldsymbol{r}^* = \sum_{e \in \mathcal{P}^*} -w_e$ and $\mathbb{E}[\boldsymbol{r}_t] = \sum_{e \in \mathcal{P}_t} -w_e$.

Since the true parameters θ_e^* and $(\sigma_e^*)^2$ are unknown in the simulated road networks, we use the final model predictions in the simulated networks for each model separately to compute an estimate of the cumulative regret:

$$\text{EstimatedRegret}(T) = \sum_{t=1}^{T} (\hat{\boldsymbol{r}}^* - \hat{\mathbb{E}}[\boldsymbol{r}_t]). \tag{3.6}$$

Here, $\hat{\boldsymbol{r}}^*$ and $\hat{\mathbb{E}}[\boldsymbol{r}_t]$ are instead computed with the weights determined by the final posterior mean: $w_e = \max(\mu_{e,T}, 0)$.

Since the models' estimations are likely inaccurate for road segments with few samples, we complement the estimated regret with the metric *random regret*. For a given set of observations of the energy consumption along each route obtained in Algorithm 1, $\{\boldsymbol{r}_t\}_{t=1}^T$, the random regret is

$$\text{RandomRegret}\,(T) = \sum_{t=1}^{T} (\hat{\boldsymbol{r}} - \boldsymbol{r}_t) \tag{3.7}$$

where $\hat{\boldsymbol{r}}$ is the negation of the estimated cost of the most energy-efficient route. We compute the average energy consumption across the last 25 time steps for each run and then use the minimum as our estimate of the average lowest energy route $\hat{\boldsymbol{r}}$. In principle, if a model has identified a strictly better route, it should drive along this route in the final time steps. Inspecting the frequency of the observed lowest energy routes revealed that most routes were only driven once. By averaging across the 25

last time steps, we reduce the risk of underestimating the lowest energy route due to an outlier.

Note that random regret uses the observed energy consumption and thus the contribution of the energy recuperation is correctly accounted for. Additionally, since we subtract the same estimate from all runs, the difference in random regret equals the difference in total energy consumed.

To measure the models' ability to estimate the noise, we use the metric quantile coverage error (QCE). Given a $q\%-$confidence interval and real datapoints, the QCE measures the accuracy of the confidence interval. For example, if $p\%$ of the datapoints lie in a $q\%$-confidence interval then the QCE is $|p - q|\%$. For a given confidence level q and QCE, the percentage of datapoints in the confidence interval is not necessarily unique and thus we use three confidence levels to understand if the model under- or overestimates the noise.

4

Results

In this chapter, we further describe the comparisons and present the experimental results.

4.1 Comparison of kernel design

To answer Q1, what is an effective kernel function for the GP, we compare the 5 kernels introduced in Section 3.1.2 on the synthetic road network and then compare a subset of them on the Luxembourg scenario.

We run a GP model with each kernel on the synthetic road network 10 times with a horizon of $T = 1000$ using Algorithms 1 and 5. The Bayesian inference method described in Section 2.1.2.1 is used as a baseline. The cumulative regret for the kernels and the Bayesian baseline is shown in Figure 4.1. The results indicate that the kernels $k_{G \cdot f+f}$, k_f and k_{G+f} improve upon the baseline. The kernels $k_{G \cdot f}$ and k_G have lower regret than the Bayesian inference model initially but fail to converge to a stable route in the end leading to accelerating cumulative regret. In the final time steps, the cumulative regret curve of the baseline is almost horizontal whilst the remaining curves are still increasing. With a longer horizon, its possible that the baseline would achieve a lower regret than all kernels.

To verify that the three well-performing kernels on the synthetic network ($k_{G \cdot f+f}$, k_f, and k_{G+f}) also perform well on a simulated network, we run Algorithm 1 5 times on Luxembourg A with a horizon of $T = 200$. As described in Section 3.6, the estimated

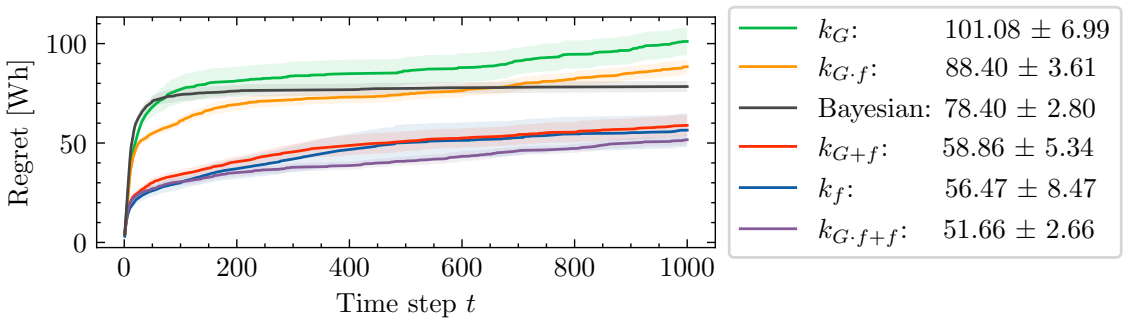


Figure 4.1: Cumulative regret on the synthetic road network for different kernels. Average final cumulative regret to the right.

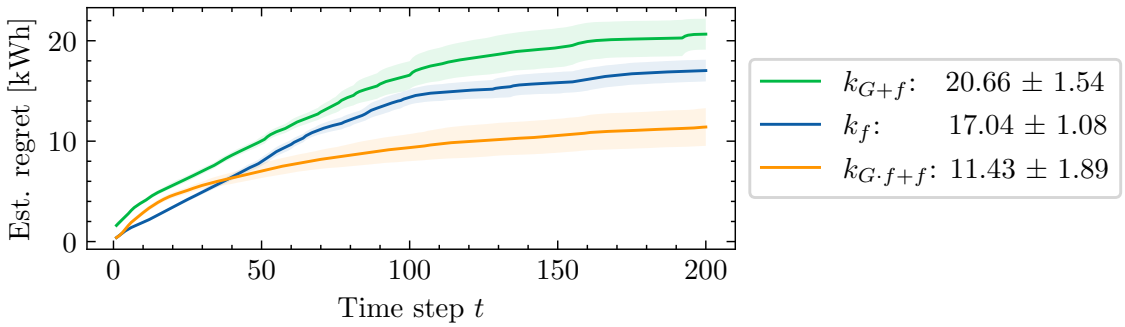


Figure 4.2: Estimated regret on Luxembourg A for 3 promising kernels.

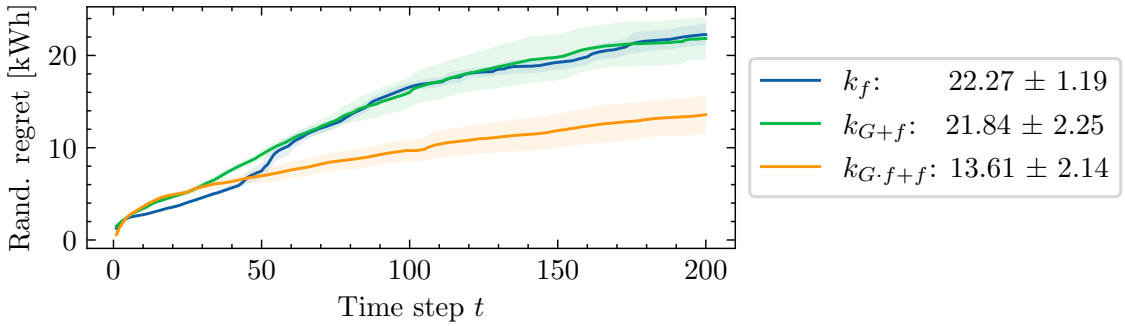


Figure 4.3: Random regret on Luxembourg A for 3 promising kernels.

Table 4.1: Average total energy consumption at $t = 200$ on Luxembourg A. Mean and standard error from 5 repetitions.

Kernel	Energy [kWh]
k_f	279.14 ± 1.19
k_{G+f}	278.71 ± 2.25
$k_{G\cdot f+f}$	270.48 ± 2.14

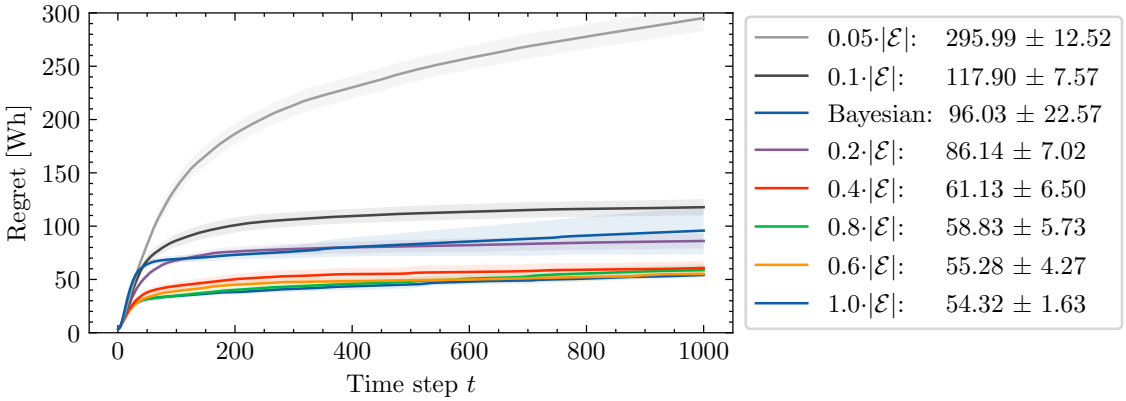


Figure 4.4: Cumulative regret on the synthetic road network with varying number of inducing points relative to the number of edges.

regret is based on the models' energy consumption predictions at $t = T$. Since a model that learns poorly will tend to underestimate its regret, we use random regret as a second performance measure. The results are shown in Figures 4.2 and 4.3. Finally, the total energy consumption is reported in Table 4.1 to contextualize the scale of the estimated regret and random regret.

