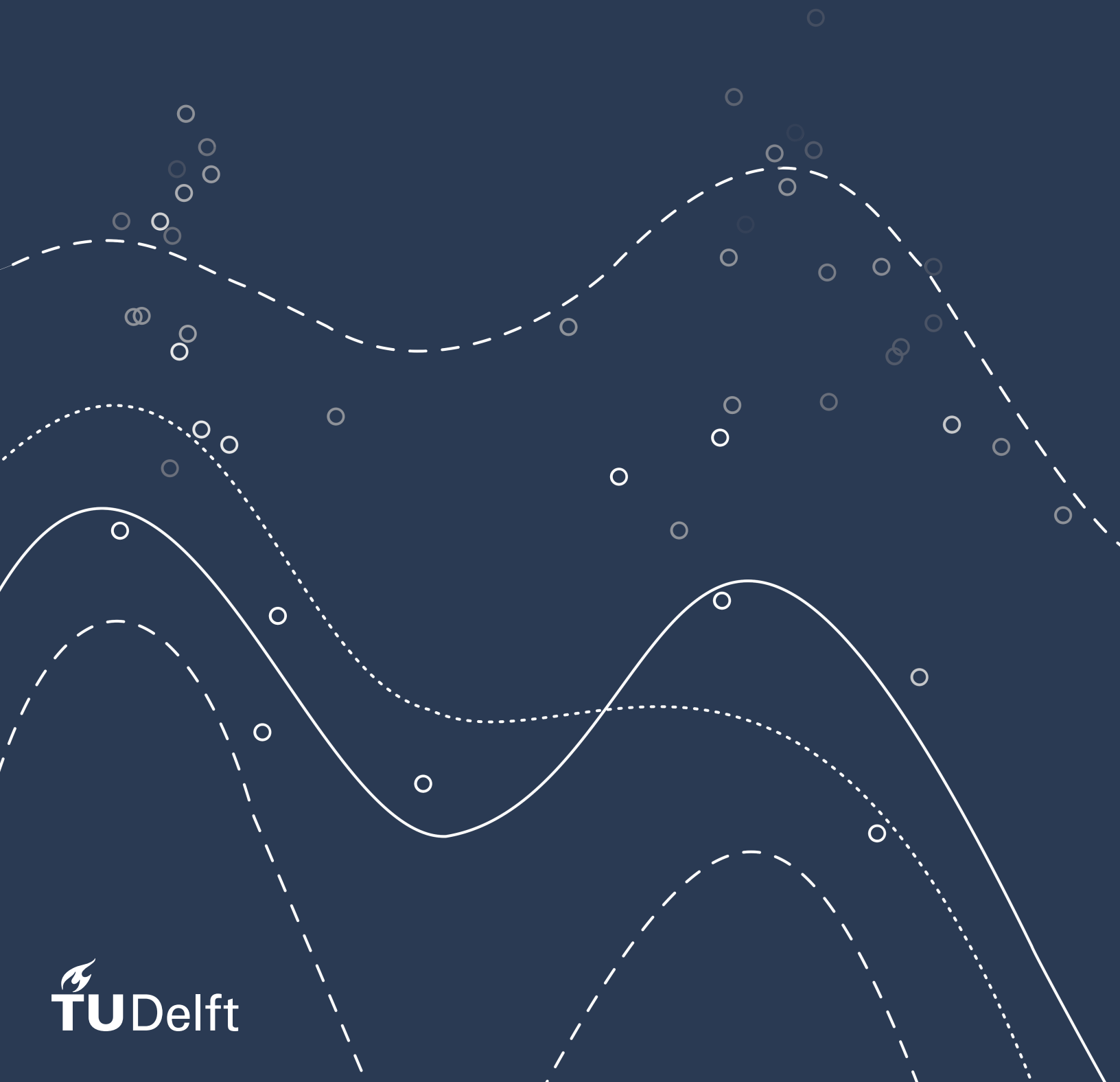
Advancing Gaussian Process Bandit Optimization for Time-Varying Functions

Matthias Mandl



TU Delft

Advancing Gaussian Process Bandit Optimization for Time-Varying Functions

Online Learning in the Continuous Time-Varying Setting

by

Matthias Mandl

Student Number

4789903

Supervisor:	Dr. Hanne Kekkonen
Committee members:	Prof. Dr. Ir. Geurt Jongbloed, Dr. Alexander Heinlein
Project Duration:	March, 2024 - September, 2024
Department:	Delft Institute of Applied Mathematics

TU Delft

Acknowledgements

First and foremost I want to acknowledge Doctor Hanne Kekkonen. She consistently encouraged me to investigate the aspects of the topic that piqued my interest while supporting me with mathematical insights and keeping me on track to produce a complete thesis at the end of the project. Throughout the thesis, she has given me detailed feedback to help me express my ideas in the language of mathematics and turn my thoughts and intuitions into a (hopefully) convincing thesis. Without Hanne this thesis would not have been possible.
I would also like to extend my gratitude to Professor Geurt Jongbloed for all of his support and guidance during this master. Despite his busy schedule Professor Jongbloed always made time to meet with me. He helped me find my path in the world of mathematics at TU Delft.
I want to thank Doctor Alexander Heinlein. He was one of the professors in my very first machine learning course, Linear Algebra and Optimization for Machine Learning, which laid the foundation for many of the techniques used in this thesis.

I would also like to acknowledge my friends and family who have enabled me to develop as a person as well as a mathematician. Specifically, my friends within the Applied Mathematics MSc who have been invaluable in my personal life as well as during my academic transition from the Aerospace Engineering BSc to the AM MSc. My partner Sterre whom I can always count on for support in all of my endeavours. And last but not least, my parents, Gaby and Paul, who have supported me in countless ways.

Matthias Mandl

Delft, September 2024

Abstract

This thesis investigates the problem of time-varying function optimization. In particular, we study techniques to minimize the cumulative regret when optimizing a time-varying function in the Gaussian process setting. First, we introduce the problem and present a literature review of the current methods and results. Following this, we we propose enhancements to existing algorithms, demonstrating improved regret bounds. We discuss the applications of these algorithms and where they can provide a benefit compared to existing methods. With these applications in mind we introduce two new temporal models for time-varying functions and their associated algorithms. We test their performance in order to validate their effectiveness and potential benefits.

Contents

1 Introduction

This thesis is concerned with the topic of online learning and can be viewed as exploring an extension of the multi-armed bandit (MAB) problem. The MAB problem was formulated by Herbert Robbins in 1952 [24] and is described as follows; suppose you are presented with $K \in \mathbb{N}^+$ levers. At each time step you choose one of these levers to pull which results in some reward (drawn from a reward distribution for each lever). Your goal is to maximize the sum of the rewards. In other words, you want to minimize the difference between the rewards which you obtained and the maximum possible reward. This presents a challenge; when you should simply stick with the lever which has given you the highest average reward so far and how often should you choose levers which have given lower rewards in the past (in order to check if you simply got unlucky with the draw from the respective reward distribution)? This is often referred to as the exploration-exploitation trade-off which characterizes these types of problems. It is important to note that the reward distributions of the levers are not affected by our choices. In this paper we are concerned with an alternative setting of this problem which retains some similar properties.

Suppose that instead of choosing from a discrete set of levers we are now concerned with choosing a point on some continuous domain, such that an underlying reward function is maximized. Bayesian optimization (BO) is a widely used tool for efficiently finding the maximum of an unknown function with computationally expensive and noisy evaluations. It aims to estimate the location in the domain at which a function will be maximized while limiting the number of function evaluations. Suppose we want to minimize the regret of our choices, that is; difference between the function value at the evaluated point and the optimal (global maximum of the function over the domain). If we assume that the function is somewhat smooth this presents a dilemma; do we choose to evaluate the function at a point which is close to an already evaluated "good" point or do we evaluate the function at a point which is far from any previously evaluated point; allowing us to learn more about the underlying function and potentially find an even higher maximum than the current best point. This is the exploration-exploitation trade-off in this new setting. This problem has been studied in detail and Bayesian optimization has been used since the 1970's without any guarantees on the regret of these algorithms. In 2012 and Srinivas et al. [28] presented the first regret bounds for BO in the setting where the underlying reward function is sampled from a Gaussian process (GP). GPs are quite flexible which implies that this regret bound could hold for many applications where the true reward function is not known.

One assumption which is often made is that the underlying function is constant and does not change when it is evaluated. However, in many real world scenarios this assumption does not hold. We can imagine that the function which is sampled at the first sample might not be the same function which is sampled later into the optimization process. For a real world example; consider a bird scientist who is aiming to observe as many birds as possible each day within a given forest. Every day the researcher chooses one point in the forest to set up her equipment and make observations of the birds. Her goal is to select a point in the forest with a high density of birds so she can make as many observations as possible. Since she has no prior knowledge of which areas in the forest have the most birds she records the number of birds every day along with the location in order to inform her future choice of location. Over time she can use this data to make better choices. However, we must also consider that the birds behaviour can change over time. This means that she should put more weight on her recent observations, an observation from 100 days ago might not be relevant to predicting the current behaviour of the birds.

In 2016 Bogunovic, Scarlett, and Cevher [2] presented a Time-Varying BO (TVBO) algorithm which is able to accommodate a function which changes over time (between each evaluation) at some rate ε. They also present regret bounds for the performance of their algorithm. This has been used for applications such as online optimization of deep reinforcement learning (DRL) hyperparameters in Parker-Holder et al. [20]. Alternatives to the TVBO algorithm have been presented in recent years, such as Event-Triggered Time-Varying Bayesian Optimization (ET-TVBO) by Brunzema et al. [3] which assumes the underlying function is static until a sudden change-point occurs which causes the function to change to a different function which then remains static until another change-point occurs. This ET-TVBO has been shown to outperform [3] the TVBO model in some real world applications which confirms that this is still an active area of research.

The difficulty in developing these models for real world applications arises from the unknown nature of the temporal dependence of the function as well as the requirement to minimize the number function evaluations. Minimizing the number of evaluations implies that we would like to develop models which work well on small datasets. This is achieved by having good priors on the underlying function; the assumptions we want to make on how function changes over time must be sufficiently strict. If we develop a model which is too flexible it will likely over-fit on the small dataset of function evaluations; so the temporal relation of our function should be informed by our prior of the underlying process. If we expect the function to change gradually over time the TVBO algorithm is applicable. However, if we expect the function to be mostly static which sudden change-points the ET-TVBO algorithm would be reasonable choice. This raises the question of which other types of temporal dependence might be relevant for real world applications.

In this thesis we derive improved regret bounds for the TVBO algorithm [2]. These new bounds provide better scaling for long timescales (many evaluations). Additionally, the new bounds build a theoretical basis for us to consider the influence of the rate of change of the function ε on our algorithm. We support these theoretical results by conducting a simulation study.

These algorithms can be applied to hyperparameter optimization of deep reinforcement learning algorithms. These are algorithms which are used to train an agent to perform certain actions within a (simulated) environment. As the agent explores the environment and encounters different challenges we expect the optimal hyperparameters to change over time. Suppose a robot is learning how to walk by controlling actuators. The robot receives a reward for standing upright and an additional reward which is proportional to its forward speed. Initially, the robot struggles to stand and falls over often. After it has mastered the task of standing without falling over it is able to start learning how to walk forward. This is quite a different task than learning to stand stably and requires different hyperparameters. Detecting and adapting to these changes can be handled by the algorithms discussed in this thesis [20]. We expect that more complex learning problems will require more flexible temporal models.

To this end, we introduce two new algorithms named Momentum Time Varying Gaussian Process Upper Confidence Bound (MTV-GP-UCB) and Transition Time Varying Gaussian Process Upper Confidence Bound (TTV-GP-UCB) which outperform existing algorithms in cases where the underlying function has a more complex temporal covariance structure. For example if the function transitions from one function h_1 to h_2 over some time period. Or if rather than assuming the changes in the function are completely random perturbations we expect changes to "trend" in a more predictable fashion. We present regret bounds for these new models and perform experiments on generated data to validate their use cases.

Theoretical Background

2
Gaussian Process Optimization for a Static Function

We will begin by introducing the case where the unknown function is static and does not change between evaluations. This is a widely studied scenario and it builds the basis for much of the work on time varying Bayesian optimization. Our goal is to find the global maximum of some function f over some domain $D \subseteq [0, r]^d$, more precisely we want to find $x \in D$ s.t. $f(x)$ is maximized:

$$x^* = \arg\max_{x \in D} f(x).$$

At each time step we sample the function at some point $x_t \in D$. This allows us to observe a noisy evaluation $y_t = f(x_t) + z_t$ with $z_t \sim \mathcal{N}(0, \sigma^2)$ and then use this information (as well as the observations of all previous samples) to inform our choice of the following point $x_{t+1} \in D$.

It is important to note that there are multiple interpretations to what is meant by "finding the maximum of f". In some settings we simply want to learn as much as possible about f and only maximize the observation of the final evaluation regardless of the observations along the way. This is known as Bayesian experimental design, see [4]. In our case we are concerned with also sampling "good" points of the function while we are exploring. This is formulated as minimizing the regret. The instantaneous regret is defined as:

$$r_t = \max_{x \in D} \left[f\left(x\right) \right] - f\left(x_t\right)$$

where x_t is the point we choose to sample and x^* is the point in which the function is maximized over the domain. Note that the regret will always be non-negative. We aim to design an algorithm which minimizes the cumulative regret $R_T = \sum_{t=1}^{T} r_t$. It should be noted that this is more complex than simply minimizing each individual r_t because the choice of x_t will affect our estimate of f in future timesteps. It can be worthwhile to accept a high r_t early on in order to ensure that future regrets $(r_{t+1}, r_{t+2}, \ldots)$ can be reduced. This is reflected in the cumulative regret R_T.