From Figures 4.2 and 4.3, the kernel $k_{G \cdot f+f}$ has both the lowest estimated regret and lowest random regret in the Luxembourg scenario. The relative order of k_f and $k_{G \cdot f+f}$ is inconsistent in the estimated regret and random regret, highlighting that we should not overemphasize the estimated regret in our analysis. From Table 4.1, the difference in total energy consumption between the kernels k_f and $k_{G \cdot f+f}$ corresponds to a 3% reduction in energy consumption over 200 time steps.

4.2 Comparison of number of inducing points

To answer Q2, what is the impact of the number of inducing points on the GP model, we vary the number of inducing points on the synthetic road network.

We use a homoskedastic GP model with the kernel $k_{G \cdot f+f}$ as it was demonstrated to work well in Section 4.1. The number of inducing points is set relative to the number of edges in the synthetic road network and varies from 5% up to all edges.

In Figure 4.4, we present the average cumulative regret on the synthetic road network with horizon $T = 1000$ over 5 runs. Between 0.4 and 1.0 relative number of inducing points, the cumulative regret of the GP model does not increase significantly. However, when the relative number of inducing points is below 0.4, we observe large increases in the cumulative regret. Although the Bayesian inference method was run with the same parameters as in Section 4.1, we observe a significant increase in cumulative regret due to one outlier where the Bayesian inference method converges to a route with high regret.

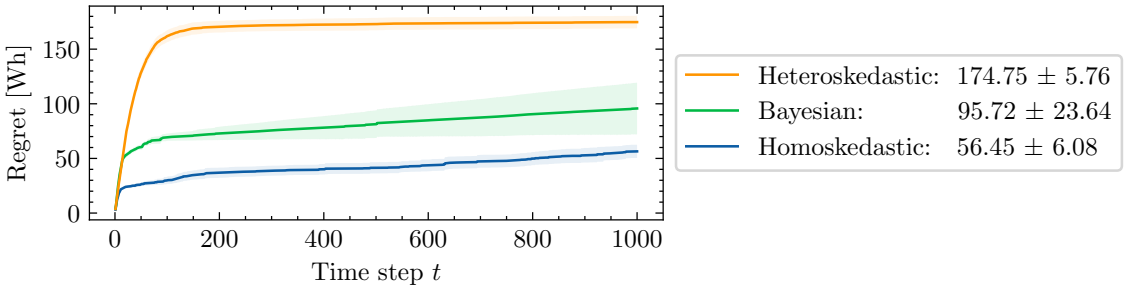


Figure 4.5: Cumulative regret on the synthetic road network with homoskedastic and heteroskedastic noise modeling compared to a Bayesian inference baseline.

Table 4.2: Quantile coverage error on synthetic road network at $t = 1000$.

Model	$q = 50\%$	$q = 75\%$	$q = 95\%$
Homoskedastic	43.0 ± 0.33	63.2 ± 0.33	76.2 ± 0.49
Heteroskedastic	$\mathbf{20.1} \pm 0.53$	$\mathbf{9.18} \pm 0.27$	$\mathbf{1.3} \pm 0.1$

4.3 Comparison of noise modeling

To answer Q3, what is the impact of homoskedastic versus heteroskedastic modeling of the noise, we compare the homoskedastic and heteroskedastic noise models on the synthetic network and Luxembourg A.

4.3.1 Noise modeling on synthetic road network

The homo- and heteroskedastic GP models on the synthetic network are compared to the Bayesian inference baseline with a horizon of $T = 1000$. The average cumulative regret across 5 runs for the 3 models is presented in Figure 4.5. The heteroskedastic model initially incurs large regret leading to a significantly higher final cumulative regret. However, the heteroskedastic model seems to identify a better final route on average than the Bayesian and homoskedastic models, leading to an almost horizontal curve at the end.

We report the quantile coverage error for three confidence levels in Table 4.2. First, we note that the noise in the synthetic scenario has been purposefully set to be heteroskedastic. Across all three confidence levels, the heteroskedastic model has a lower quantile coverage error indicating that it can effectively model the noise when the underlying data is heteroskedastic. For $q = 75\%$ and $q = 95\%$, the QCE of the homoskedastic GP exceeds $1 - q$, and thus the noise is mostly underestimated.

4.3.2 Noise modeling on simulated road networks

The same models are also compared on Luxembourg A and Monaco with a horizon of $T = 200$. The estimated regret and random regret is shown in Figures 4.6 and 4.7. On Luxembourg A, both models have similar estimated and random regret curves. However, the heteroskedastic model has larger estimated regret and random regret

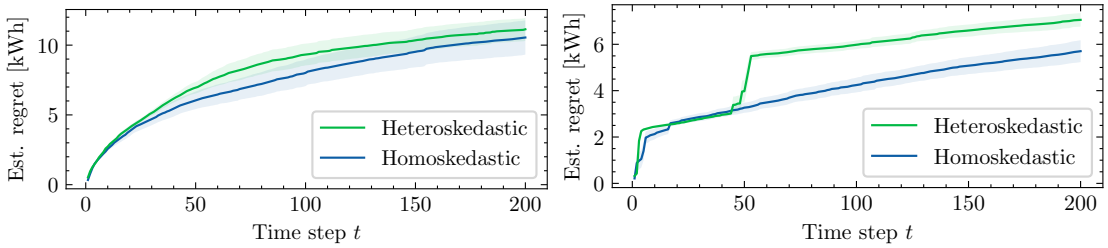


Figure 4.6: Estimated regret on Luxembourg A (left) and Monaco (right).

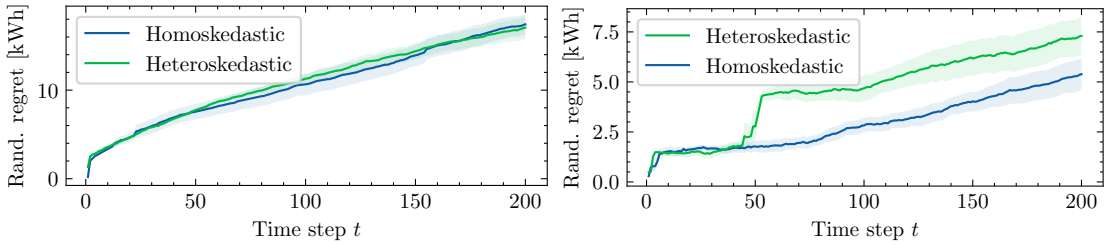


Figure 4.7: Random regret on Luxembourg A (left) and Monaco (right).

on Monaco due to exploring inefficient routes around $t = 50$. Unlike the synthetic scenario, the heteroskedastic model does not accumulate significant regret in the start.

Again, to contextualize the regret metrics, the total estimated regret and energy consumption are reported in Table 4.3. The homoskedastic model has slightly lower estimated regret in both scenarios and slightly lower total energy consumption on Monaco. Overall, both models are comparable in their ability to find energy-efficient routes in the simulated scenarios with a small advantage for the homoskedastic model.

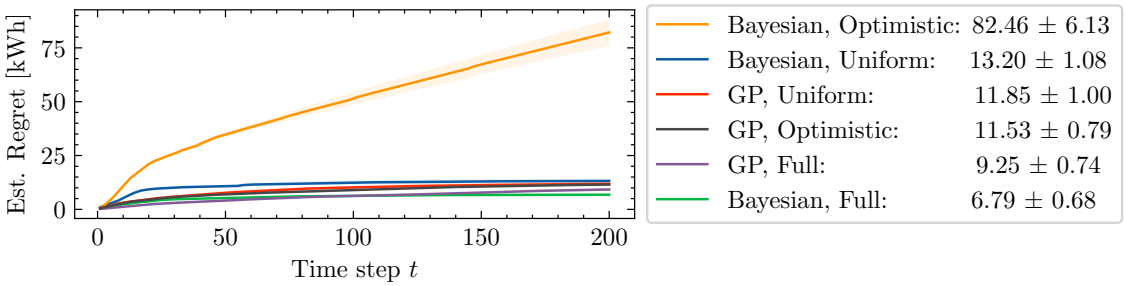
Next, we report the QCE in the simulated scenarios. The results in Table 4.4 show that the homoskedastic model has a higher QCE overall and once again tends to overestimate the noise. The heteroskedastic model has significantly lower QCE but the QCE increases with q for Luxembourg A, unlike the synthetic scenario and Monaco where the opposite is observed. The high QCE for $q = 75\%$ and $q = 95\%$ on Luxembourg A indicate that the heteroskedastic model also tends to underestimate the noise, although by a smaller margin than the homoskedastic model.

Table 4.3: Final estimated regret and total energy consumption on Luxembourg and Monaco at $t = 200$, in kWh.

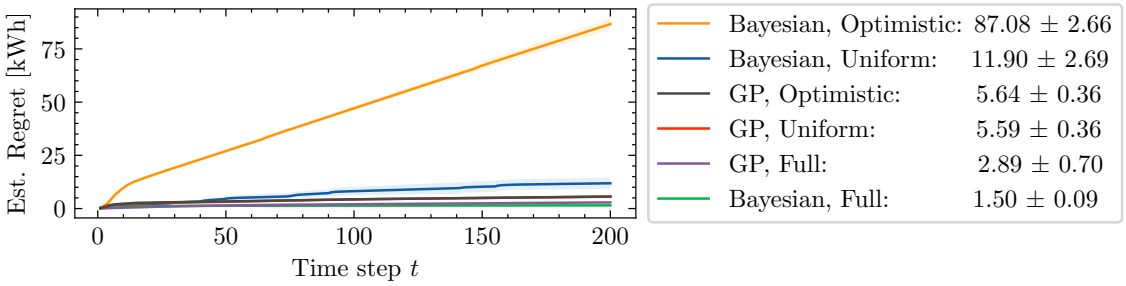
Model	Luxembourg A		Monaco	
	Est. Regret	Energy	Est. Regret	Energy
Homo.	**10.5** $\pm$ 1.2	272 $\pm$ 1.1	**5.7** $\pm$ 0.47	**51.6** $\pm$ 0.8
Hetero.	11.1 $\pm$ 0.83	**271** $\pm$ 1.2	7.05 $\pm$ 0.29	53.5 $\pm$ 0.96

Table 4.4: Quantile coverage error on Luxembourg and Monaco at $t = 200$.

Model	Luxembourg A			Monaco		
	$q = 50\%$	$q = 75\%$	$q = 95\%$	$q = 50\%$	$q = 75\%$	$q = 95\%$
Homo.	35.9	52.8	62.0	**19.0**	32.3	38.9
Hetero.	**15.8**	**26.3**	**29.2**	23.7	**10.9**	**1.43**



(a)



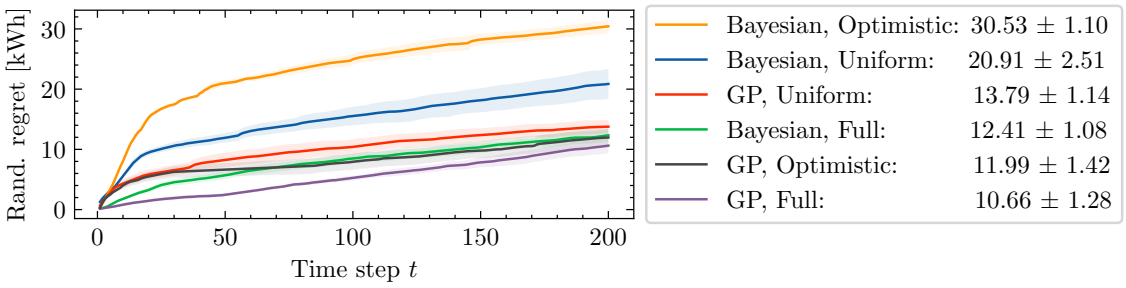
(b)

Figure 4.8: Estimated regret for homoskedastic GP and Bayesian inference model using the three priors, Optimistic, Uniform and Full, on a) Luxembourg A and b) Monaco.

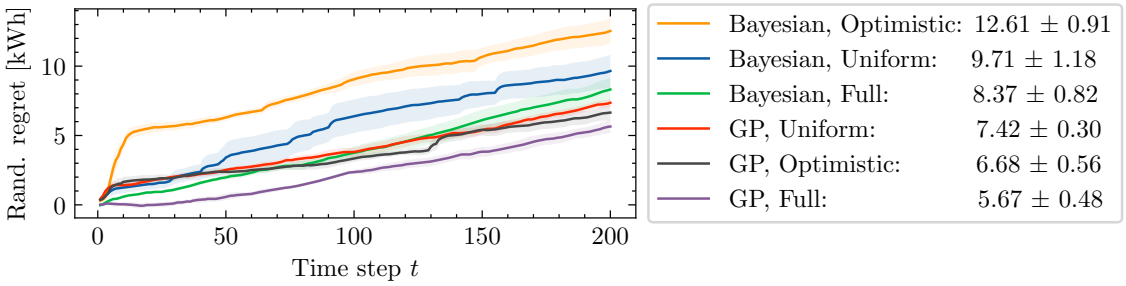
4.4 Comparison of informative prior

To answer Q4, what is the impact of informative priors on the GPs' performance and exploration, we test the three priors on Luxembourg A and Monaco. As before, we use the Bayesian inference method as a baseline to understand the difference in exploration between the GP and the baseline.

The estimated regret in both scenarios is presented in Figure 4.8. The Bayesian baseline with optimistic priors incurs a significantly higher estimated regret than all other models, likely because it is still optimistic that there are low-cost routes after 200 time steps and thus underestimates the most energy-efficient route. The homoskedastic models have lower estimated regret than the Bayesian models with equivalent prior in both scenarios - except when using the full prior. As in previous experiments, we should not over interpret the estimated regret since the models have not necessarily converged.



(a)



(b)

Figure 4.9: Random regret for homoskedastic GP and Bayesian inference model using the three priors, Optimistic, Uniform and Full, on the a) Luxembourg A and b) Monaco scenarios.

Inspecting the random regret in Figure 4.9, the optimistic Bayesian model has the significantly highest random regret in both scenarios, but the margin to the uniform Bayesian model has narrowed. The homoskedastic models, including the full prior, have lower random regret than their Bayesian baseline in both scenarios. For the Bayesian baselines, priors with more information perform better. However, for the homoskedastic model, the optimistic prior achieves lower random regret than the uniform prior in both scenarios by a small margin.

To better understand the impact of the priors, we visualize the chosen routes across all runs for each combination of prior and model. The chosen routes in Luxembourg A and Monaco is shown in Figure 4.10 and Figure 4.11 respectively. The visualizations are cropped to only show the selected routes. In addition, the total number of visited edges for the combinations of priors and models are reported in Table 4.5.

In Luxembourg A, a quick glance reveals that the Bayesian models explore more than the GP models with equivalent prior. The Bayesian model with optimistic prior explores the most and selects long routes (or detours) that are unlikely to be energy-efficient simply due to their length. Comparatively, the optimistic GP model does not explore excessively. The uniform Bayesian model also explores large parts of the network that seem unpromising but to a lesser degree. In both Figure 4.10 and Table 4.5, we observe that the exploration of the uniform GP model is very similar to the optimistic GP with only minor differences in Luxembourg A. With the full prior, the difference in exploration between the GP and Bayesian model is similar. Still, a close inspection reveals that the Bayesian model explores parts of

Table 4.5: Total number of edges visited across 5 runs with horizon $T = 200$ on Luxembourg A and Monaco.

Model	Luxembourg A, Prior			Monaco, Prior		
	Opti.	Unif.	Full	Opti.	Unif.	Full
Bayesian	1697	996	716	522	464	388
GP	526	519	496	444	365	288

the network that the GP model does not, which is evident from Table 4.5. Using the full prior helps both models focus on exploring the most promising routes in the network. The most frequently driven route is almost identical for all six models, with only minor deviations.

In Monaco, we can observe the same general patterns. The Bayesian models explore more and all models explore less when the priors are more informative. The GP and Bayesian model with full prior explore only central Monaco and avoid the northwestern hills. However, the full Bayesian model still explores the central region excessively and is unable to achieve lower estimated regret than the GP with optimistic and uniform priors.