Depending on the nature of f we can employ different strategies in order to minimize R_T. If we assume that f is sampled from a Gaussian Process, that is $f \sim \mathcal{GP}(0, k)$, this can provide a versatile solution for non-linear functions. If we assume the function is sufficiently smooth and can be modelled using Gaussian Process regression (GPR) we can use this to inform our decisions. Throughout this thesis Gaussian Processes play two roles. Firstly, we assume that the process which is generating the function f is a Gaussian process. This is a prior on our function f. Secondly, we use GPR to compute a posterior which combines our prior on f with the knowledge we have gained about f from our samples $\{(x_1, y_1), ..., (x_t, y_t)\}$.

The functions which are sampled from the Gaussian Process are dictated by the mean and chosen kernel function. The kernel function describes how strongly two points are correlated, providing a type of pseudo-distance measure for the points in our domain. A common kernel is the square exponential kernel which is given by:

$$k_{\text{SE}}(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2l^2}\right) \tag{2.1}$$

where $\|\cdot\|$ is defined as the euclidean distance. For the square exponential kernel the value is determined by the squared euclidean distance and the "lengthscale" hyperparameter l. The lengthscale controls how the correlation between points changes with distance. Intuitively, a kernel with a lower lengthscale will lead to more erratic functions. The SE kernel is just one possible kernel which serves as an example for the reader. For the remainder of this thesis the notation $k(x, x')$ refers to any kernel which satisfies the assumptions of the given theorem.

The GPR model allows us to use past evaluations $\mathbf{x_t} = [x_1, ..., x_t]$ and corresponding evaluations $y_1, ..., y_t$ to predict the function mean $\mu_t(x)$ and variance $\sigma_t^2(x, x)$ using the matrix equations [28]:

$$\mu_t(x) = \mathbf{k}_t(x)^T(\mathbf{K}_t + \sigma^2\mathbf{I})^{-1}y_t \tag{2.2}$$

$$\sigma_t^2(x, x) = k(x, x) - \mathbf{k}_t(x)^T(\mathbf{K}_t + \sigma^2\mathbf{I})^{-1}\mathbf{k}_t(x) \tag{2.3}$$

where

$$\mathbf{k}_t(x) := [k(x_i, x)]_{i=1}^t \tag{2.4}$$

$$\mathbf{K}_t := [k(x, x')]_{x,x' \in \mathbf{x_t}}. \tag{2.5}$$

This GPR can now serve as a tool to inform the choice of x_t in a process often referred to as Bayesian optimization. One method of informing the choice is given by:

$$x_t = \arg\max_{x \in D} \left[\mu_{t-1}(x) + \beta^{1/2}\sigma_{t-1}(x)\right].$$

where $\mu_{t-1}(x)$ and $\sigma_{t-1}(x)$ is the predicted mean and variance provided by the GPR, this method is often called the Gaussian process upper confidence bound (GP-UCB) [6]. The parameter β in the UCB algorithm controls the exploration-exploitation trade-off (it is sometimes referred to as the trade-off parameter). Depending on the implementation and use case β can be a constant or depend on t, if it depends on t we will denote this by β_t. In 2012 the seminal work [28] was the first paper to present an algorithm which is guaranteed to achieve sub-linear regret in this setting. This means that using their algorithm we have that:

$$\lim_{T\to\infty} \frac{R_T}{T} = 0.$$

In their proof β depends on t and is hence referred to β_t. They presented the following theorem (Theorem. 2 [28]):

Theorem 2.1 Let $D \subseteq [0, r]^d$ be compact and convex with $d \in \mathbb{N}$ and $r > 0$. Assume that $f \sim \mathcal{GP}(0, k)$. And that the kernel $k(x, x')$ satisfies the following probabilistic bound on the derivatives of f for some $a, b > 0$:

$$\mathbb{P}\left\{\sup_{\boldsymbol{x}\in D}\left|\frac{\partial f}{\partial x^{(j)}}(x)\right| > L\right\} \leq ae^{-(L/b)^2}, \quad j = 1, \ldots, d \tag{2.6}$$

where $x^{(j)}$ denotes the partial derivative in the j^{th} dimension. Pick some $\delta \in (0, 1)$, and define:

$$\beta_t = 2\log\left(\frac{t^2 2\pi^2}{3\delta}\right) + 2d\log\left(t^2 dbr\sqrt{\log\left(\frac{4da}{\delta}\right)}\right).$$

At every timestep we observe some $y_t = f(x_t) + z_t$ with $z_t \sim \mathcal{N}(0, \sigma^2)$. If we fit a GPR to the observations and select x_t according to:

$$x_t = \arg\max_{x\in D} \mu_{t-1}(x) + \beta_t^{1/2}\sigma_{t-1}(x)$$

where $\mu_{t-1}(x)$ and $\sigma_{t-1}(x)$ are defined as in (2.2) and (2.3). Then we will achieve the following bounds on the cumulative regret R_T:

$$\mathbb{P}\left\{R_T \leq \sqrt{C_1 T\beta_T\gamma_T} + 2 \quad \forall T \geq 1\right\} \geq 1 - \delta$$

with $C_1 = \frac{8}{\log(1+\sigma^{-2})}$ and γ_T the maximum information gain which will be described in more detail at the end of this section.

Note that the assumption (2.6) ensures that f is smooth with high probability. Theorem 5 in [10] showed that this holds for any stationary kernel where $k(x, x') = k(x - x')$ is four times differentiable. For example, this will hold for the commonly used Matérn kernel with $\nu > 2$ or any square exponential kernel. It will not hold for the Ornstein-Uhlenbeck kernel.

This theorem is a powerful result as it shows that with mild assumptions on the process which generates the underlying function f we can achieve sub-linear regret bounds. If the cumulative regret is sub-linear it means that the average regret R_T/T will converge to 0 in the limit as $T \to \infty$. Algorithm 1 presents pseudo-code of how this theorem would be applied in practice.

Algorithm 1 GP-UCB algorithm

Require: Domain D, GP prior $(\mu_0 = 0, k)$ and noise variance σ^2

1: **for** $t = 1, 2, \ldots,$ T **do**
2: Choose $x_t = \arg\max_{x \in D} \left[\mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x)\right]$
3: Sample $y_t = f(x_t) + z_t$
4: Perform Bayesian update according to (2.2) and (2.3)
5: **end for**

Now we will discuss the maximum information gain γ_T. This is an information theoretic [5] property of the function generating process (in this case the Gaussian process) which describes how much can be "learned" about the underlying function with each sample. For a set of selected points A in the GPR setting the information gain is given by:

$$I(\mathbf{y}_A; f) = \frac{1}{2} \log |\mathbf{I} + \sigma^{-2} \mathbf{K}_A| \tag{2.7}$$

where $|\cdot|$ refers to the determinant of the matrix. In the proof of Theorem 2.1 the maximum information gain is used to bound the predictive variance $\sigma_{t-1}(x)$. As we learn more about the function f the predictive variance $\sigma_{t-1}(x)$ is non-increasing for every $x \in D$. In the proof of Theorem 2.1 the maximum information gain allows us to create a bound on $\sum_{t=1}^{T} \sigma_{t-1}(x_t)$. The detailed proof can be found in [28], a similar proof for a new regret bound is presented in chapter 4.

A visual representation of the maximum information gain is provided in Figure 2.1. The functions sampled from a Gaussian process with a lower lengthscale are easier to learn with a small number of samples (this is the low γ_T case). Alternatively, the function shown in red has a low lengthscale resulting in a higher γ_T and therefore likely a higher cumulative regret

if we try to optimize it. With the same number of samples there is more uncertainty for the red function with the higher maximum information gain.

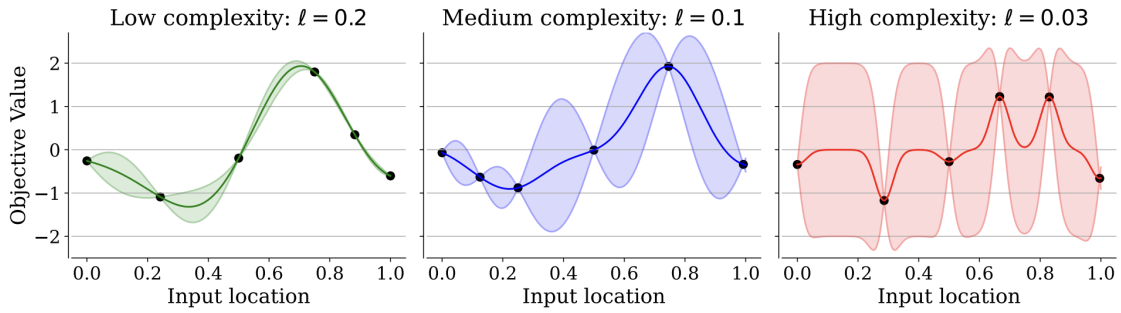


Figure 2.1: Impact of lengthscale on function complexity [14]

It is interesting to note that in the setting of Bayesian experimental design which is an alternative setting where we simply want to learn as much about the function as possible without considering regret we are actually trying to maximize this information gain. So for Bayesian experimental design we simply want to choose an A such that (2.7) is maximized. The optimal set of points to sample if the goal is to maximize the information gain can be approximated by the greedy algorithm of selecting the point with the highest uncertainty at every step:

$$x_t = \arg\max_{x \in D} \sigma_{t-1}(x). \tag{2.8}$$

So if we were simply concerned with exploring the function as much as possible equation (2.8) provides a close to optimal solution (as shown in [28]). However, this method is independent of the expected function mean $\mu_{t-1}(x)$ which highlights that it is not concerned with exploiting (sampling good function values along the way). Even when are concerned with minimizing the regret R_T we still observe in Theorem 2.1 that the maximum information gain plays a large role in the performance of our algorithm.

The cumulative regret scales with the maximum information gain term γ_T which forms a connection between Gaussian Process optimization and experimental design. A function with a higher maximum information gain will result in a larger cumulative regret when applying the GP-UCB algorithm. An interpretation of this is that we must sufficiently explore the function in order to guarantee a bounded regret. A function which has more maximum information gain will require more exploration which limits the exploitation (and therefore increases the regret).

3
Gaussian Process Optimization for a Time-Varying Function

So far we have assumed that the function f is fixed and doesn't not change over time. However, in some real world settings this might not be the case. In 2016 Bogunovic, Scarlett, and Cevher [2] presented a Time-Varying BO (TVBO) model which aims to optimize a function which is changing over time (in between evaluations). They made the assumption that the data was generated by the following underlying model:

$$\begin{aligned} f_1(x) &= g_1(x) \\ f_{t+1}(x) &= \sqrt{1-\varepsilon} f_t(x) + \sqrt{\varepsilon} g_{t+1}(x) \quad \forall t \geq 1, \end{aligned} \tag{3.1}$$

where $g_1, g_2, ...$ are functions which are independently sampled from a Gaussian Process $g_i \sim \mathcal{GP}(0, k)$. In this case $\varepsilon \in (0, 1]$ controls the rate at which the function changes between each evaluation. It is important to note that with this model we have that for all ε and all t our prior is $f_t \sim \mathcal{GP}(0, k)$. The terms $\sqrt{1-\varepsilon}$ and $\sqrt{\varepsilon}$ ensure that the variance of the Gaussian process does not diverge over time. Similarly to chapter 2 we again want to create an algorithm which can select the optimal point x_t to sample in order to minimize the cumulative regret. It is important to note that in this time varying setting the maximum of the function changes over time, so the optimal point x^* is now also time varying and denoted by $x_t^* = \arg\max_{x \in D} f_t(x)$. The instantaneous regret is then defined as:

$$r_t = \max_{x \in D} \left[f_t(x) \right] - f_t(x_t) = f_t(x_t^*) - f_t(x_t)$$

In this setting we can no longer obtain any sub-linear regret such as the static function setting of chapter 2. Intuitively; this is because the function changes significantly over time which causes the data collected to become stale. We cannot perfectly track the maximum of the function over time. The quality of an algorithm is now measured in how the regret scales with ε the rate of change of the function.