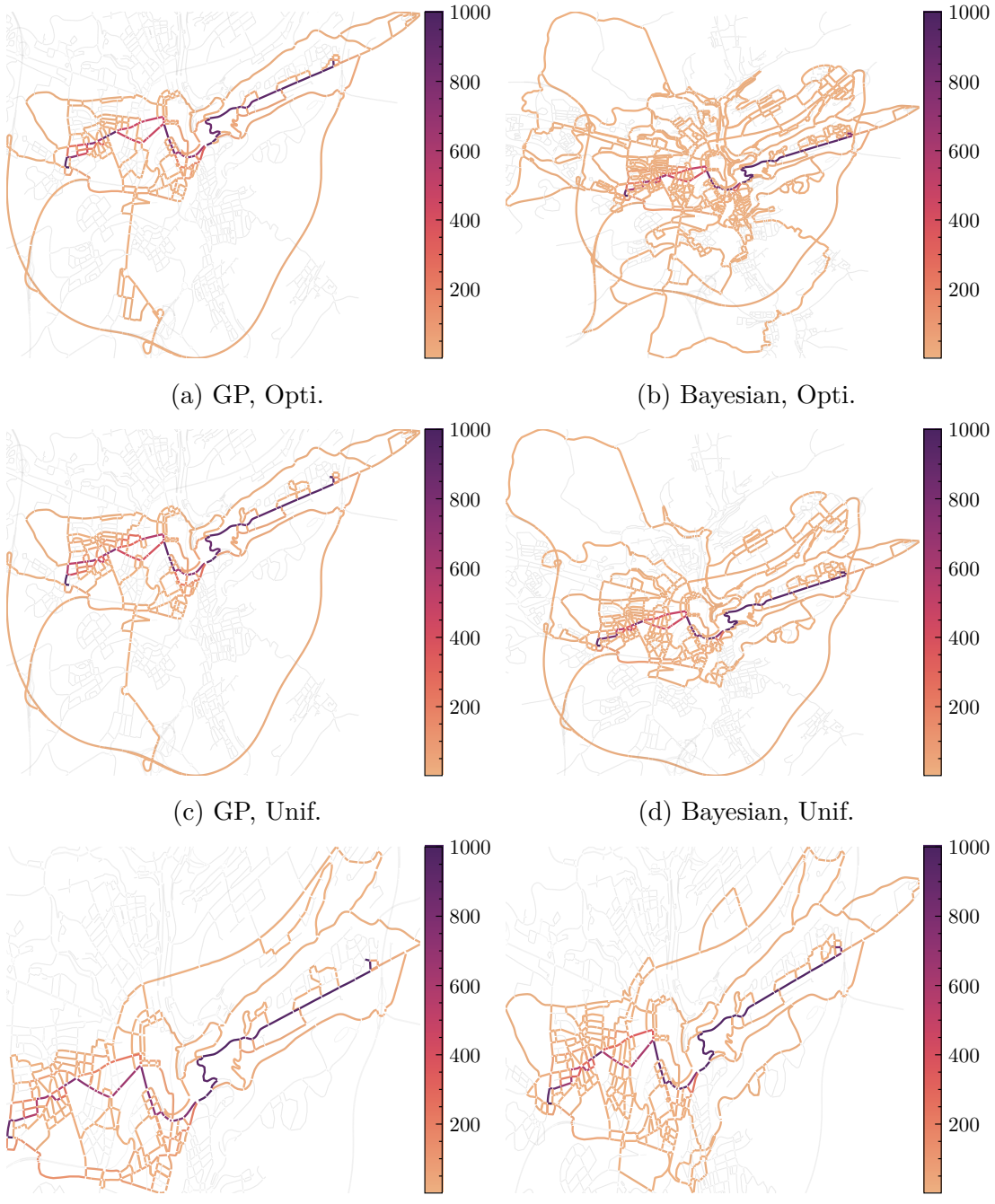
4.5 Comparison of bandit algorithm

To answer Q5, what is the impact of informative priors on the GPs' performance and exploration, we compare Thompson sampling and UCB with the GP model on Luxembourg B and Monaco.

The estimated regret of both bandit algorithms is visualized in Figure 4.12. In the start, both algorithms have similar estimated regret but at $t = 50$ the regret diverges in both scenarios. By the end, UCB has a significantly higher estimated regret than Thompson sampling. In the Monaco scenario, a large increase in estimated regret can be observed around $t = 50$ for both algorithms which is likely due to the exploration of routes in the north-west.

In Figure 4.13, the random regret of UCB and Thompson sampling in the two scenarios is reported. The random regret is mostly consistent with the estimated regret in Figure 4.12. However, in the Monaco scenario, the random regret between the two algorithms diverges immediately. In the Monaco scenario, the random regret for both UCB and Thompson sampling has a higher slope in the beginning than in the end - suggesting that neither algorithm has converged.

In Figure 4.14, the chosen routes for the two bandit algorithms are visualized. At a quick glance, both algorithms explore a similar amount of the network. However, upon closer inspection, the UCB algorithm seems to explore a couple of longer routes which Thompson sampling does not. In both scenarios, Thompson sampling identifies a preferred route that is run most frequently and where only minor deviations are tested. UCB partially finds a preferred route but attempts two or more variants



(a) GP, Opti.

(b) Bayesian, Opti.

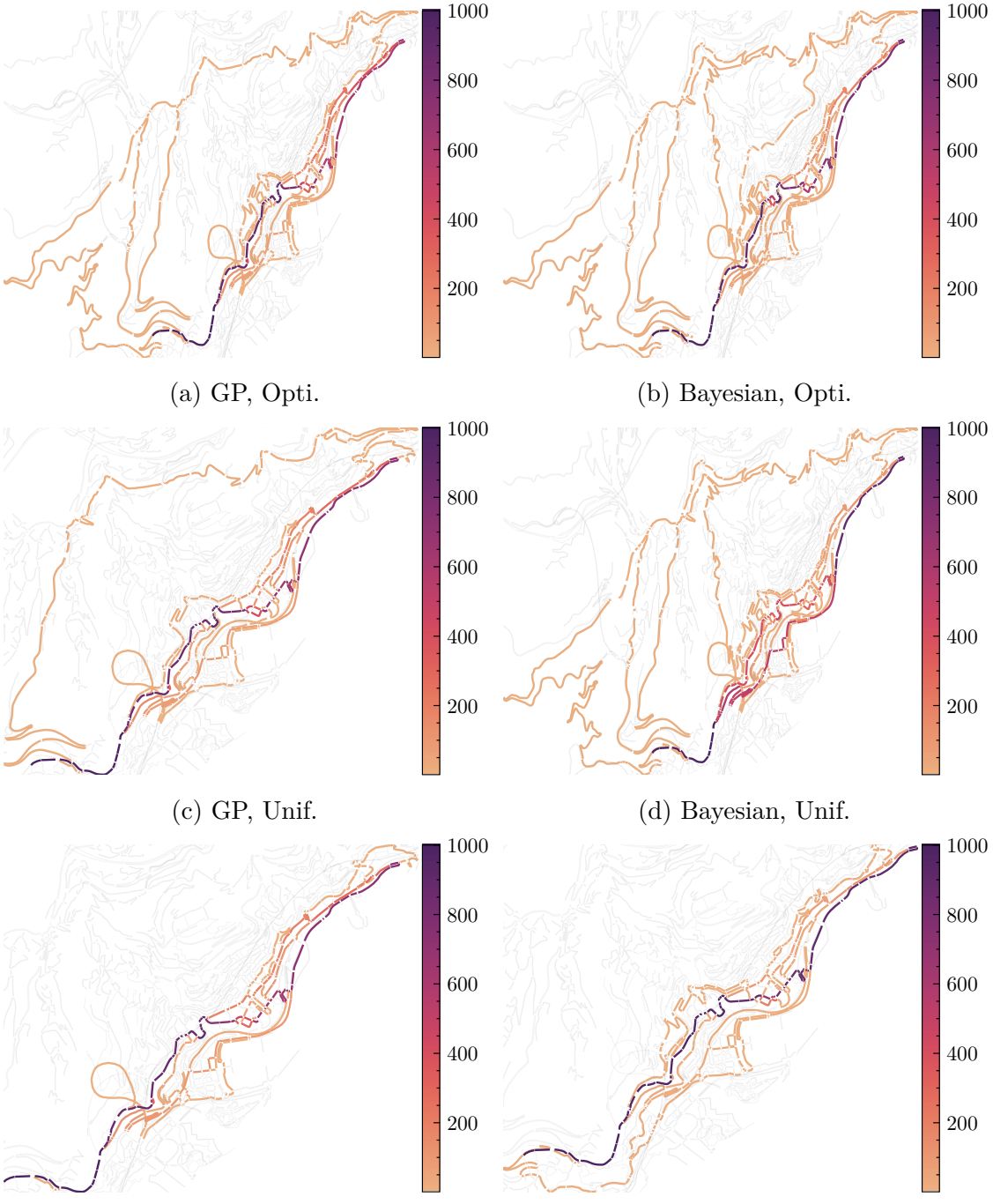
(c) GP, Unif.

(d) Bayesian, Unif.

(e) GP, Full.

(f) Bayesian, Full.

Figure 4.10: All selected routes across 5 runs on Lust A with the GP and Bayesian models using different priors. Edges are colored based on the number of traversals.



(a) GP, Opti.

(b) Bayesian, Opti.

(c) GP, Unif.

(d) Bayesian, Unif.

(e) GP, Full.

(f) Bayesian, Full.

Figure 4.11: All selected routes across 5 runs on Monaco with the GP and Bayesian models using different priors. Edges are colored based on the number of traversals.

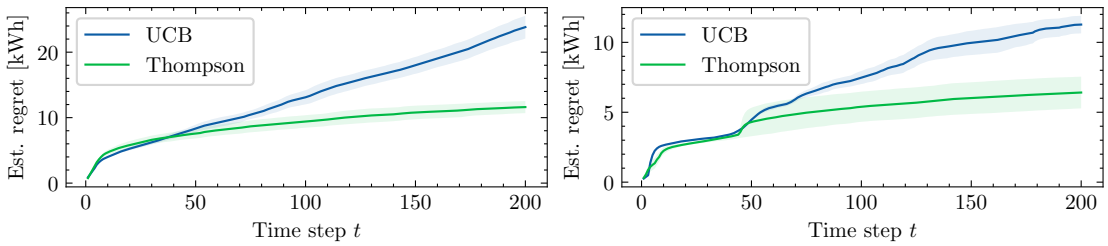


Figure 4.12: Estimated regret on Luxembourg B (left) and Monaco (right).

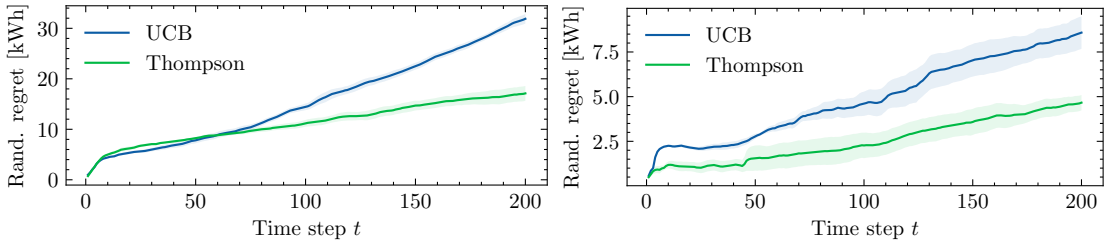
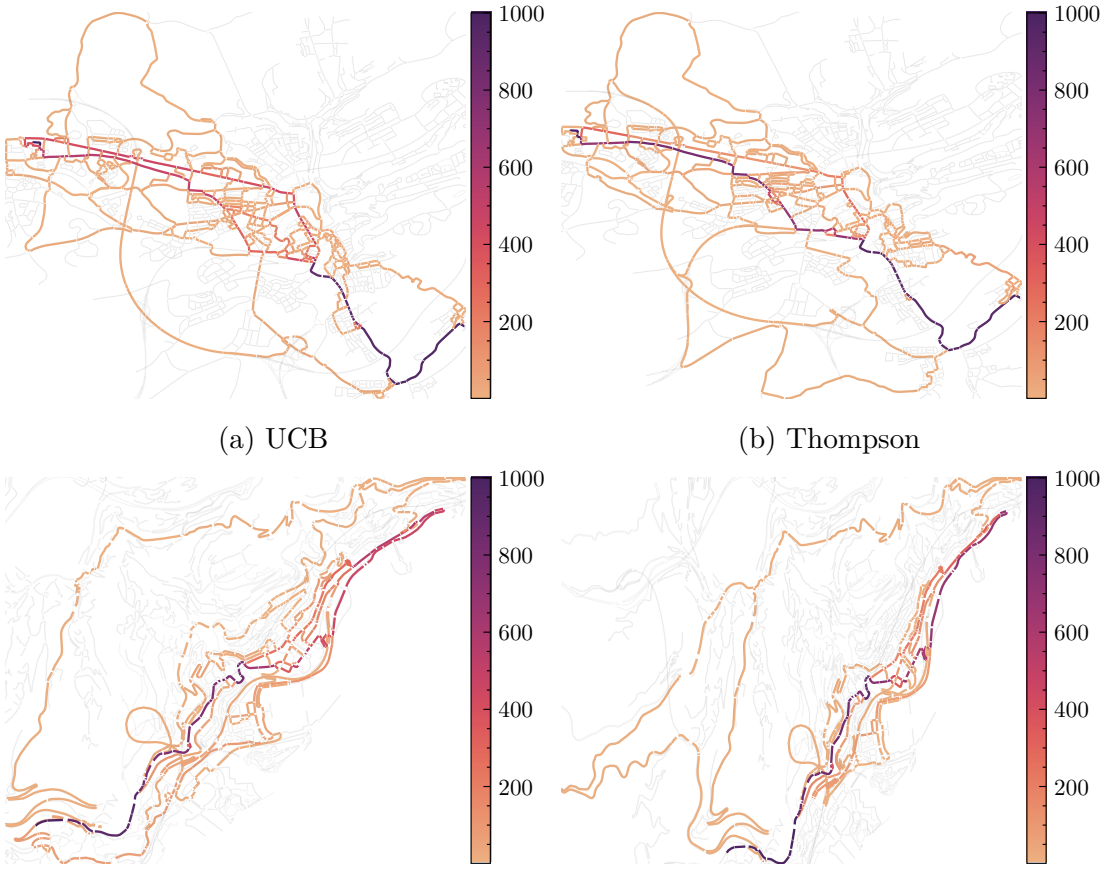


Figure 4.13: Random regret on Luxembourg B (left) and Monaco (right).

in equal proportions. In Table 4.6, the total number of edges visited for the two algorithms is presented. As could be observed in Figure 4.14, UCB explores slightly more edges in both scenarios.

Table 4.6: Total number of edges visited across 5 runs with horizon $T = 200$ on Luxembourg B and Monaco.

Scenario	UCB	Thompson
Luxembourg B	810	772
Monaco	547	385



(a) UCB (b) Thompson

(c) UCB (d) Thompson

Figure 4.14: All selected routes across 5 runs on Luxembourg B and Monaco with the GP using different bandit algorithms.

5

Conclusion

This chapter discusses the results presented in Chapter 4 and provides a conclusion for the thesis.

5.1 Discussion

In this section, we discuss the obtained results, the methodology, and the proposed Gaussian process.

5.1.1 Kernel design

The results in Section 4.1 demonstrate that the feature kernel combined with the graph kernel can efficiently learn to find energy-efficient routes. In the synthetic road network, the feature kernel is shown to be critical, whereas the combined feature and graph kernel $k_{G \cdot f + f}$ is a minor improvement.

The graph kernel requires discrete inputs which introduces difficulties in the optimization of the inducing points. Our approach in Algorithm 5 is to set the inducing points (edge index and features) to the top M most frequently visited edges and then optimize the feature dimensions of the inducing points using natural gradient descent. The benefit of this heuristic approach is that the discrete inducing dimension is set to relevant edges. However, this is potentially a limiting heuristic that could:

1. Make the predictions unstable if the most frequently visited edges shift too often,
2. Cluster the inducing points too close in the graph and,
3. Reset the progress of continuous inducing feature optimization.

From Tables 4.5 and 4.6, the GP model did not explore more edges than the number of inducing points and the risk of these three potential issues should be limited - but this may not be true for much larger networks.

Work such as [52] propose and test methods of optimizing discrete inducing points. Since we only consider static features of the edges, the features and inducing points could be considered completely discrete. Applying the methods of [52] could have provided a more elegant optimization scheme.

When only using the feature kernel, the heuristic is not necessary. Removing the heuristic and only optimizing the inducing points $\mathbf{z} \in \mathbb{R}^d$ could improve the results of the feature kernel.

In [1], the graph kernels was used to estimate the average speed on segments of the road network. In our GP model, we combine a graph and feature kernel to estimate energy consumption instead. Another usage of the graph kernel could have been to estimate the average speed based on the observed speed of our vehicle. The estimated speed could then be fed to the feature kernel. The graph kernel assumes that the road segments in close proximity have similar energy consumption. By instead estimating the average speed with the graph kernel, we make the more reasonable assumption that road segments in close proximity have similar relative speeds or congestion levels.

The feature kernel was also only given the length, speed limit and incline as features. Additional features such as number of lanes, curvature or presence of traffic lights could also improve the feature kernel and allow it to emulate the graph kernel. Initially, the intention was to test the importance of these different features but it is left for future work.

5.1.2 Inducing points

The results in Section 4.2 demonstrate that the cumulative regret increases as the number of inducing points decreases which is to be expected since the accuracy of the GP decreases. However, the GP model outperforms the Bayesian inference model in the synthetic scenario until the relative number of inducing points is below 20%.

The two simulated road networks have 5779 and 3613 edges, respectively. From the results in the synthetic scenarios, 1000 inducing points should be sufficient for the GP model. The data in Tables 4.5 and 4.6 show that the number of explored edges did not exceed 1000 for the GP model in any runs. Thus, the model assigns an inducing point to every visited edge. The non-visited edges thus have a higher uncertainty and will eventually be explored if promising enough.

As discussed in Section 5.1.1, the inducing points are both discrete and continuous. If the inducing points were treated as only continuous (by using only the feature kernel), then the number of inducing points required might be lower due to the simplified optimization procedure.

5.1.3 Noise modeling

Whilst the results on the synthetic scenario showed a significant difference in cumulative regret between the homoskedastic GP and heteroskedastic GP, the random regret was almost identical on Luxembourg A and only slightly better for the homoskedastic GP on Monaco.

As expected, the heteroskedastic model estimated the variance in energy consumption better on both the synthetic and simulated road networks. Modeling the vari-

ance accurately does not improve the average energy consumption but could be useful for suggesting routes with both low average energy consumption and low energy consumption variance.

Out of simplicity, we let the kernel for the mean and variance in the heteroskedastic model be the same. Since most GP applications utilize heteroskedastic noise, we did not encounter any literature discussing how the kernel for the variance should be designed. It is possible that either a different kernel or different features could improve the variance estimation.

5.1.4 Informative prior

The results in Section 3.4 demonstrate that the GP model achieves lower random regret than the Bayesian inference model regardless of the prior. When the prior is uninformative and misleading (such as the optimistic prior), the GP model learns quickly through correlations using the kernel function. The kernel function's length-scale parameter $\boldsymbol{\ell}$ was initialized as 1 for each feature, which is a relatively large value when using standardized features. With a smaller length scale, the GP model would extrapolate less and revert to the mean faster, which could lead to a higher exploration rate.

The relative amount of information gained from using the features decreases when using a more informative prior. This can be observed on Luxembourg A, where the difference in random regret between the GP and Bayesian inference model shrinks as the prior becomes more informative.

For the GP model, the optimistic prior has lower random regret than the uniform prior in both scenarios. The two priors explore almost the same number of edges in Luxembourg A, but the uniform prior explores significantly less in Monaco. It is unclear why the prior uniform performs worse for the GP model; a possibility is that $\overline{E^{\text{det}}}$ is too high since its the average of all edges and introduces a bias in the GP that prevents it from estimating road segments with low energy consumption accurately.