The time varying GP-UCB algorithm was presented in [2] in order to tackle this problem. It builds on previous methods and adjusts them in order to handle a function which changes between evaluations. The previously defined equation (2.2) and equation (2.3) for the

Gaussian Process regression posterior mean and variance are used. However, the kernel function is adapted to handle the timestamp data:

$$\tilde{\mathbf{K}}_t = \mathbf{K}_t \circ \mathbf{D}_t \tag{3.2}$$

$$\tilde{\mathbf{k}}_t(x) = \mathbf{k}_t(x) \circ \mathbf{d}_t, \tag{3.3}$$

with $\circ$ as the Hadamard product (which is the element-wise product) and $\mathbf{K}_t$ and $\mathbf{k}_t(x)$ representing the original spatial kernels for the Gaussian Process regression. The time varying kernel is defined as; $\mathbf{D}_t = [(1-\varepsilon)^{|i-j|/2}]_{i,j=1}^{t}$ and $\mathbf{d}_t = [(1-\varepsilon)^{|t+1-i|/2}]_{i=1}^{t}$. The GPR model for the time varying case now uses $\tilde{\mathbf{K}}_t$ and $\tilde{\mathbf{k}}_t(x)$:

$$\mu_t(x) = \tilde{\mathbf{k}}_t(x)^T(\tilde{\mathbf{K}}_t + \sigma^2\mathbf{I})^{-1}y_t \tag{3.4}$$

$$\sigma_t^2(x) = \tilde{k}(x,x) - \tilde{\mathbf{k}}_t(x)^T(\tilde{\mathbf{K}}_t + \sigma^2\mathbf{I})^{-1}\tilde{\mathbf{k}}_t(x) \tag{3.5}$$

By using the same UCB algorithm with this adjusted GPR and a modified equation for β_t, regret bounds can be derived for this time varying case. In a derivation which builds on the work described in chapter 2 the regret bounds for this algorithm are found to scale according to (Theorem. 4.3 [2]):

Theorem 3.1 Let $D \subseteq [0,r]^d$ be compact and convex with $d \in \mathbb{N}$ and $r > 0$. Suppose a time varying function is generated according to:

$$f_1(x) = g_1(x)$$
$$f_{t+1}(x) = \sqrt{1-\varepsilon}f_t(x) + \sqrt{\varepsilon}g_{t+1}(x) \quad \forall t \geq 1,$$

where $g_i \sim \mathcal{GP}(0,k)$ and $\varepsilon \in (0,1]$. Assume that the spatial kernel k generates functions such that there exists some $a, b > 0$ for which it holds that:

$$\mathbb{P}\left\{\sup_{\boldsymbol{x}\in D}\left|\frac{\partial f}{\partial x^{(j)}}\right| > L\right\} \leq ae^{-(L/b)^2}, \quad j = 1,\ldots,d.$$

At each time step we select one point in the domain x_t to sample the function and observe a noisy evaluation $y_t = f(x_t) + z_t$ with $z_t \sim \mathcal{N}(0,\sigma^2)$. Now, choose some $\delta \in (0,1)$ and define:

$$\beta_t = 2\log\frac{\pi^2 t^2}{2\delta} + 2d\log\left(rdbt^2\sqrt{\log\frac{da\pi^2 t^2}{2\delta}}\right). \tag{3.6}$$

Now select the point to sample at each timestep according to the UCB rule:

$$x_t = \arg\max_{x\in D}\left[\mu_{t-1}(x) + \beta_t^{1/2}\sigma_{t-1}(x)\right].$$

with $\mu_{t-1}(x)$ and $\sigma_{t-1}(x)$ from (3.4) and (3.5). Note that this is the Gaussian process regression which uses the combined spatial and temporal kernel. Then we will achieve cumulative regret R_T:

$$\mathbb{P}\left\{R_T \leq \sqrt{C_1 T\beta_T\tilde{\gamma}_T} + 2\right\} \geq 1-\delta \tag{3.7}$$

where $C_1 = 8/\log(1+\sigma^{-2})$ and $\tilde{\gamma}_T$ is the maximum information gain for the time varying function f_t. Alternatively:

$$\mathbb{P}\left\{R_T \leq \sqrt{C_1 T\beta_T\left(\frac{T}{\tilde{N}}+1\right)\left(\gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\varepsilon\right)} + 2\right\} \geq 1-\delta \tag{3.8}$$

where the term $\gamma_{\tilde{N}}$ is the maximum information gain for the static function without changes between samples (this would be the maximum information gain if we fixed t and sampled f_t with $\tilde{N}$ samples). The variable $\tilde{N} \in [1, ..., T]$ is simply a tool in the analysis and can be freely chosen to achieve the tightest bound.

The result provides the first regret bound for this time varying setting. However, the current regret bound still leaves much to be desired. Firstly, since $\beta_T = \mathcal{O}(\log(T))$ we see from equation (3.8) that the bound on the cumulative regret becomes $\mathcal{O}(T\sqrt{\log(T)})$. But then the average regret $\frac{R_T}{T}$ still grows according to $\sqrt{\log(T)}$ which implies that our algorithm will start to perform worse over time (even though we are gaining information about the underlying function).

Secondly, in equation (3.6) we observe that β_t is chosen independently of ε. This is odd as we would expect the rate of change of the function to have an influence on the optimal choice in the exploration-exploitation trade-off. Exploration is less valuable if the function is changing quickly because it means that there is little time to exploit any gained information.

Finally, β_t depends on the choice of δ. The user must specify the probability with which they want the bound to hold $(1-\delta)$ before they start the algorithm. In chapter 4 we will aim to remedy these issues.

When applied in practice the algorithm is used as follows:

Algorithm 2 TV-GP-UCB algorithm

Require: Domain D, GP prior $(\mu_0 = 0, k)$, parameter ε and noise variance σ^2

1: **for** $t = 1, 2, \ldots,$ T **do**
2: Choose $x_t = \arg\max_{x \in D} \left[\mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x)\right]$
3: Sample $y_t = f(x_t) + z_t$
4: Perform Bayesian update according to (3.4) and (3.5)
5: **end for**

The maximum information gain for the time varying function $\tilde{\gamma}_T$ is a more complex term than for the static function case. This is because at every timestep the function is perturbed by a random function. This introduces new uncertainty over time which increases the predictive variance $\sigma_{t-1}(x)$ of previously sampled points. In [2] (Theorem. 4.3) it is shown that the maximum information gain for a time varying function can be bounded. Due to a small error in their analysis the corrected results which are presented here are slightly different to [2]. We also present the (corrected) proof.

Theorem 3.2

Consider the setting of theorem 3.1, then the maximum information gain for the time varying function can be bounded as follows:

$$\tilde{\gamma}_T \leq \left(\frac{T}{\tilde{N}} + 1\right)\left(\gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\varepsilon\right) \tag{3.9}$$

where $\gamma_{\tilde{N}}$ is the maximum information gain for a static function $f \sim \mathcal{GP}(0, k)$ which is sampled $\tilde{N}$ times.

Proof. The information gain for a time varying function which is sampled from a Gaussian process is defined as:

$$\tilde{I}\left(\mathbf{f}_T; \mathbf{y}_T\right) = \frac{1}{2}\log\left|\mathbf{I}_T + \sigma^{-2}\tilde{\mathbf{K}}_T\right|$$
$$\tilde{\gamma}_T := \max_{x_1,\ldots,x_T} \tilde{I}\left(\mathbf{f}_T; \mathbf{y}_T\right)$$

where $\tilde{\gamma}_T$ is the maximum information gain. We use the notation $\mathbf{x}_T = (x_1, \ldots, x_T)$ for the sampled points, $\mathbf{f}_T = (f_1(x_1), \ldots, f_T(x_T))$ the function values at those locations, and

$\mathbf{y}_T = (y_1, \ldots, y_T)$ the noisy observations. We will bound the information gain for the time varying function by splitting the steps $\{1, \ldots, T\}$ into $\frac{T}{\tilde{N}}$ [1] blocks on length $\tilde{N}$. This is useful because within the small time interval of each block the function f_t is close to static (and we will add a small overhead to account for the fact that the function is still changing within each block). From the chain rule for mutual information and the fact that the noise in the observations is independent ([5], Lemma 7.9.2):

$$\tilde{I}(\mathbf{f}_T; \mathbf{y}_T) \leq \sum_{i=1}^{T/\tilde{N}} \tilde{I}(\mathbf{f}_{\tilde{N}}^{(i)}; \mathbf{y}_{\tilde{N}}^{(i)})$$

where $\mathbf{y}_{\tilde{N}}^{(i)} = (\mathbf{y}_{\tilde{N}(i-1)+1}, \ldots, \mathbf{y}_{\tilde{N}i})$ refers to the i-th block of $\mathbf{y}_T$ and $\mathbf{f}_{\tilde{N}}^{(i)}$ is defined in the same manner. This leads to the bound:

$$\tilde{\gamma}_T \leq \frac{T}{\tilde{N}} \tilde{\gamma}_{\tilde{N}}$$

where $\tilde{\gamma}_{\tilde{N}}$ is the maximum information gain for the time varying function within the block of size $\tilde{N}$. We must now bound this term (and account for the fact that f_t is not a static function within the block). For this we return to the definition of the information gain (this time for the case of one block of size $\tilde{N}$):

$$\tilde{I}(\mathbf{f}_{\tilde{N}}^{(i)}; \mathbf{y}_{\tilde{N}}^{(i)}) = \frac{1}{2} \log |\mathbf{I}_{\tilde{N}} + \sigma^{-2} \tilde{\mathbf{K}}_{\tilde{N}}| \leq \tilde{\gamma}_{\tilde{N}}. \tag{3.10}$$

We will focus on the relevant covariance matrix $\tilde{\mathbf{K}}_{\tilde{N}}$. Recall from equation (3.2) that this matrix can be written as an element-wise product of the spatial and temporal covariance matrix:

$$\tilde{\mathbf{K}}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} \circ \mathbf{D}_{\tilde{N}}.$$

We introduce a new matrix:

$$\mathbf{A}_{\tilde{N}} := \mathbf{K}_{\tilde{N}} \circ \mathbf{D}_{\tilde{N}} - \mathbf{K}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} \circ (\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})$$

where $\mathbf{D}_{\tilde{N}}$ is the $\tilde{N} \times \tilde{N}$ matrix of ones. This now allows us to write:

$$\tilde{\mathbf{K}}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}}.$$

[1] We assume $\frac{T}{\tilde{N}}$ is an integer for now and we will discuss the impact of this at the end of the proof

Recall that we defined $\mathbf{D}_t = [(1-\varepsilon)^{|i-j|/2}]_{i,j=1}^t$ which now implies that the (i,j)-th entry of $(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})$ has absolute value:

$$1 - (1-\varepsilon)^{\frac{|i-j|}{2}} \leq \varepsilon|i-j| \quad \forall \varepsilon \in [0,1]. \tag{3.11}$$

Consider the inequality $1-(1-\varepsilon)^{\frac{|i-j|}{2}} - \varepsilon|i-j| \leq 0$. For $|i-j| = 0$ we have equality. The inequality holds for $|i-j| \geq 2$ because the function is concave, passes through the origin and is non-positive (equal to $-\varepsilon$) for $|i-j| = 2$. For $|i-j| = 1$ consider the inequality $1-(1-\varepsilon)^{\frac{1}{2}} - \varepsilon \leq 0$. Observe that we have equality for $\varepsilon \in \{0,1\}$ and $1-(1-\varepsilon)^{\frac{1}{2}} - \varepsilon$ is convex w.r.t. ε so the inequality also holds for $\varepsilon \in (0,1)$.

Since we assumed $k(x,x) \leq 1$ we know that the entries of $\mathbf{K}_{\tilde{N}}$ are all ≤ 1. Then we can obtain the following bound on the Frobenius norm of the matrix $\mathbf{A}_{\tilde{N}}$:

$$\begin{aligned}
\|\mathbf{A}_{\tilde{N}}\|_F^2 = \|\mathbf{K}_{\tilde{N}} \circ (\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})\|_F^2 &\leq \|(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})\|_F^2 && (3.12)\\
&\leq \sum_{i,j} (i-j)^2\varepsilon^2 && (3.13)\\
&= \frac{1}{6}\tilde{N}^2(\tilde{N}-1)^2\varepsilon^2 && (3.14)\\
&\leq \tilde{N}^4\varepsilon^2 && (3.15)
\end{aligned}$$

where (3.12) follows from the fact that every entry of $\mathbf{K}_{\tilde{N}}$ is ≤ 1. Equation (3.13) results from equation (3.11) which provides an upper bound on the entries of $(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})$. Equation (3.14) is a result of computing the double summation. The final inequality is a simplification for readability; $(\tilde{N}-1)^2 \leq \tilde{N}^2$. We will now use this inequality to bound $\tilde{\gamma}_{\tilde{N}}$. We introduce Mirsky's theorem, which allows us to make a statement about the eigenvalues of $\tilde{\mathbf{K}}_{\tilde{N}}$.

Lemma 3.1 (Mirsky's theorem [[13], Cor. 7.4.9.3]) For any matrices $\mathbf{U}_{\tilde{N}}$ and $\mathbf{V}_{\tilde{N}}$, and any unitarily invariant norm $|||\cdot|||$, we have

$$|||\mathrm{diag}(\lambda_1(\mathbf{U}_{\tilde{N}}), ..., \lambda_{\tilde{N}}(\mathbf{U}_{\tilde{N}})) - \mathrm{diag}(\lambda_1(\mathbf{V}_{\tilde{N}}), ..., \lambda_{\tilde{N}}(\mathbf{V}_{\tilde{N}}))||| \leq |||\mathbf{U}_{\tilde{N}} - \mathbf{V}_{\tilde{N}}|||$$

where λ_i is the i-th largest eigenvalue.