In Figure 4.9, the random regret curves of the GP with the full prior are initially almost flat but the slope later increases. This indicates either that the GP has not converged or that it converges to a suboptimal route. If the GP has not converged, it could be due to instability in Algorithm 5 or that the time horizon of $T = 200$ is too short.

5.1.5 Bandit algorithm

In Section 4.5, the results showed that Thompson sampling achieves both lower estimated and random regret compared to UCB, which is consistent with other empirical studies on bandit problems [10], [24], [53]. The difference in random regret between the two bandit algorithms increased steadily in both road networks. Inspecting the selected routes and the number of edges explored, UCB explores more and does not converge to a single route as quickly as Thompson sampling. To answer

Q5, from the results, the choice of bandit algorithm impacts the performance of the GP model, and Thompson sampling is the favorable algorithm.

5.2 Conclusion

In this thesis, we demonstrated that GPs could increase the data efficiency of the online learning framework for energy-efficient navigation of electric vehicles introduced in [10] by; representing the road network as a line graph, extending the Graph Matérn GP of [1] to directed graphs, and using a feature kernel.

We found that the feature kernel is necessary for efficiently learning the energy consumption in an online setting and found that the graph kernel can yield a minor improvement. It was demonstrated that heteroskedastic modeling of the noise estimated the variance better with either small or no degradation in regret but yields a more complex and computationally demanding GP. Informative priors lead to less exploration and improve the performance of the GP model. However, the GP model can overcome a misleading prior efficiently since it uses the correlations between the edges. Finally, Thompson sampling had lower regret than UCB for the GP model.

Bibliography

[1] V. Borovitskiy, I. Azangulov, A. Terenin, P. Mostowsky, M. Deisenroth, and N. Durrande, "Matérn Gaussian Processes on Graphs," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, A. Banerjee and K. Fukumizu, Eds., ser. Proceedings of Machine Learning Research, vol. 130, PMLR, Apr. 2021, pp. 2593–2601. [Online]. Available: https://proceedings.mlr.press/v130/borovitskiy21a.html.

[2] H. Ritchie, M. Roser, and P. Rosado, "CO and Greenhouse Gas Emissions," *Our World in Data*, 2020. [Online]. Available: https://ourworldindata.org/co2-and-greenhouse-gas-emissions.

[3] "World Energy Outlook 2022," en-GB, IEA, Paris, Tech. Rep., 2022. [Online]. Available: https://www.iea.org/reports/world-energy-outlook-2022 (visited on 04/04/2023).

[4] "Global Electric Vehicle Outlook 2022," en, IEA, Paris, Tech. Rep., 2022. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2022.

[5] J. Holden, N. Reinicke, and J. Cappellucci, "RouteE: A Vehicle Energy Consumption Prediction Engine," Apr. 2020, pp. 2020–01–0939. DOI: 10.4271/2020-01-0939. [Online]. Available: https://www.sae.org/content/2020-01-0939/ (visited on 05/18/2023).

[6] A. Brooker, J. Gonder, L. Wang, E. Wood, S. Lopp, and L. Ramroth, "FASTSim: A Model to Estimate Vehicle Efficiency, Cost and Performance," Apr. 2015, pp. 2015–01–0973. DOI: 10.4271/2015-01-0973. [Online]. Available: https://www.sae.org/content/2015-01-0973/ (visited on 05/18/2023).

[7] Google, "Google Maps Eco Friendly Routing: Accelerating the journey to sustainability for one billion people," en, 2021. [Online]. Available: https://www.gstatic.com/gumdrop/sustainability/google-maps-eco-friendly-routing.pdf (visited on 05/18/2023).

[8] A. Artmeier, J. Haselmayr, M. Leucker, and M. Sachenbacher, "The Shortest Path Problem Revisited: Optimal Routing for Electric Vehicles," en, in *KI 2010: Advances in Artificial Intelligence*, R. Dillmann, J. Beyerer, U. D. Hanebeck, and T. Schultz, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2010, pp. 309–316, ISBN: 978-3-642-16111-7. DOI: 10.1007/978-3-642-16111-7_35.

[9] R. Abousleiman and O. Rawashdeh, "A Bellman-Ford approach to energy efficient routing of electric vehicles," in *2015 IEEE Transportation Electrification Conference and Expo (ITEC)*, Jun. 2015, pp. 1–4. DOI: 10.1109/ITEC.2015.7165772.

[10] N. Åkerblom, Y. Chen, and M. Haghir Chehreghani, "Online Learning of Energy Consumption for Navigation of Electric Vehicles," en, *Artificial Intelligence*, p. 103 879, Feb. 2023, ISSN: 0004-3702. DOI: 10.1016/j.artint.2023.103879. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370223000255 (visited on 02/10/2023).

[11] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian Processes for Big Data," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'13, event-place: Bellevue, WA, Arlington, Virginia, USA: AUAI Press, 2013, pp. 282–290.

[12] D. E. W, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959. [Online]. Available: https://cir.nii.ac.jp/crid/1571698600523742592 (visited on 03/27/2023).

[13] A. Shimbel, "Structure in communication nets," *Proceedings of the Symposium on Information Networks*, pp. 119–203, 1954, Publisher: Polytechnic Institute of Brooklyn. [Online]. Available: https://cir.nii.ac.jp/crid/1573950399431729664 (visited on 04/17/2023).

[14] R. Bellman, "On a routing problem," en, *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958, ISSN: 0033-569X, 1552-4485. DOI: 10.1090/qam/102435. [Online]. Available: https://www.ams.org/qam/1958-16-01/S0033-569X-1958-0102435-2/ (visited on 04/17/2023).

[15] L. R. Ford, "Network Flow Theory.," en, RAND Corporation, Tech. Rep., Jan. 1956. [Online]. Available: https://www.rand.org/pubs/papers/P923.html (visited on 04/17/2023).

[16] A. Bjorklund, T. Husfeldt, and S. Khanna, "Approximating longest directed paths and cycles," in *ICALP*, vol. 3142, Springer, 2004, pp. 222–233.

[17] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction* (Adaptive computation and machine learning series), en, Second edition. Cambridge, Massachusetts: The MIT Press, 2018, ISBN: 978-0-262-03924-6.

[18] H. Robbins, "Some aspects of the sequential design of experiments," *Herbert Robbins Selected Papers*, pp. 169–177, 1985.

[19] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial Multi-Armed Bandit: General Framework and Applications," en, in *Proceedings of the 30th International Conference on Machine Learning*, ISSN: 1938-7228, PMLR, Feb. 2013, pp. 151–159. [Online]. Available: https://proceedings.mlr.press/v28/chen13a.html (visited on 04/27/2023).

[20] N. Cesa-Bianchi and G. Lugosi, "Combinatorial bandits," en, *Journal of Computer and System Sciences*, JCSS Special Issue: Cloud Computing 2011, vol. 78, no. 5, pp. 1404–1422, Sep. 2012, ISSN: 0022-0000. DOI: 10.1016/j.jcss.2012.01.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022000012000219 (visited on 04/27/2023).

[21] L. Zhou, *A Survey on Contextual Multi-armed Bandits*, arXiv:1508.03326 [cs], Feb. 2016. DOI: 10.48550/arXiv.1508.03326. [Online]. Available: http://arxiv.org/abs/1508.03326 (visited on 04/27/2023).

[22] A. Nika, S. Elahi, and C. Tekin, *Contextual Combinatorial Bandits with Changing Action Sets via Gaussian Processes*, arXiv:2110.02248 [cs, stat], Oct. 2022.

DOI: 10.48550/arXiv.2110.02248. [Online]. Available: http://arxiv.org/abs/2110.02248 (visited on 01/23/2023).

[23] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, Dec. 1933, ISSN: 0006-3444. DOI: 10.1093/biomet/25.3-4.285. [Online]. Available: https://doi.org/10.1093/biomet/25.3-4.285 (visited on 04/27/2023).

[24] O. Chapelle and L. Li, "An Empirical Evaluation of Thompson Sampling," in *Advances in Neural Information Processing Systems*, vol. 24, Curran Associates, Inc., 2011. [Online]. Available: https://papers.nips.cc/paper_files/paper/2011/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html (visited on 04/27/2023).

[25] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, 2010, pp. 1015–1022, ISBN: 978-1-60558-907-7.

[26] E. Kaufmann, O. Cappe, and A. Garivier, "On Bayesian Upper Confidence Bounds for Bandit Problems," en, in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, ISSN: 1938-7228, PMLR, Mar. 2012, pp. 592–600. [Online]. Available: https://proceedings.mlr.press/v22/kaufmann12.html (visited on 04/27/2023).

[27] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[28] M. Titsias, "Variational Learning of Inducing Variables in Sparse Gaussian Processes," en, in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, ISSN: 1938-7228, PMLR, Apr. 2009, pp. 567–574. [Online]. Available: https://proceedings.mlr.press/v5/titsias09a.html (visited on 01/16/2023).

[29] D. J. C. Mackay, "Introduction to Gaussian processes," en, in *NATO ASI series. Series F : computer and system sciences*, ISSN: 0258-1248, 1998, pp. 133–165, ISBN: 978-3-540-64928-1. [Online]. Available: http://pascal-francis.inist.fr/vibad/index.php?action=getRecordDetail&idt=1572419 (visited on 02/28/2023).