For readability, we now define:

$$\Delta_i := \lambda_i(\tilde{\mathbf{K}}_{\tilde{N}}) - \lambda_i(\mathbf{K}_{\tilde{N}}). \tag{3.16}$$

Applying lemma 3.1 with $\mathbf{U}_{\tilde{N}} = \tilde{\mathbf{K}}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}}$ and $\mathbf{V}_{\tilde{N}} = \mathbf{K}_{\tilde{N}}$. We choose $|||\cdot||| = \|\cdot\|_F^2$ which gives us the following bound on $\{\Delta_i\}_{i=1}^{\tilde{N}}$:

$$\begin{aligned}
\|\text{diag}(\lambda_1(\mathbf{U}_{\tilde{N}}), ...) - \text{diag}(\lambda_1(\mathbf{V}_{\tilde{N}}), ...)\|_F^2 &= \sum_{i=1}^{\tilde{N}} \left(\lambda_i(\tilde{\mathbf{K}}_{\tilde{N}}) - \lambda_i(\mathbf{K}_{\tilde{N}})\right)^2 && (3.17)\\
&= \sum_{i=1}^{\tilde{N}} \Delta_i^2 && (3.18)\\
&\leq \|\mathbf{U}_{\tilde{N}} - \mathbf{V}_{\tilde{N}}\|_F^2 = \|\mathbf{A}_{\tilde{N}}\|_F^2 && (3.19)\\
&\leq \tilde{N}^4 \varepsilon^2. && (3.20)
\end{aligned}$$

Here equation (3.17) follows from the definition of the Frobenius norm and (3.18) from our definition of Δ_i. The first inequality (3.19) results from applying lemma 3.1. Finally, equation (3.20) results from our previous bound in equation (3.15). We have shown that:

$$\sum_{i=1}^{\tilde{N}} \Delta_i^2 \leq \tilde{N}^4 \varepsilon^2. \tag{3.21}$$

Observe that we can rewrite equation (3.16) as $\lambda_i(\tilde{\mathbf{K}}_{\tilde{N}}) = \lambda_i(\mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}}) = \lambda_i(\mathbf{K}_{\tilde{N}}) + \Delta_i$. Combining this with equation (3.21) we now have the following bound on the eigenvalues $\lambda_i(\tilde{\mathbf{K}}_{\tilde{N}})$:

$$\sum_{i=1}^{\tilde{N}} \lambda_i(\tilde{\mathbf{K}}_{\tilde{N}}) = \sum_{i=1}^{\tilde{N}} \lambda_i(\mathbf{K}_{\tilde{N}}) + \Delta_i \leq \tilde{N}^4 \varepsilon^2 + \sum_{i=1}^{\tilde{N}} \lambda_i(\mathbf{K}_{\tilde{N}}).$$

We now have a bound on the sum of $\lambda_i(\tilde{\mathbf{K}}_{\tilde{N}})$. We will now return to bounding the maximum information gain for the time-varying function:

$$\tilde{\gamma}_{\tilde{N}} = \max_{x_1,\ldots,x_{\tilde{N}}} \frac{1}{2} \log |\mathbf{I}_{\tilde{N}} + \sigma^{-2}\tilde{\mathbf{K}}_{\tilde{N}}| \tag{3.22}$$

$$= \max_{x_1,\ldots,x_{\tilde{N}}} \frac{1}{2} \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\lambda_i(\mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}})\right) \tag{3.23}$$

$$= \max_{x_1,\ldots,x_{\tilde{N}}} \frac{1}{2} \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\left[\lambda_i(\mathbf{K}_{\tilde{N}}) + \Delta_i\right]\right) \tag{3.24}$$

$$\leq \max_{x_1,\ldots,x_{\tilde{N}}} \frac{1}{2} \left[\sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\lambda_i(\mathbf{K}_{\tilde{N}})\right) + \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\Delta_i\right)\right] \tag{3.25}$$

$$= \gamma_{\tilde{N}} + \max_{x_1,\ldots,x_{\tilde{N}}} \frac{1}{2} \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\Delta_i\right) \tag{3.26}$$

$$\leq \gamma_{\tilde{N}} + \max_{x_1,\ldots,x_{\tilde{N}}} \frac{1}{2}\sigma^{-2} \sum_{i=1}^{\tilde{N}} \Delta_i \tag{3.27}$$

$$\leq \gamma_{\tilde{N}} + \frac{1}{2}\sigma^{-2}\tilde{N}^{5/2}\varepsilon \leq \gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\varepsilon. \tag{3.28}$$

Here, equation (3.22) follows from the definition of the maximum information gain (3.10). Equation (3.23) results from the fact that the determinant of a matrix is equal to the product of its eigenvalues. Equation (3.24) follows from our definition (3.16). The first inequality (3.25) results from the inequality $\log(1 + a + b) \leq \log(1 + a) + \log(1 + b)$ which holds for non-negative a, b. Equation (3.26) results from our original definition of the maximum information gain. The second inequality (3.27) follows from $\log(1 + a) \leq a$ for non-negative a. Finally, equation (3.28) results from observing that the sum of Δ_i is maximized (while respecting (3.21)) by choosing $\Delta_i = \tilde{N}^{3/2}\varepsilon$ for all i. We disregard the factor $\frac{1}{2}$ in order to simplify notation.

We have shown that for integer $\frac{T}{\tilde{N}}$ we have the bound:

$$\tilde{\gamma}_T \leq \frac{T}{\tilde{N}}\tilde{\gamma}_{\tilde{N}} \leq \frac{T}{\tilde{N}}\left(\gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\varepsilon\right).$$

So far we have assumed that $\frac{T}{\tilde{N}}$ is an integer. Observe that $\tilde{\gamma}_T$ is an increasing function of T which means that we can generalize the bound for non-integer $\frac{T}{\tilde{N}}$ by adding 1 which yields the final bound:

$$\tilde{\gamma}_T \leq \left(\frac{T}{\tilde{N}} + 1\right)\left(\gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\varepsilon\right).$$

□

In equation (3.9) $\tilde{N}$ is a variable which is used to split the data into smaller blocks (where the function does not vary too much). The block size $\tilde{N} \in [1, ..., T]$ can be chosen such that the r.h.s. of equation (3.9) is minimized (such that the tightest bound is achieved).

Improvements Time Varying Gaussian Process Optimization

4
Improved Regret Bounds

The regret bounds derived by [2] for the time varying case were presented in chapter 3. They have some undesirable properties which we will aim to remedy in this chapter. We will present new regret bounds for time-varying GP optimization and show that they can be achieved by choosing a value for β_t which remains constant over time. On a high level this bound is achieved by combining a probabilistic bound on the regret for the specific algorithm with an algorithm agnostic bound on the tail probability. These are used to bound the instantaneous regret $\mathbb{E}[r_t]$. Finally we show that for any $\varepsilon \in (0, 1]$ the average regret $\frac{R_T}{T}$ will converge in probability to the expected average regret $\frac{\mathbb{E}[R_T]}{T}$. We provide two bounds on the cumulative regret. Firstly, we show that $\mathbb{E}[R_T] = \mathcal{O}(T)$. We also provide a bound of the from: $R_T \leq h(T)$ with probability $(1 - \omega)$ which can be compared to the bounded presented in [2]. This will be explained in more detail after introducing the theorem.

Theorem 4.1 Consider a compact and convex set $D \subseteq [0, r]^d$ with $d \in \mathbb{N}$ and $r > 0$. Suppose a time varying function f_t is generated according to:

$$\begin{aligned} f_1(x) &= g_1(x) \\ f_{t+1}(x) &= \sqrt{1-\varepsilon} f_t(x) + \sqrt{\varepsilon} g_{t+1}(x) \quad \forall t \geq 1, \end{aligned} \tag{4.1}$$

where $g_i \sim \mathcal{GP}(0, k)$ and $\varepsilon \in (0, 1]$. Assume that for functions sampled from $\mathcal{GP}(0, k)$ there exists some $a, b > 0$ for which it holds that:

$$\mathbb{P}\left\{ \sup_{\boldsymbol{x} \in D} \left| \frac{\partial f}{\partial x^{(j)}} \right| > L \right\} \leq a e^{-(L/b)^2}, \quad \forall L \quad j = 1, \ldots, d. \tag{4.2}$$

At each time step we select one point in the domain x_t to sample the function and observe a noisy evaluation $y_t = f(x_t) + z_t$ with $z_t \sim \mathcal{N}(0, \sigma^2)$. Choose some $\delta \in (0, 1)$ and $\tau \in \mathbb{N}^+$, then if we select the point to sample at each timestep according to the UCB rule:

$$\beta = 2(\log(\frac{3}{2\delta}) + d\log(\tau)) \tag{4.3}$$

$$x_t = \arg\max_{x \in D} \left[\mu_{t-1}(x) + \beta^{1/2}\sigma_{t-1}(x)\right].$$

we will obtain cumulative regret R_T with the following upper bounds on the expectation:

$$\mathbb{E}[R_T] \leq (1-\delta)\left[\sqrt{C_1 T\beta\tilde{\gamma}_T} + TC_2\right] + T\delta C_3. \tag{4.4}$$

with:

$$C_1 = \frac{8}{\log(1+\sigma^{-2})}$$
$$C_2 = \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau}$$
$$C_3 = rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}}\right).$$

The assumption described by equation (4.2) implies that steep gradients are sufficiently unlikely. That is, for a given kernel k and domain D there exists some $a, b > 0$ such that the statement holds for all L. This assumption holds for many of the commonly used kernels in Bayesian optimization such as the square exponential (RBF) and Matérn kernel with $\nu > 2$. According to [10] Theorem 5, a bound of this form exists for any stationary kernels ($k(x, x') = k(x - x')$) which are four times differentiable.

The parameters δ and τ are mainly tools for the analysis. In practice, these parameters only affect the algorithm through their influence on β according to equation (4.3). We will briefly describe the role that δ and τ play in the analysis (more detail will be found in the following sections). The proof of the regret bound combines a bound on the specific (UCB) algorithm with a algorithm agnostic bound. The algorithm specific bound holds with probability $(1 - \delta)$ and the influence of the remaining probability δ is bounded using the algorithm agnostic bound. Hence, δ can be seen as a term which controls the amount to which each bound is used. If δ is close to 0 then most of the probability will be bounded by the algorithm specific bound (but this bound will be bigger) and vice versa. Now, the parameter τ is introduced because we are optimizing the function over a continuous domain. In the analysis it is difficult to compare our choice x_t to infinitely many alternatives $x \in D$.

This is handled by using a discretization of D where we approximate it by a finite set D^*. We add an additional overhead term to bound the effect (additional regret) due to using this discretization. The parameter number of points in the discretization is chosen as τ^d (so the parameter τ determines the number of points in our discretization). Choosing a larger τ will decrease the required overhead term as we using a better approximation of the continuous domain D. However, a larger τ will increase β which in turn increases the first term in equation (4.4). In order to achieve the best performing algorithm we must choose δ and τ to minimize the bound.

In the following sections we will introduce lemmas which we will use to prove this theorem.

4.1. Algorithm Specific Regret Bound

We will provide a probabilistic bound on the instantaneous regret r_t for the UCB algorithm (this proof builds on the work in [28] and [2]). First we will define two common bounds which we will utilize throughout the rest of this chapter:

Lemma 4.1 (Gaussian tail bound) Let $X \sim \mathcal{N}(\mu, \sigma^2)$. We claim that:

$$\mathbb{P}(X - \mu \geq t) \leq \frac{1}{2} e^{-\frac{t^2}{2\sigma^2}}.$$

Note that due to symmetry of the Gaussian distribution around the mean this also implies:

$$\mathbb{P}(\mu - X \geq t) \leq \frac{1}{2} e^{-\frac{t^2}{2\sigma^2}}.$$

Proof. Define $Z = \frac{X-\mu}{\sigma}$ and $z = \frac{t}{\sigma} \geq 0$, then define the function:

$$d(z) = \mathbb{P}(Z \geq z) - \frac{1}{2} e^{-\frac{z^2}{2}} \tag{4.5}$$

We claim that this function is ≤ 0 which represents the same claim in the original lemma. Note that $Z \sim \mathcal{N}(0, 1)$. We can take the derivative of $d(z)$ (here we use that $\mathbb{P}(Z \geq z) = 1 - F_Z(z)$, and the derivative of the cumulative distribution function CDF is the probability density function PDF):

$$d'(z) = (z - \sqrt{2/\pi}) \frac{1}{2} e^{-\frac{z^2}{2}}.$$

So $d(z)$ is decreasing on $[0, \sqrt{2/\pi}]$ and increasing on $[\sqrt{2/\pi}, \infty)$. We also have that $d(0) = 0$ and $\lim_{z\to\infty}(d(z)) = 0$, but this implies that the claimed inequality in (4.5) holds which completes the proof. □

Lemma 4.2 (Union bound) For a countable set of events $A_1, A_2, A_3, \ldots$ we have that the probability that at least one of the events happens is no greater than the sum of the probabilities of the individual events:

$$\mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i)$$

$$1 - \mathbb{P}\left(\bigcup_{i=1}^{\infty} A_i\right) \geq 1 - \sum_{i=1}^{\infty} \mathbb{P}(A_i).$$

This is a common result known as Boole's inequality.

Lemma 4.3 (Algorithm specific regret bound) Consider the setting of Theorem 4.1. Choose $\delta \in (0, 1)$ and $\tau \in \mathbb{N}^+$. If we use $\beta = 2(\log(\frac{3}{2\delta}) + d\log(\tau))$ and select the point to sample according to the UCB rule $x_t = \arg\max_{x\in D} \left[\mu_{t-1}(x) + \beta^{1/2}\sigma_{t-1}(x)\right]$ then we will achieve the following bound on the instantaneous regret:

$$r_t = \max_{x\in D} \left[f_t(x) - f_t(x_t)\right] \leq 2\beta^{1/2}\sigma_{t-1}(x_t) + \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau}$$

with probability $(1 - \delta)$.

Proof. Given τ choose a set of points $D^* \subset D \subseteq [0, r]^d$ of size $|D^*| = \tau^d$ and choose the points such that:

$$\|x - [x]\|_1 \leq \frac{rd}{\tau}, \quad \forall x \in D \tag{4.6}$$

where $[x]$ is the ℓ^1-closest point in D^* to x. We will refer to D^* as the discretization of D. An example of a sufficient discretization is a uniformly spaced grid. We need a discretization where any chosen point $x \in D$ has a closest point in D^* that is at most rd/τ away when

distance is measured in the ℓ^1-norm. If a uniformly spaced grid is used as the discretization then τ can be interpreted as the number points used to discretize each dimension. For example consider the case of $D = [0, 5]^2$ and $\tau = 6$, this would lead to a 6x6 grid in two dimensions. This is shown in Figure 4.1, you can observe that for any point $x \in D$ there exists a point in D^* which closer than $\frac{rd}{\tau} = \frac{10}{6}$ w.r.t. the ℓ^1-norm.

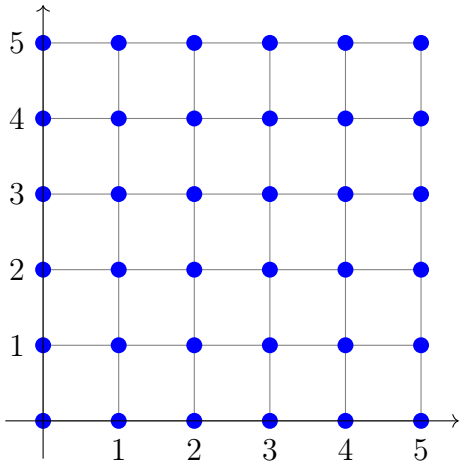


Figure 4.1: Grid discretization example

Given some $\delta \in (0, 1)$, fix $t \geq 1$. We will consider three high probability events which should each occur with probability at least $1 - \frac{\delta}{3}$:

1.) We claim that if we choose $\beta \geq 2\log(\frac{3}{2\delta})$ and fix some $x_t \in D$ the following holds with probability $1 - \frac{\delta}{3}$:

$$\mu_{t-1}(x_t) - f_t(x_t) \leq \beta^{\frac{1}{2}}\sigma_{t-1}(x_t). \tag{4.7}$$

Since our function is sampled from a Gaussian process. We can condition on the previously sampled points $(x_1, \ldots, x_{t-1})$ and outputs $(y_1, \ldots, y_{t-1})$. Then for any $x_t \in D$ we have that $f_t(x_t) \sim \mathcal{N}(\mu_{t-1}(x_t), \sigma^2_{t-1}(x_t))$. Where $\mu_{t-1}(x)$ and $\sigma_{t-1}(x)$ from (3.4) and (3.5). From lemma 4.1 we have that:

$$\mathbb{P}(\mu - f_t(x_t) \leq s) \leq \frac{1}{2}e^{-(s^2/2\sigma^2_{t-1}(x_t))}.$$

Now, if we choose $s = \beta^{\frac{1}{2}}\sigma_{t-1}(x_t)$ we directly see that the statement in equation (4.7) occurs with probability at least $1 - \frac{\delta}{3}$.

2.) We also want an upper bound on function for every point in the discretization. This is a bound on every point $x \in D^*$ rather than one specific point such as in 1.). If we choose $\beta \geq 2\log(\frac{3|D^*|}{2\delta}) = 2(\log(\frac{3}{2\delta}) + d\log(\tau))$ we can show that:

$$f_t(x) \leq \mu_{t-1}(x) + \beta^{\frac{1}{2}}\sigma_{t-1}(x) \quad \forall x \in D^*. \tag{4.8}$$

holds with probability at least $1 - \frac{\delta}{3}$. Similarly to 1.) we can use lemma 4.1 for some fixed $x \in D^*$. In this case we find that the inequality holds with probability $1 - \frac{\delta}{3|D^*|}$ for that specific x. Now in order to obtain the probability that it holds $\forall x \in D^*$ we apply De Morgan's law with lemma 4.2. Fix $x_i \in D^*$ and denote the event that (4.8) does not hold as E_i. Lemma 4.2 states that the probability that at least one event happens is at most the the sum of the probabilities of the individual events:

$$1 - \mathbb{P}\left(\bigcup_{i=1}^{|D^*|} E_i\right) \geq 1 - \sum_{i=1}^{|D^*|} \mathbb{P}(E_i) \geq 1 - \sum_{i=1}^{|D^*|} \frac{\delta}{3|D^*|} = 1 - \frac{\delta}{3}.$$

We see that the probability that every event E_i does not happen; that is, the probability that the bound holds for every $x \in D^*$, is at least $1 - \frac{\delta}{3}$

3.) Finally we also claim that setting $L = b\sqrt{\log(\frac{3ad}{\delta})}$ we obtain:

$$|f_t(x) - f_t(x')| \leq L \|x - x'\|_1 \quad \forall x, x' \in D \tag{4.9}$$

with probability at least $1 - \frac{\delta}{3}$.

By equation (4.2), it follows that for each $j \in \{1, ..., d\}$ the partial derivative with respect to $x^{(j)}$ is at most L with probability $1 - ae^{-(L/b)^2}$. Now taking the union bound over every coordinate $x^{(j)}$ of x using Boole's inequality as in 2.), we find that the partial derivative in each direction is at most L with probability $1 - ade^{-(L/b)^2}$. Substituting our choice of L yields $1 - ade^{-(L/b)^2} = 1 - \frac{3}{\delta}$. Finally, due to the fundamental theorem of calculus, a bound on the partial derivative of a continuous function can be used to bound the change of that function between two points:

$$f(b) - f(a) = \int_a^b f'(x)dx \leq L\int_a^b dx = (b-a)L$$

where the inequality holds if $f'(x) \leq L$. This implies:

$$\sup_{\boldsymbol{x}\in D,\, j\in\{1,\ldots,d\}} \left|\frac{\partial f}{\partial x^{(j)}}\right| \leq L \Rightarrow |f_t(x) - f_t(x')| \leq L \|x - x'\|_1 \quad \forall x, x' \in D.$$

We have shown that the three events 1.), 2.) and 3.) all occur with probability at least $1 - \frac{\delta}{3}$. Now we again apply lemma 4.2 over the complements of all three events which yields that the probability of them all occurring is at least $1 - \delta$. For the remainder of the proof we assume $\beta = 2(\log(\frac{3}{2\delta}) + d\log(\tau))$ and $L = b\sqrt{\log(\frac{3ad}{\delta})}$.

We start by making a statement about the difference between $f_t(x)$ and $f_t([x])$, where, as introduced earlier, $[x] \in D^*$ is the point closest to x (w.r.t. the ℓ^1 norm). For any $x \in D$ we have (with probability at least $1 - \frac{\delta}{3}$):

$$|f_t(x) - f_t([x])| \leq L \|x - [x]\|_1 \leq \frac{Lrd}{\tau} = \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau} = C_2. \tag{4.10}$$

The first inequality holds due to equation (4.9) and the second inequality follows from equation (4.6). By 3.) this inequality holds with probability at least $1 - \frac{\delta}{3}$. This can be interpreted as an extra overhead term we must add to our regret bound later in order to account for the fact that we are using a finite discretization of a continuous domain. It can be observed that this term decreases as we increase τ since the discretization becomes a better approximation of our true continuous domain as we increase the number of points. For readability we refer to this term as C_2.

Take x_t^* as the maximizing point over the domain D, $x_t^* = \arg\max_{x\in D} f_t(x)$. Then with probability at least $1 - \frac{2\delta}{3}$ we have that:

$$f_t(x_t^*) \leq f_t([x_t^*]) + C_2 \leq \mu_{t-1}([x_t^*]) + \beta^{\frac{1}{2}}\sigma_{t-1}([x_t^*]) + C_2 \tag{4.11}$$

where the first inequality holds due to equation (4.10) and the second inequality follows from equation (4.8) because $[x_t^*] \in D^*$. This holds with probability at least $1 - \frac{2\delta}{3}$ because it requires both 2.) and 3.) to hold.

Now we will bound the instantaneous regret r_t (with probability at least $1 - \delta$):

$$\begin{aligned} r_t &= f_t(x_t^*) - f_t(x_t) \\ &\leq \mu_{t-1}([x_t^*]) + \beta^{1/2}\sigma_{t-1}([x_t^*]) + C_2 - f_t(x_t) && (4.12)\\ &\leq \mu_{t-1}(x_t) + \beta^{1/2}\sigma_{t-1}(x_t) + C_2 - f_t(x_t) && (4.13)\\ &\leq 2\beta^{1/2}\sigma_{t-1}(x_t) + C_2. && (4.14) \end{aligned}$$

Here (4.12) follows directly from (4.11). (4.13) follows from the algorithm since we maximize the UCB so by definition we have $\mu_{t-1}([x_t^*]) + \beta^{1/2}\sigma_{t-1}([x_t^*]) \leq \mu_{t-1}(x_t) + \beta^{1/2}\sigma_{t-1}(x_t)$. Finally, (4.14) follows from (4.7). Combining 1.), 2.) and 3.) implies that this holds with probability at least $1 - \delta$.

□

We now have a bound for r_t which holds with probability at least $1 - \delta$. The goal is to bound $\mathbb{E}[r_t]$, the current bound is not sufficient for this as the remaining probability δ could have an unbounded impact on the expectation. We would like to create a bound of the following form:

$$\mathbb{E}[r_t] \leq (1 - \delta)B_1 + \delta B_2 \tag{4.15}$$

where B_2 represents the contribution of the remaining probability δ to the expectation. One way to interpret B_2 is as an upper bound on the expectation of r_t given that the three events don't all occur $\mathbb{E}[r_t|\neg(1. \wedge 2. \wedge 3.)] \leq B_2$. In order to derive the value of B_2 we will create an algorithm agnostic bound for the regret.

4.2. Algorithm Agnostic Regret Bound

In this section we will outline how we can use an algorithm agnostic bound on r_t to bound the contribution of the remaining probability δ to the expectation of r_t. This can be viewed as bounding the tail mass:

$$\begin{aligned}\mathbb{E}[r_t] &= \int_0^\infty \rho f_{r_t}(\rho) d\rho \\ &= \int_0^a \rho f_{r_t}(\rho) d\rho + \int_a^\infty \rho f_{r_t}(\rho) d\rho\end{aligned}$$

where $f_{r_t}(\rho)$ is the PDF of r_t. First we will introduce a general lemma which we will use later in the section in order to bound the tail mass.

Lemma 4.4 Let Y be a non-negative continuous random variable taking values on $[0, \infty)$ with CDF $F_Y(y)$. If there exists a lower bound $L(y)$ on the CDF such that $F_Y(y) \geq L(y)$, $\lim_{y\to\infty} L(y) = 1$ and $L(y)$ is continuous and monotone on $[0, \infty)$, then the following bound holds:

$$\int_{F_Y^{-1}(1-\delta)}^{\infty} y f_Y(y) dy \leq \int_0^{\delta} H(p) dp$$

where $H(p)$ is the inverse of $1 - L(y)$, defined as:

$$H(p) = \{y : 1 - L(y) = p\}$$

Proof. Define the following functions (these are well defined because $F_Y(y)$ is monotone):

$$\begin{aligned} p &= F_Y(y) \\ Q_Y(p) &= F_Y^{-1}(p) \\ G_Y(p) &= (1 - F_Y(p))^{-1} \\ H(p) &= (1 - L(p))^{-1} \end{aligned}$$

where the notation g^{-1} refers to the inverse of the function of g. We have that:

$$\begin{aligned} \int_{F_Y^{-1}(1-\delta)}^{\infty} y f_Y(y) dy &= \int_{Q_Y(1-\delta)}^{\infty} y f_Y(y) dy && (4.16) \\ &= \int_{1-\delta}^{1} Q_Y(p) dp && (4.17) \\ &= \int_0^{\delta} G_Y(p) dp. && (4.18) \end{aligned}$$

Here we have that (4.16) follows from our definition of Q_Y. (4.17) follows from our definition of $p = F_Y(y)$ and hence $dp = f_Y(y)dy$ and from the fact that $Q_Y(p) = F_Y^{-1}(p) = F_Y^{-1}(F_Y(y)) = y$. Finally, (4.18) follows from the inverse of a composition of functions $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$. Hence we have that $Q_Y(p) = G_Y(1 - p)$.