[30] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/27e8e17134dd7083b050476733207ea1-Abstract.html (visited on 01/25/2023).

[31] R. M. Neal, *Bayesian Learning for Neural Networks* (Lecture Notes in Statistics), P. Bickel, P. Diggle, S. Fienberg, *et al.*, Eds. New York, NY: Springer, 1996, vol. 118, ISBN: 978-1-4612-0745-0. DOI: 10.1007/978-1-4612-0745-0. [Online]. Available: http://link.springer.com/10.1007/978-1-4612-0745-0 (visited on 04/20/2023).

[32] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951, Publisher: Institute of Mathematical Statistics, ISSN: 0003-4851. [Online]. Available: https://www.jstor.org/stable/2236703 (visited on 05/29/2023).

[33] J. M. Joyce, "Kullback-Leibler Divergence," en, in *International Encyclopedia of Statistical Science*, M. Lovric, Ed., Berlin, Heidelberg: Springer, 2011, pp. 720–722, ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_327. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_327 (visited on 05/29/2023).

[34] S. Ruder, *An overview of gradient descent optimization algorithms*, arXiv:1609.04747 [cs], Jun. 2017. DOI: 10.48550/arXiv.1609.04747. [Online]. Available: http://arxiv.org/abs/1609.04747 (visited on 04/19/2023).

[35] V. Borovitskiy, A. Terenin, P. Mostowsky, and M. Deisenroth (he/him), "Matérn Gaussian Processes on Riemannian Manifolds," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 12 426–12 437. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/92bf5e6240737e0326ea59846a83e076-Abstract.html (visited on 02/03/2023).

[36] P. Whittle, "Stochastic-processes in several dimensions," *Bulletin of the International Statistical Institute*, vol. 40, no. 2, pp. 974–994, 1963, Publisher: Int Statistical Institute 428 Prinses Beatrixlaen, Voorburg, Netherlands.

[37] R. Horaud and R. Horaud, "A Short Tutorial on Graph Laplacians, Laplacian Embedding, and Spectral Clustering," en, 2009.

[38] N. Li, W. Li, J. Sun, Y. Gao, Y. Jiang, and S.-T. Xia, "Stochastic Deep Gaussian Processes over Graphs," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 5875–5886. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/415e1af7ea95f89f4e375162b21ae38c-Abstract.html (visited on 01/30/2023).

[39] Y. C. Ng, N. Colombo, and R. Silva, "Bayesian Semi-supervised Learning with Graph Gaussian Processes," in *Advances in Neural Information Processing Systems*, vol. 31, Curran Associates, Inc., 2018. [Online]. Available: https://papers.nips.cc/paper/2018/hash/1fc214004c9481e4c8073e85323bfd4b-Abstract.html (visited on 01/19/2023).

[40] F. Opolka, Y.-C. Zhi, P. Lió, and X. Dong, "Adaptive Gaussian Processes on Graphs via Spectral Graph Wavelets," en, in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, ISSN: 2640-3498, PMLR, May 2022, pp. 4818–4834. [Online]. Available: https://proceedings.mlr.press/v151/opolka22a.html (visited on 05/19/2023).

[41] A. Venkitaraman, S. Chatterjee, and P. Handel, "Gaussian Processes Over Graphs," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5640–5644. DOI: 10.1109/ICASSP40776.2020.9053859.

[42] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[43] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[44] H. Salimbeni, S. Eleftheriadis, and J. Hensman, "Natural Gradients in Practice: Non-Conjugate Variational Inference in Gaussian Process Models," in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, A. Storkey and F. Perez-Cruz, Eds., ser. Proceedings of Machine Learning Research, vol. 84, PMLR, Apr. 2018, pp. 689–697. [Online]. Available: https://proceedings.mlr.press/v84/salimbeni18a.html.

[45] A. G. d. G. Matthews, M. v. d. Wilk, T. Nickson, *et al.*, "GPflow: A Gaussian Process Library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, no. 40, pp. 1–6, 2017, ISSN: 1533-7928. [Online]. Available: http://jmlr.org/papers/v18/16-537.html (visited on 01/25/2023).

[46] P. A. Lopez, M. Behrisch, L. Bieker-Walz, *et al.*, "Microscopic Traffic Simulation using SUMO," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, ISSN: 2153-0017, Nov. 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

[47] L. Codeca and J. Härri, "Monaco SUMO Traffic (MoST) Scenario: A 3D Mobility Scenario for Cooperative ITS," in *SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems, May 14-16, 2018, Berlin, Germany*, Berlin, GERMANY, May 2018.

[48] L. Codeca, R. Frank, S. Faye, and T. Engel, "Luxembourg SUMO Traffic (LuST) Scenario: Traffic Demand Evaluation," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 52–63, 2017, Conference Name: IEEE Intelligent Transportation Systems Magazine, ISSN: 1941-1197. DOI: 10.1109/MITS.2017.2666585.

[49] A. de la navigation aérienne, *Digital Terrain Model (high DEM resolution) - Portail Open Data*, en. [Online]. Available: https://data.public.lu/en/datasets/digital-terrain-model-high-dem-resolution/ (visited on 04/17/2023).

[50] QGIS Development Team, *QGIS Geographic Information System*. QGIS Association, 2023. [Online]. Available: https://www.qgis.org.

[51] M. Rapelli, C. Casetti, and G. Gagliardi, "Vehicular Traffic Simulation in the City of Turin From Raw Data," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4656–4666, Dec. 2022, Conference Name: IEEE Transactions on Mobile Computing, ISSN: 1558-0660. DOI: 10.1109/TMC.2021.3075985.

[52] V. Fortuin, G. Dresdner, H. Strathmann, and G. Rätsch, "Sparse Gaussian Processes on Discrete Domains," *IEEE Access*, vol. 9, pp. 76 750–76 758, 2021, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3082761.

[53] S. Wang and W. Chen, "Thompson Sampling for Combinatorial Semi-Bandits," en, in *Proceedings of the 35th International Conference on Machine Learning*, ISSN: 2640-3498, PMLR, Jul. 2018, pp. 5114–5122. [Online]. Available: https://proceedings.mlr.press/v80/wang18a.html (visited on 05/24/2023).

[54] C. P. Robert, *The Bayesian Choice* (Springer Texts in Statistics), en. New York, NY: Springer, 2007, ISBN: 978-0-387-71598-8. DOI: 10.1007/0-387-71599-1. [Online]. Available: http://link.springer.com/10.1007/0-387-71599-1 (visited on 05/25/2023).

[55] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, en, 3rd. CRC press, 2013.

[56] Y. L. Tong, *The Multivariate Normal Distribution* (Springer Series in Statistics). New York, NY: Springer, 1990, ISBN: 978-1-4613-9655-0. DOI: 10.1007/978-1-4613-9655-0. [Online]. Available: http://link.springer.com/10.1007/978-1-4613-9655-0 (visited on 02/28/2023).

[57] N. Socci, D. Lee, and H. S. Seung, "The Rectified Gaussian Distribution," in *Advances in Neural Information Processing Systems*, vol. 10, MIT Press, 1997. [Online]. Available: https://papers.nips.cc/paper_files/paper/1997/hash/28fc2782ea7ef51c1104ccf7b9bea13d-Abstract.html (visited on 04/27/2023).

[58] M. Harva and A. Kabán, "Variational learning for rectified factor analysis," en, *Signal Processing*, vol. 87, no. 3, pp. 509–527, Mar. 2007, ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2006.06.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0165168406002118 (visited on 04/27/2023).

[59] M. Beauchamp, "On numerical computation for the distribution of the convolution of N independent rectified Gaussian variables," en, *Journal de la Société Française de Statistique*, vol. 159, no. 1, pp. 88–111, Mar. 2018, Number: 1, ISSN: 2102-6238. [Online]. Available: http://journal-sfds.fr/article/view/669 (visited on 04/27/2023).

A

Appendix

In this chapter, we provide a brief description of Bayesian inference, the multivariate normal distribution and the rectified normal distribution.

A.1 Bayesian inference

In this section, we provide a short introduction to Bayesian inference and conjugate priors.

In Bayesian inference, we are typically interested in estimating a parameter θ of some probability distribution p_θ. Unlike classical inference, we assume a *prior belief* of θ's distribution that incorporates our prior knowledge of the parameter and the context of the problem. Prior knowledge could be results from similar experiments or limitations of the parameter due to physics.