It remains to show that:

$$\int_0^{\delta} G_Y(p) dp \leq \int_0^{\delta} H(p) dp.$$

Since both $1 - F_Y(y)$ and $1 - L(y)$ are monotonic decreasing with limit 0 (and $F_Y(0) = 0$ since Y is non-negative); fix some $s \in (0, 1]$ then there is some $y_1, y_2 \in [0, \infty)$ such that:

$$1 - L(y_1) = s = 1 - F_Y(y_2) \leq 1 - L(y_2)$$

where the inequality follows from $1 - F_Y(y) \leq 1 - L(y)$. Thus we have that $1 - L(y_1) \leq 1 - L(y_2) \Rightarrow y_1 \geq y_2$ by the fact that $1 - L(y_1)$ is a decreasing monotonic function. Note that by definition of the inverse function $G_Y(s) = y_2$ and $H(s) = y_1$. But this implies that $G_Y(s) \leq H(s)$ which completes the proof. A visual representation of the functions in this proof are given in Figure 4.2.

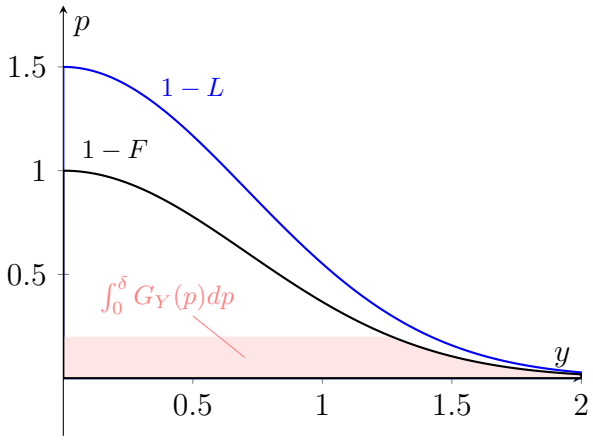


Figure 4.2: Visual representation of lemma 4.4

□

We will now return to our specific setting where we are interested in bounding the second term on the tail mass of our expected regret $\mathbb{E}[r_t]$:

$$\begin{aligned}\mathbb{E}[r_t] &= \int_0^\infty \rho f_{r_t}(\rho) d\rho \\ &= \int_0^a \rho f_{r_t}(\rho) d\rho + \int_a^\infty \rho f_{r_t}(\rho) d\rho\end{aligned}$$

where $f_{r_t}(\rho)$ is the PDF of r_t. If $F_{r_t}(\rho)$ is the cumulative distribution function and we choose $a = F_{r_t}^{-1}(1 - \delta)$ we will refer to this second integral as the contribution of the tail with probability mass δ to the expectation of r_t.

Lemma 4.5 Consider the setting of Theorem 4.1. We claim that regardless of the algorithm used to select x_t, we have the following bound on the contribution of the tail with probability mass δ to the expectation of r_t:

$$\int_{F_{r_t}^{-1}(1-\delta)}^{\infty} \rho f_{r_t}(\rho) d\rho \leq \delta r d b \left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}} \right)$$

Proof. The algorithm agnostic bound will be largely based on equation (4.2). If there is an upper bound on the derivative of the function this allows us to bound the difference between any two points within the domain D. First observe that since $D \subseteq [0, r]^d$ we have that:

$$\|x - x'\|_1 \leq rd, \quad \forall x, x' \in D.$$

Now, by 4.2 we have that with probability at least $1 - ade^{-(L/b)^2}$:

$$|f_t(x) - f_t(x')| \leq L \|x - x'\|_1 \leq Lrd = \rho \quad \forall x, x' \in D$$

this follows by the same logic as 3.) in lemma 4.3. Since we have that this inequality holds $\forall x, x' \in D$ this allows us to create a probabilistic bound on r_t, since r_t can never be more than the maximum difference between the function at two points. Observe that even if we select the worst possible x_t we still have that r_t can be at most ρ:

$$r_t = \max_{x \in D} \left[f_t(x) - f_t(x_t) \right] \leq \max_{x, x' \in D} (f_t(x) - f_t(x')) \leq \rho \tag{4.19}$$

where the second inequality holds with probability at least $1 - ade^{-(L/b)^2} = 1 - ade^{-(\rho/rdb)^2}$.

Ideally we would like to use the cumulative distribution function $F_{r_t}(\rho)$ of r_t. Unfortunately it is hard to explicitly quantify this function (especially because it also depends on the chosen x_t). Instead we will create a lower bound for $F_{r_t}(\rho)$ which holds for any x_t and show that this can be used to control the contribution of the remaining probability δ to the expectation:

$$F_{r_t}(\rho) = P(r_t \leq \rho) \geq 1 - ade^{-(\rho/rdb)^2}. \tag{4.20}$$

The inequality holds as a result of equation (4.19). We will now bound the expectation of the tail probability. (4.20) gives us that that $F_{r_t}(\rho) \geq 1 - ade^{-(\rho/rdb)^2} \Rightarrow 1 - F_{r_t}(\rho) \leq ade^{-(\rho/rdb)^2}$. The inverse of $ade^{-(\rho/rdb)^2}$ equals $rdb\sqrt{\log(\frac{ad}{\rho})}$. Then we can apply lemma 4.4:

$$\int_{F_{r_t}^{-1}(1-\delta)}^{\infty} y f_{r_t}(y) dy \leq \int_0^{\delta} rdb \sqrt{\log(\frac{ad}{\rho})} d\rho$$

We can now compute the r.h.s. of this expression to get an upper bound. The resulting upper bound is given by:

$$\int_0^{\delta} rdb\sqrt{\log(\frac{ad}{\rho})}d\rho = \left[\frac{rdb\left(2\sqrt{\log\left(\frac{ad}{\rho}\right)}\rho - \sqrt{\pi}\,ad\,\mathrm{erf}\left(\sqrt{\log\left(\frac{ad}{\rho}\right)}\right)\right)}{2}\right]_0^{\delta}$$

where erf$(\cdot)$ refers to the Gauss error function given by:

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-y^2}dy.$$

We use the following upper and lower bounds of the Gauss error function in order to simplify the expression:

$$1 - \frac{e^{-x^2}}{2x} \leq \mathrm{erf}(x) \leq 1.$$

the integral is upper bounded by:

$$rdb\left[\sqrt{\log\left(\frac{ad}{\rho}\right)}\rho - \frac{\sqrt{\pi}\,ad\,\mathrm{erf}\left(\sqrt{\log\left(\frac{ad}{\rho}\right)}\right)}{2}\right]_0^{\delta} \leq rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)}\delta + \frac{\sqrt{\pi}\delta}{2\sqrt{\log(\frac{ad}{\delta})}}\right).$$

□

This now completes the agnostic upper bound. In equation (4.15) we can say that:

$$\delta B_2 = \delta rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}}\right).$$

Combining this with the previous results in lemma 4.3 provides an upper bound on $\mathbb{E}[r_t]$:

$$\mathbb{E}[r_t] \leq (1-\delta)\left(2\beta^{1/2}\sigma_{t-1}(x_t) + \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau}\right) + \delta rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}}\right). \quad (4.21)$$

The parameters δ and τ can be chosen such that this upper bound is minimized for a given problem (there is no explicit solution but optimal values can be approximated numerically). This will also allow us to determine the optimal value of β given by $\beta = 2(\log(\frac{3}{2\delta}) + d\log(\tau))$ which is required as an input to the algorithm.

4.3. Cumulative regret

The following step will be to take sum of the instantaneous regrets $r_1 + \ldots + r_t = R_t$ so that we can consider the cumulative regret R_T. In equation (4.21) the term $\sigma_{t-1}(x_t)$ is not constant between timesteps which means we cannot take it out of the sum. However, we can bound this sum using the maximum information gain (in a similar fashion as [28]). We will handle the remaining sum using Lemma 5.3 from [28] which states that the information gain can be expressed in terms of predictive variances:

$$\mathrm{I}(\boldsymbol{y}_T; \boldsymbol{f}_T) = \frac{1}{2}\sum_{t=1}^{T}\log\left(1+\sigma^{-2}\sigma_{t-1}^2(x_t)\right). \quad (4.22)$$

However, we currently have a sum of $\sigma_{t-1}^2(x_t)$ and not $\log\left(1+\sigma^{-2}\sigma_{t-1}^2(x_t)\right)$. We must use Lemma 5.4 from [28] to rewrite this sum. We include the relevant result for completeness:

Lemma 4.6 Consider the setting of Theorem 2.1. The predictive variance of the GPR can be upper bounded as:

$$\sigma_{t-1}^2(x_t) \leq \frac{\log\left(1+\sigma^{-2}\sigma_{t-1}^2(x_t)\right)}{\log(1+\sigma^{-2})}$$

Proof. We first claim that for any $s^2 \in [0, \sigma^{-2}]$:

$$s^2 \leq \frac{\sigma^{-2}}{\log(1+\sigma^{-2})}\log(1+s^2). \quad (4.23)$$

For the case of $s^2 = 0$ and $s^2 = \sigma^{-2}$ we achieve equality. For $s^2 \in (0, \sigma^{-2})$ we will show that:

$$l(s) = \frac{\sigma^{-2}}{\log(1+\sigma^{-2})} \log(1+s^2) - s^2 \geq 0.$$

This is done by showing that the derivative is initially positive for $s^2 \in (0, a)$ and then negative for $s^2 \in (a, \sigma^{-2})$. Since $l(s) = 0$ for $s^2 = 0$ or σ^{-2} this then implies that the inequality holds on then entire interval. The derivative is given by:

$$l'(s) = \frac{2s}{\log\left(\frac{1}{\sigma^2}+1\right)\,\sigma^2\,(s^2+1)} - 2s = 2s\left[\frac{1}{\log\left(\sigma^{-2}+1\right)\,\sigma^2\,(s^2+1)} - 1\right]. \tag{4.24}$$

Note that:

$$\log(1+\sigma^{-2})\sigma^2 \leq \sigma^{-2}\sigma^2 = 1.$$

And hence:

$$\frac{1}{\log(1+\sigma^{-2})\sigma^2} \geq 1.$$

We have that equation (4.24) is positive for small s and it has at most one root on the interval $s^2 \in (0, \sigma^{-2})$ which occurs at:

$$s^2 = \frac{1}{\log\left(\frac{1}{\sigma^2}+1\right)\,\sigma^2} - 1.$$

This proves the claim (4.23). Now we will use this to complete the proof of the lemma. We have the following steps:

$$\sigma^2_{t-1}(x_t) = \sigma^2\left(\sigma^2_{t-1}(x_t)\,\sigma^{-2}\right) \leq \sigma^2 \frac{\sigma^{-2}}{\log(1+\sigma^{-2})} \log(1+\sigma^{-2}\sigma^2_{t-1}(x_t)) = \frac{\log(1+\sigma^{-2}\sigma^2_{t-1}(x_t))}{\log(1+\sigma^{-2})}$$

where the inequality follows from applying (4.23) with $s^2 = \sigma^{-2}\sigma^2_{t-1}(x_t)) \leq \sigma^{-2}k(x_t, x_t) \leq \sigma^{-2}$. Recall that we assumed in Theorem 2.1 that $k(x, x) \leq 1 \quad \forall x \in D$. This completes the proof. □

We now have all the required tools to bound the expectation of cumulative regret which is the sum of the instantaneous regrets over all timesteps. This leads to the proof of theorem 4.1.

Proof of Theorem 4.1. Let $f_{r_t}(\rho)$ denote the PDF and $F_{r_t}(\rho)$ the CDF of r_t. We start by splitting the expectation of the instantaneous regret into two parts:

$$\mathbb{E}[r_t] = \int_0^{F_{r_t}^{-1}(1-\delta)} \rho f_{r_t}(\rho) d\rho + \int_{F_{r_t}^{-1}(1-\delta)}^{\infty} \rho f_{r_t}(\rho) d\rho$$

this follows from the definition of the expectation and the fact that r_t is a non-negative random variable. Fix t (so we are considering r_t of a specific timestep) and assume that x_t is selected according to the UCB rule (algorithm 2). We can then make the following two statements:

$$\int_0^{F_{r_t}^{-1}(1-\delta)} \rho f_{r_t}(\rho) d\rho \leq (1-\delta)(2\beta^{1/2}\sigma_{t-1}(x_t) + C_2) \tag{4.25}$$

$$\int_{F_{r_t}^{-1}(1-\delta)}^{\infty} \rho f_{r_t}(\rho) d\rho \leq \delta C_3. \tag{4.26}$$

Here (4.25) follows from lemma 4.3 since it states that with probability at least $(1-\delta)$: $r_t \leq 2\beta^{1/2}\sigma_{t-1}(x_t) + C_2$. This lemma holds in the case that x_t is selected using the UCB algorithm which is assumed here. And (4.26) follows from lemma 4.5. Now we will compute the cumulative regret R_T:

$$R_T = \sum_{t=1}^{T} r_t.$$

By the linearity of the expectation we can simply sum the expectations of the instantaneous regrets r_t in order to compute the expectation of the cumulative regret. This may seem counter-intuitive as we first used lemma 4.3 to provide a probabilistic bound on r_t for some fixed t, but now we are taking the sum over all t. However, since we are simply concerned with the expectation $\mathbb{E}[r_t]$ this method is valid. In section 4.4 we consider R_T in more detail and investigate how it might deviate from it's expectation.

$$\mathbb{E}[R_T] = \sum_{t=1}^{T} \mathbb{E}[r_t] \leq \sum_{t=1}^{T} (1-\delta) \left(2\beta^{1/2}\sigma_{t-1}(x_t) + \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau} \right) + \delta rdb \left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}} \right).$$

Most terms are independent of t and can be taken out of the sum which yields:

$$\mathbb{E}[R_T] \leq T\delta rdb \left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}} \right) + (1-\delta) \left(\frac{Trdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau} + \sum_{t=1}^{T} 2\beta^{1/2}\sigma_{t-1}(x_t) \right). \tag{4.27}$$

This result can now be combined with the alternative form of the information gain presented in equation (4.22) in order to bound the sum of the predictive variances using the maximum information gain. We will bound the sum as follows (this is not a probabilistic bound, it will always hold):

$$\sum_{t=1}^{T} 2\beta^{1/2}\sigma_{t-1}(x_t) \leq \sqrt{T\sum_{t=1}^{T} 4\beta\sigma_{t-1}^2(x_t)} \tag{4.28}$$

$$\leq \sqrt{C_1 T\beta\tilde{\gamma}_T} \tag{4.29}$$

where $C_1 = \frac{8}{\log(1+\sigma^{-2})}$ and $\tilde{\gamma}_T$ is the maximum information gain for the time varying Gaussian process. Equation (4.28) follows from the fact that $\|z\|_1 \leq \sqrt{T}\|z\|_2$ for any $z \in \mathbb{R}^T$. Equation (4.29) follows from applying lemma 4.6 and equation (4.22).

Similarly to previous results in literature ([28], [2]) we observe that there is a strong connection between the maximum information gain and the cumulative regret. A higher maximum information gain will result in a higher cumulative regret because it is "harder" to learn the underlying function. Placing this result into (4.27) yields:

$$\mathbb{E}[R_T] \leq (1-\delta)\left[\sqrt{C_1 T\beta\tilde{\gamma}_T} + TC_2\right] + T\delta C_3 \tag{4.30}$$

with:

$$C_1 = \frac{8}{\log(1+\sigma^{-2})}$$

$$C_2 = \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau}$$

$$C_3 = rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}} \right).$$

Now we have an upper bound for the expectation of R_T which completes the proof.

□

In theorem 3.2 we showed that the maximum information gain $\tilde{\gamma}_T$ for the time varying function f_t can be bounded:

$$\tilde{\gamma}_T \leq \left(\frac{T}{\tilde{N}} + 1\right)\left(\gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\varepsilon\right)$$

where $\tilde{N} \in \mathbb{N}$ is a parameter which can be chosen. This result was discussed in more detail in chapter 3 equation (3.9). This inequality implies that $\tilde{\gamma}_T = \mathcal{O}(T)$ which means that we can now state that with our bound $\mathbb{E}[R_T] = \mathcal{O}(T)$.

4.4. Convergence

We have provided an upper bound for the expectation of R_T. But how close can we expect the cumulative regret R_T to be to $\mathbb{E}[R_T]$? And with what probability can we expect the algorithm to under-perform and achieve a much higher regret $R_T >> \mathbb{E}[R_T]$ if we get unlucky? To answer this question we introduce the following lemma:

Lemma 4.7 Consider the setting of Theorem 4.1. Then for any $\omega \in (0, 1)$ we have with probability at least $(1 - \omega)$:

$$R_T \leq \mathbb{E}[R_T] + \frac{3T}{\sqrt{\omega}}.$$

Proof. We start by providing a bound on this variance based on the Cauchy–Schwarz inequality. We are interested in bounding:

$$\mathrm{Var}\left(R_T\right) = \mathrm{Var}\left(\sum_{t=1}^{T} r_t\right) = \sum_{i,j=1}^{T} \mathrm{Cov}\left(r_i, r_j\right) \tag{4.31}$$

where the second equality follows from Bienaymé's identity [17]. Now, we are left to bound $\mathrm{Cov}\left(r_i, r_j\right)$. Let's start by considering the case of $i = j$, in other words $\mathrm{Var}[r_t]$. Recall that the instantaneous regret is defined as:

$$r_t = \max_{x \in D} \left[f_t\left(x\right) - f_t\left(x_t\right) \right] = \max_{x \in D} \left[f_t\left(x\right) \right] - f_t\left(x_t\right).$$

The difficulty in analysing this term stems from the $\max_{x \in D} \left[f_t\left(x\right) \right]$ which is the maximum of $f_t \sim \mathcal{GP}(0, k)$. We will use the Borell-TIS inequality [1]:

Theorem 4.2 (Borell–TIS inequality) Let T be a topological space and let $X(t)$ be a centered Gaussian process with $\sup_{t \in T} X(t)$ a.s. finite. Let $\sigma = \mathbb{E}[X(t)]$. Then

$$\mathbb{P}\left(\sup_{t \in T} X(t) > \mathbb{E}[\sup_{t \in T} X(t)] + u \right) \leq e^{-u^2/2\sigma^2}$$

and by symmetry:

$$\mathbb{P}\left(|\sup_{t \in T} X(t) - \mathbb{E}[\sup_{t \in T} X(t)]| > u \right) \leq 2e^{-u^2/2\sigma^2}.$$

Which implies that:

$$\begin{aligned}
\mathrm{Var}\left(\sup_{t \in T} X(t) \right) &= \mathbb{E}[\sup_{t \in T} X(t) - \mathbb{E}[\sup_{t \in T} X(t)]] \\
&= \int_0^\infty 2u\mathrm{d}u\mathbb{P}\left(|\sup_{t \in T} X(t) - \mathbb{E}[\sup_{t \in T} X(t)]| > u \right) \\
&\leq \int_0^\infty 2u\mathrm{d}u 2e^{-u^2/2\sigma^2} \\
&= 4\sigma^2
\end{aligned}$$

Recall that in our case we have made the assumption that $k(x, x) \leq 1$. This means that $\mathrm{Var}[f_t(x)] \leq 1$ for all $x \in D$. By theorem 4.2 this implies that:

$$\mathrm{Var}[\max_{x \in D} \left[f_t\left(x\right) \right]] \leq 4 \tag{4.32}$$

We can now bound the variance of r_t:

$$\mathrm{Var}[r_t] = \mathrm{Var}[\max_{x \in D} \left[f_t\left(x\right) \right]] + \mathrm{Var}[f_t(x_t)] + 2\mathrm{Cov}\left(\max_{x \in D} \left[f_t\left(x\right) \right], f_t(x_t) \right) \leq 9 \tag{4.33}$$

Where the equality follow again from Bienaymé's identity and the inequality follows from equation (4.32) combined with the Cauchy–Schwarz inequality which states that:

$$\mathrm{Cov}[X, Y] \leq \sqrt{\mathrm{Var}[X]\mathrm{Var}[Y]}.$$

Since equation (4.33) hold for all t we can now use the same Cauchy–Schwarz inequality to state that:

$$\mathrm{Cov}\left(r_i, r_j\right) \leq 9.$$

This yields an upper bound for the variance of R_T by evaluating equation (4.31):

$$\mathrm{Var}\left(R_T\right) \leq 9T^2$$

In order to translate this bound on the variance into a probabilistic bound on R_T we shall use Chebyshev's inequality [9]:

Lemma 4.8 (Chebyshev's inequality)

Let X be a random variable with finite variance $\sigma^2 > 0$ with mean $\mu \in \mathbb{R}$. Then for any $k \in \mathbb{R}$:

$$\mathbb{P}(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

Now suppose we want a bound which holds with probability at least $(1 - \omega)$. Then we apply lemma 4.8 with $k = \frac{1}{\sqrt{\omega}}$. Note that we have shown $\sigma^2 \leq 9T^2$. This yields the bound:

$$\mathbb{P}(R_T \geq \mathbb{E}[R_T] + \frac{3T}{\sqrt{\omega}}) \leq \omega$$

and thus with probability at least $(1 - \omega)$:

$$R_T \leq \mathbb{E}[R_T] + \frac{3T}{\sqrt{\omega}}. \tag{4.34}$$

□

As both terms on the right hand side are $\mathcal{O}(T)$ we now have a probabilistic bound of $\mathcal{O}(T)$ which holds for any $\omega \in (0, 1)$. This bound can be compared with the result achieved in [2]. We have introduced a new variable ω which was not present in the analysis of the original regret bound [2]. This is because in the original regret bound the choice of δ influenced β as well as the probability that the bound would hold. In this new bound δ is simply a variable used for the analysis (and it plays a similar role to the proof of the previous bound). However, δ no longer impacts the probability that the bound will hold. This is convenient as the user should simply choose δ and τ in order to minimize the r.h.s. of equation (4.4). These will determine the choice of β (constant), but they will not impact the probability. For applications, if the user wants to determine the probability of a certain bound holding they can use equation (4.34) to determine which bound will hold with what probability. There is an optimal β regardless of the desired certainty of the bound holding.

4.5. Improved Convergence Conjecture

We are not completely satisfied with the bound obtain in section 4.4. While we have shown that $R_T = \mathcal{O}(T)$, we believe it should be possible to show that $\frac{R_T}{T}$ will converge in probability to $\frac{\mathbb{E}[R_T]}{T}$. This is equivalent to showing that for any $\eta > 0$:

$$\lim_{T\to\infty} \mathbb{P}\left[\left|\frac{R_T}{T} - \frac{\mathbb{E}[R_T]}{T}\right| > \eta\right] = 0.$$

On a high level we expect that this is possible because the instantaneous regrets $r_t, r_{t'}$ for two timesteps t, t' will become increasingly uncorrelated for large differences in time $|t - t'|$. We can then apply the following lemma:

Lemma 4.9 Consider a sum of random variables X_i with finite variance:

$$S_T = \sum_{i=1}^{T} X_i.$$

Note that the RVs X_i are not required to be independent or identically distributed, they must each individually have a finite variance. We claim that if we have a bound on the covariance of these random variables of the following form:

$$\mathrm{Cov}(X_i, X_j) \leq m \cdot c^{|i-j|}$$

with $c \in [0, 1)$ and $m \in \mathbb{R}^+$, then we can make the following statements about the distribution of S_T:

1.) For any $\eta > 0$:

$$\lim_{T\to\infty} \mathbb{P}\left[\left|\frac{S_T}{T} - \frac{\mathbb{E}[S_T]}{T}\right| > \eta\right] = 0.$$

2.) For any $\omega \in (0, 1)$, we have with probability at least $1 - \omega$:

$$S_T \leq \mathbb{E}[S_T] + \sqrt{m\left[T + \frac{2c}{(c-1)^2}\left(c^{T+1} + 1\right) + \frac{2c}{1-c}T\right]\frac{1}{\omega}}.$$

Proof. 1.) Choose some $\eta > 0$ then for any $T \in \mathbb{N}$ we have by the Chebyshev inequality:

$$\mathbb{P}\left[\left|\frac{S_T}{T} - \frac{\mathbb{E}[S_T]}{T}\right| > \eta\right] \leq \frac{\mathrm{Var}\left(\frac{S_T}{T}\right)}{\eta^2}. \tag{4.35}$$

It remains to bound $\mathrm{Var}\left(\frac{S_T}{T}\right)$:

$$\mathrm{Var}\left(\frac{S_T}{T}\right) = \frac{\mathrm{Var}\left(S_T\right)}{T^2} = \frac{\sum_{i=1}^{T}\sum_{j=1}^{T}\mathrm{Cov}(X_i, X_j)}{T^2} \leq \frac{m\sum_{i=1}^{T}\sum_{j=1}^{T}c^{|i-j|}}{T^2}$$

We can compute the nested sum using the properties of geometric series:

$$\begin{aligned}
\sum_{i=1}^{T}\sum_{j=1}^{T} c^{|i-j|} &= \sum_{i=1}^{T} c^{|i-i|} + 2\sum_{i=1}^{T}\sum_{j=1}^{i-1} c^{|i-j|} \\
&= T + 2\sum_{i=1}^{T}\sum_{j=1}^{i-1} c^{|i-j|} \\
&= T + 2\sum_{i=1}^{T}\sum_{j=1}^{i-1} c^{i-j} \\
&= T + 2\sum_{i=1}^{T} c^{i}\frac{1-c^{-i}}{1-c^{-1}} \\
&= T + 2\sum_{i=1}^{T} \frac{c^{i}-1}{1-c^{-1}} \\
&= T + \frac{2c}{c-1}\sum_{i=1}^{T} c^{i} - 1 \\
&= T + \frac{2c}{c-1}\left(\frac{1-c^{T+1}}{1-c} - T\right) \\
&= T + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right) + \frac{2c}{1-c}T
\end{aligned}$$

Substituting this back into equation (4.35) and taking the limit yields:

$$\lim_{T\to\infty} \mathbb{P}\left[\left|\frac{S_T}{T} - \frac{\mathbb{E}[S_T]}{T}\right| > \eta\right] \leq \lim_{T\to\infty} \frac{\mathrm{Var}\left(\frac{S_T}{T}\right)}{\eta^2} \leq \lim_{T\to\infty} \frac{m\left[T + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right) + \frac{2c}{1-c}T\right]}{T^2\eta^2} = 0.$$

The choice of η was arbitrary which implies that the equation holds for any $\eta > 0$ which means $\frac{S_T}{T}$ converges to $\frac{\mathbb{E}[S_T]}{T}$ in probability.