Let $x_1, \ldots, x_n$ denote n i.i.d. samples from the distribution p_θ. Then, a *posterior belief* of θ's distribution can be obtained by combining the observed data and the priors using Bayes' theorem

$$p(\theta|x_1, \ldots, x_n) = \frac{p(x_1, \ldots, x_n|\theta)p(\theta)}{p(x_1, \ldots, x_n)} = \frac{p(x_1, \ldots, x_n|\theta)p(\theta)}{\int_\theta p(x_1, \ldots, x_n|\theta)p(\theta)\mathrm{d}\theta} \tag{A.1}$$

where $p(\theta|x_1, \ldots, x_n)$, $p(x_1, \ldots, x_n|\theta)$ and $p(\theta)$ denotes the posterior, likelihood and prior respectively. The denominator can be thought of as a normalizing constant without a θ-dependency, and the posterior can be expressed as

$$p(\theta|x_1, \ldots, x_n) \propto_\theta p(x_1, \ldots, x_n|\theta)p(\theta). \tag{A.2}$$

Computing the normalizing constant can be intractable even with state-of-the-art computational techniques if θ is high-dimensional. However, suppose one can identify the distribution type and parameters of the explicit expression $p(x_1, \ldots, x_n|\theta)p(\theta)$. In that case, the normalizing constant can be looked up in a reference table. For example, try to identify the distribution and parameters of the expression below:

$$p(\theta|x_1, \ldots, x_n) \propto_\theta e^{-\frac{(x-3)^2}{2 \cdot 5^2}}. \tag{A.3}$$

If the posterior is within the same distribution family as the prior, then the prior $p(\theta)$ is called a conjugate prior to the likelihood $p(x|\theta)$. For example, if the prior

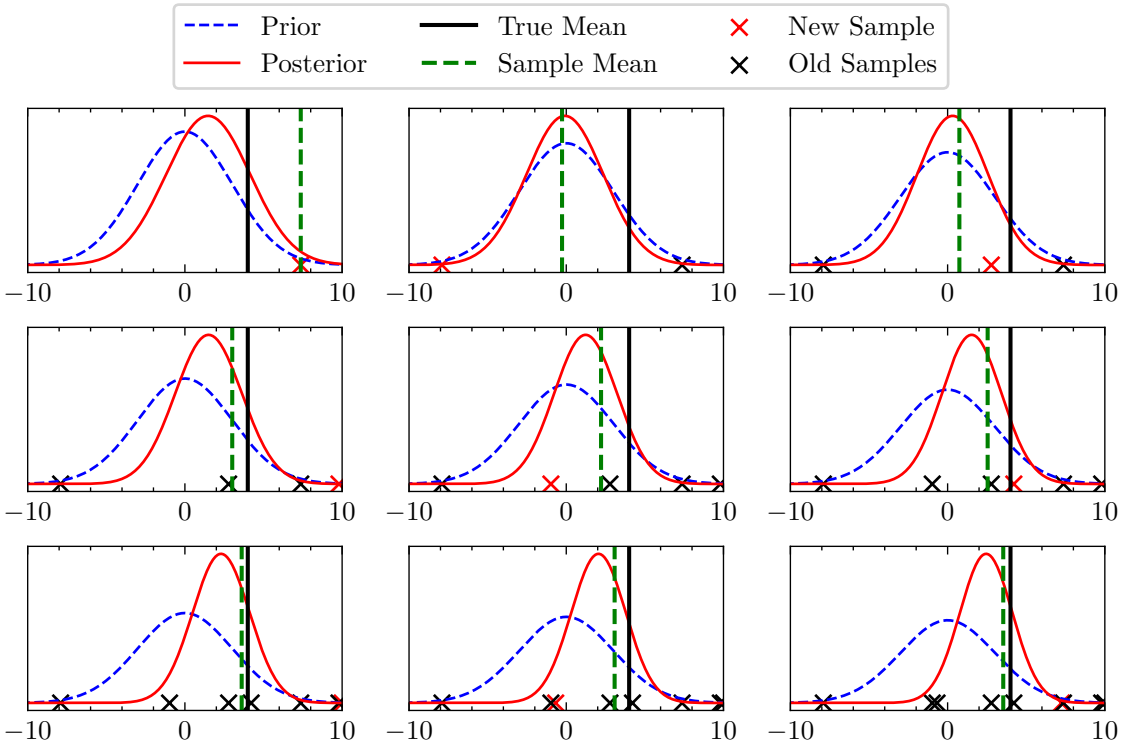


Figure A.1: Visualization of Bayesian inference updates.

and likelihood are both normal, then the posterior will be normal. Simple update rules can often be derived for the posterior parameters using conjugate priors, such as in Algorithm 2.

In Figure A.1, we visualize how the posterior changes as we sample from the true distribution $\mathcal{N}(4, 4^2)$. Initially, the posterior is close to the prior $\mathcal{N}(0, 3^2)$, but as the number of data points increases, the variance of the posterior decreases, and the mean tends to the true mean.

For a complete and thorough treatment of Bayesian inference, see [54], [55].

A.2 Multivariate normal

The multivariate normal distribution (or multivariate Gaussian distribution) is a generalization of the univariate normal distribution from $\mathbb{R}$ to $\mathbb{R}^n$. This section discusses properties of the multivariate normal distribution useful for this thesis. The definitions and results presented in this section are all based on [56].

The multivariate normal distribution can be defined as follows:

Definition 1. *An n-dimensional random variable $\mathbf{X}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is said to have a non-singular multivariate normal distribution if (i) $\boldsymbol{\Sigma}$ is positive definite, and (ii) the density function of $\mathbf{X}$ is of the form*

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right). \tag{A.4}$$

In this thesis we will mainly consider non-singular multivariate normal distributions and will refer to them as multivariate normal distributions. If not otherwise stated we will assume that $\mathbf{\Sigma}$ is positive definite.

We use the notation $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ to denote a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\mathbf{\Sigma}$.

The multivariate normal distribution is the backbone of this thesis. In subsequent sections, we will generalize the multivariate Gaussian, but correctly manipulating multivariate Gaussians will remain an important skill. The first such manipulation we will consider is conditioning a Gaussian.

Consider the following partition of a multivariate normal distribution:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix}, \quad \boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{21} & \mathbf{\Sigma}_{22} \end{pmatrix} \tag{A.5}$$

where $\mathbf{X}_1$ denotes the first $k < n$ variables of $\mathbf{X}$.

Theorem 1. *(Conditional Gaussian distribution) Let* $\mathbf{X}$ *be partitioned as in Equation* (A.5) *for* $k < n$. *If* $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, *then the conditional distribution of* $\mathbf{X}_1 | \mathbf{X}_2 = \mathbf{x}_2$ *is given by*

$$\mathbf{X}_1 | \mathbf{X}_2 = \mathbf{x}_2 \sim \mathcal{N}(\boldsymbol{\mu}_{1|2}, \mathbf{\Sigma}_{1|2}) \text{ where} \tag{A.6}$$

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2) \tag{A.7}$$

$$\mathbf{\Sigma}_{1|2} = \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}\mathbf{\Sigma}_{21}. \tag{A.8}$$

Theorem 1 is an important theorem as it provides a closed-form expression for a Gaussian distribution conditioned on another Gaussian. We will use Theorem 1 in Section 2.2.2 to derive Gaussian process regression.

Next, we consider how a linear transformation affects the distribution of a multivariate Gaussian:

Theorem 2. *(Linear transformation of multivariate normal distribution) Let* $\mathbf{Y} = \mathbf{C}\mathbf{X} + \mathbf{b}$ *where* $X \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ *and* $\mathbf{C}$ *is an* $n \times n$ *real and invertible matrix, then* $\mathbf{Y} \sim \mathcal{N}(\boldsymbol{\mu}_\mathbf{Y}, \mathbf{\Sigma}_\mathbf{Y})$ *where*

$$\boldsymbol{\mu}_\mathbf{Y} = \mathbf{C}\boldsymbol{\mu} + \mathbf{b}, \quad \mathbf{\Sigma}_\mathbf{Y} = \mathbf{C}\mathbf{\Sigma}\mathbf{C}^\top. \tag{A.9}$$

Applying Theorem 2, we obtain the following corollary:

Corollary 1. *(Cholesky decomposition of covariance matrix)* $\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$ *with* $\mathbf{\Sigma}$ *positive definite holds only if and only if there exists a non-singular* $n \times n$ *matrix* $\mathbf{C}$ *such that*

1. $\mathbf{C}\mathbf{C}^\top = \mathbf{\Sigma}$ *and*
2. $\mathbf{X}$ *and* $\mathbf{C}\mathbf{Z} + \boldsymbol{\mu}$ *are identically distributed if* $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{I})$.

From Corollary 1, we see that sampling from an arbitrary multivariate Gaussian is equivalent to multiplying the Cholesky decomposition with n samples from a unit Gaussian. In Section 2.2.6.1, we will use Corollary 1 to derive a Graph Gaussian process.

For a complete treatment of the multivariate normal distribution and detailed proofs of the theorems, see Chapter 3 of [56].

A.3 Rectified normal distribution

A random variable $Y = \max(0, X)$ is said to have a rectified normal distribution $\mathcal{N}_R(\mu, \sigma^2)$ if the variable $X \sim \mathcal{N}(\mu, \sigma^2)$ [57]. The rectified normal distribution is useful for restricting normally distributed variables to be non-negative. The mean and variance of Y is given by

$$\mathbb{E}[Y] = \mu\left(1 - \Phi\left(-\frac{\mu}{\sigma}\right)\right) + \sigma\phi\left(-\frac{\mu}{\sigma}\right) \tag{A.10}$$

$$\mathrm{Var}(Y) = (\mu^2 + \sigma^2)\left(1 - \Phi\left(-\frac{\mu}{\sigma}\right)\right) + \mu\sigma\phi\left(-\frac{\mu}{\sigma}\right) - \mathbb{E}[Y]^2 \tag{A.11}$$

where Φ and ϕ denote the cumulative distribution function and probability density function, respectively, for the unit Gaussian $\mathcal{N}(0, 1)$ [58], [59].