2.) By the same logic we have that:

$$\mathbb{P}\left[|S_T - \mathbb{E}[S_T]| > \eta\right] \leq \frac{m\left[T + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right) + \frac{2c}{1-c}T\right]}{\eta^2}.$$

Choose $\omega \in (0,1)$ now we define:

$$\eta = \sqrt{m\left[T + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right) + \frac{2c}{1-c}T\right]\frac{1}{\omega}}.$$

Then we obtain:

$$\mathbb{P}\left[|S_T - \mathbb{E}[S_T]| > \eta\right] \leq \omega$$

So we have with probability at least $1 - \omega$:

$$S_T \leq \mathbb{E}[S_T] + \eta = \mathbb{E}[S_T] + \sqrt{m\left[T + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right) + \frac{2c}{1-c}T\right]\frac{1}{\omega}}.$$

□

This lemma can't be directly applied to our sum of r_t's as we must have a bound on the covariance of the form:

$$\mathrm{Cov}(r_i, r_j) \leq m \cdot c^{|i-j|}$$

We introduce the following conjecture which makes this possible:

Conjecture 4.1 Let $D \subseteq [0,r]^d$ be compact and convex. We sample two functions (i.i.d.) from a Gaussian process $g_1, g_2 \sim \mathcal{GP}(0,k)$ with $k(x,x) \leq 1$. Define:

$$f_1(x) := g_1(x)$$

$$f_2(x) := \sqrt{1-\varepsilon}g_1(x) + \sqrt{\varepsilon}g_2(x)$$

where $\varepsilon \in [0,1]$.

$$\mathrm{Corr}\left(f_1(x), f_2(x')\right) \leq \sqrt{1-\varepsilon} \qquad \forall x, x' \in D \tag{4.36}$$

$$\mathrm{Corr}\left(\max_{x\in D}\left[f_1(x)\right], \max_{x\in D}\left[f_2(x)\right]\right) \leq \sqrt{1-\varepsilon} \tag{4.37}$$

$$\mathrm{Corr}\left(\max_{x\in D}\left[f_1(x)\right], f_2(x)\right) \leq \sqrt{1-\varepsilon} \qquad \forall x \in D. \tag{4.38}$$

We can show that equation (4.36) holds by simply considering the covariance of our two functions for any arbitrary $x, x' \in D$. First observe that we can write f_2 as:

$$f_2(x) = \sqrt{1-\varepsilon} f_1(x) + \sqrt{\varepsilon} g_2(x) \tag{4.39}$$

$$\mathrm{Cov}(f_1(x), f_2(x')) = \sqrt{1-\varepsilon}\mathbb{E}[f_1(x) f_2(x')] = \sqrt{1-\varepsilon} k(x, x') \leq \sqrt{1-\varepsilon}$$

where $k(x, x')$ is the spatial covariance due to the kernel k. The first equality follows from the definition of covariance and equation (4.39) with the fact that $g_2(x')$ is independent of $f_1(x)$ for all $x, x' \in D$. The second equality follows from our prior on $f_1, f_2 \sim \mathcal{GP}(0, k)$. Recall that we assumed $k(x, x) \leq 1$ which yields the inequality and hence the first statement in the conjecture holds.

Unfortunately it is not possible to prove equation (4.37) and equation (4.38) in a similar manner. This is due to the complexities associated with analysing the probability distribution of the maximum of $f_1, f_2 \sim \mathcal{GP}(0, k)$. The conjecture clearly holds for $\varepsilon = 0$ (from the bound on the variance provided by theorem 4.2) and $\varepsilon = 1$ since the two functions are independent in that case. We will assume the conjecture holds for $\varepsilon \in (0, 1)$ and show that we can use this to prove that $\frac{\mathbb{E}[R_T]}{T}$ converges in probability.

Lemma 4.10 Consider the setting of theorem 4.1. Assume that conjecture 4.1 holds. Then for any $\omega \in (0, 1)$ we have that the following statement holds with probability at least $(1 - \omega)$:

$$R_T \leq \mathbb{E}[R_T] + \sqrt{9\left[T\frac{1 + (1-\varepsilon)^{1/2}}{1 - (1-\varepsilon)^{1/2}} + \frac{2(1-\varepsilon)^{1/2}}{((1-\varepsilon)^{1/2} - 1)^2}\left((1-\varepsilon)^{(T+1)/2} + 1\right)\right]\frac{1}{\omega}} \tag{4.40}$$

and as $T \to \infty$, the average regret will converge in probability to the expected average regret:

$$\frac{R_T}{T} \to \frac{\mathbb{E}[R_T]}{T}.$$

Proof. First, recall that the time varying function f_t is generated according to (4.1):

$$\begin{aligned} f_1(x) &= g_1(x) \\ f_{t+1}(x) &= \sqrt{1-\varepsilon} f_t(x) + \sqrt{\varepsilon} g_{t+1}(x) \quad \forall t \geq 1, \end{aligned}$$

where $g_i \sim \mathcal{GP}(0, k)$ i.i.d. and $\varepsilon \in (0, 1]$. Observe that we can write:

$$f_{t+j}(x) = (1-\varepsilon)^{j/2} \cdot f_t(x) + \sqrt{\varepsilon} \sum_{i=1}^{j} (1-\varepsilon)^{(j-i)/2} g_{t+i}(x) \tag{4.41}$$

This means that the covariance of the function at two different timesteps t and $t+j$ and two spatial locations $x, x' \in D$ is of the following form:

$$\text{Cov}(f_t(x), f_{t+j}(x')) = (1-\varepsilon)^{j/2} \mathbb{E}[f_t(x) f_t(x')] = (1-\varepsilon)^{j/2} k(x, x') \leq (1-\varepsilon)^{j/2} \tag{4.42}$$

where $k(x, x')$ is the spatial covariance due to the kernel k. The first equality follows from the definition of covariance and equation (4.41) with the fact that $g_{t+i}(x)$ is independent of $f_t(x)$ for all $i \geq 1$. The second equality follows from our prior on $f_t \sim \mathcal{GP}(0, k)$ for all t. Recall that we assumed $k(x, x) \leq 1$ which yields the inequality. Now, consider the instantaneous regret associated with two timesteps $t, t' \geq 1$:

$$r_t = f_t(x_t^*) - f_t(x_t) \qquad r_{t'} = f_{t'}(x_{t'}^*) - f_{t'}(x_{t'})$$

where $f_t(x_t^*) = \max_{x \in D} [f_t(x)]$. We can express the covariance of the instantaneous regrets:

$$\begin{aligned} \text{Cov}\left[r_t, r_{t'}\right] &= \text{Cov}\left[f_t(x_t^*) - f_t(x_t), f_{t'}(x_{t'}^*) - f_{t'}(x_{t'})\right] \\ &= \text{Cov}\left[f_t(x_t^*), f_{t'}(x_{t'}^*)\right] + \text{Cov}\left[f_t(x_t), f_{t'}(x_{t'})\right] - \text{Cov}\left[f_t(x_t^*), f_{t'}(x_{t'})\right] - \text{Cov}\left[f_t(x_t), f_{t'}(x_{t'}^*)\right] \\ &\leq 9c^{|t-t'|}. \end{aligned} \tag{4.43}$$

with $c = (1-\varepsilon)^{1/2}$, the inequality holds due to equation (4.42) combined with conjecture 4.1. Note that here we applied the assumption that $-1 \leq k(x, x') \leq 1$. In most common kernels used for Bayesian optimization (such as the RBF kernel or the Matérn kernel) we actually have $0 \leq k(x, x') \leq 1$ which would improve the final inequality to $8c^{|t-t'|}$ rather than $9c^{|t-t'|}$. Equation (4.43) provides an upper bound on the covariance between two regrets. Now we can apply lemma 4.9 which states that for any $\eta > 0$:

$$\lim_{T \to \infty} \mathbb{P}\left[\left|\frac{R_T}{T} - \frac{\mathbb{E}[R_T]}{T}\right| > \eta\right] = 0.$$

as η is arbitrary this means we have that:

$$\frac{R_T}{T} \to \frac{\mathbb{E}[R_T]}{T}$$

in probability. Additionally we have from lemma 4.9 that for any $\omega \in (0,1)$ we can achieve a bound on $\frac{R_T}{T}$ which holds with probability $(1-\omega)$. This is achieved by choosing:

$$\begin{aligned}\eta &= \sqrt{9\left[T + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right) + \frac{2c}{1-c}T\right]\frac{1}{\omega}} \\ &= \sqrt{9\left[T\frac{1+c}{1-c} + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right)\right]\frac{1}{\omega}}.\end{aligned}$$

Then we have that:

$$\mathbb{P}\left[|R_T - \mathbb{E}[R_T]| > \eta\right] \leq \frac{4\left[T\frac{1+c}{1-c} + \frac{2c}{(c-1)^2}\left(c^{T+1}+1\right)\right]}{\eta^2} = \omega.$$

So we have with probability at least $(1-\omega)$:

$$R_T \leq \mathbb{E}[R_T] + \eta = \mathcal{O}(T) \tag{4.44}$$

since $\mathbb{E}[R_T] = \mathcal{O}(T)$ by (4.30). This concludes the proof of Theorem 4.1. □

We have improved the regret bound provided in [2] and shown that it is possible to achieve this regret with a constant value of β. The optimal value of β depends on the parameters of the problem. We achieved the probabilistic bound in equation (4.44) by taking advantage of the fact that the time-varying function f_t allows us to bound the covariance between regrets at different timesteps. Our method varies from [2] in that rather than taking a union bound over every timestep, which requires an unbounded increasing β_t, we can bound the instantaneous regret and show that the variance of the average regret converges to zero in the limit. This method of bounding the regret would not be feasible for a static function and hence only holds for $\varepsilon \in (0,1]$ as this is required to upper bound the covariance when applying lemma 4.9.

4.6. Optimal Beta Dependence on Epsilon

It is interesting to consider how the optimal β might change with rate of change ε of the function f_t. In the regret bounds provided in [2] the choice of β was independent of ε. For our bound we want to choose a $\beta = 2(\log(\frac{3}{2\delta}) + d\log(\tau))$ such that the r.h.s. is minimized:

$$\mathbb{E}[R_T] \leq (1-\delta)\left[\sqrt{C_1 T\beta(\delta,\tau)\tilde{\gamma}_T} + TC_2(\delta,\tau)\right] + T\delta C_3(\delta). \tag{4.45}$$

Depending on the choice of δ and τ we are making a trade-off between the algorithm specific regret $(1-\delta)\sqrt{C_1 T\beta(\delta,\tau)\tilde{\gamma}_T}$, and the overhead terms $T(1-\delta)C_2(\delta,\tau)$ and $T\delta C_3(\delta)$. Unfortunately it is not possible to explicitly solve for the optimal choice of δ and τ. On a high level we see that increasing β will decrease the overhead terms (but increase the first term). If we consider for example the square exponential kernel; [2] showed that $\tilde{\gamma}_T = \mathcal{O}(T\varepsilon^{1/5})$. Suppose we consider a function which is changing more rapidly; increase ε and thereby increase $\tilde{\gamma}_T$. This would increase only the first term on the r.h.s. of equation (4.45) which implies that we should lower the value of β to find a new optimal bound. This is intuitive as it means the model should put less effort into exploration when the function is changing more rapidly. Exploring has less value as the data will become stale more quickly. The inverse is also true; if we have a more constant function (lower ε) it is more worthwhile to explore (and we should increase β).

In a more detailed analysis we can fix τ and investigate how varying δ will affect the expected cumulative regret bound. We will make a number of assumptions and then show that we expect the optimal β to decrease with the rate of the change of the function ε. We will consider the case where the spatial kernel is the square exponential kernel and the maximum information gain is $\mathcal{O}(T\varepsilon^{2/5})$. We start with:

$$\begin{aligned}\mathbb{E}[R_T] &\leq (1-\delta)\left(\sqrt{C_1 T\beta\tilde{\gamma}_T} + TC_2\right) + T\delta C_3 \\ &\simeq T\left[\varepsilon^{1/5}(1-\delta)\sqrt{C_1\beta} + (1-\delta)C_2 + \delta C_3\right]\end{aligned}$$

Since we simply wanna minimize the bound we can divide by T and consider the remaining term:

$$U = \varepsilon^{1/5}(1-\delta)\sqrt{C_1\beta} + (1-\delta)C_2 + \delta C_3.$$

By optimizing δ we want to find δ such that:

$$\frac{dU}{d\delta} = 0.$$

Let's consider the derivatives of the three components of U separately:

$$K_1 = \frac{d(1-\delta)\sqrt{C_1\beta}}{d\delta} \simeq \frac{d(1-\delta)\sqrt{\log(\frac{3}{2\delta})}}{d\delta} = -\frac{\left(2\log\left(\frac{3}{2\delta}\right)-1\right)\delta+1}{2\sqrt{\log\left(\frac{3}{2\delta}\right)}\,\delta} = -\sqrt{\log(\frac{3}{2\delta})} + \frac{\delta-1}{\sqrt{\log(\frac{3}{2\delta})}} < 0$$

$$K_2 = \frac{d(1-\delta)\frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau}}{d\delta} \simeq \frac{d(1-\delta)\sqrt{\log(\frac{3ad}{\delta})}}{d\delta} = -\frac{\left(2\log\left(\frac{3ad}{\delta}\right)-1\right)\delta+1}{2\sqrt{\log\left(\frac{3ad}{\delta}\right)}\,\delta} < 0$$

$$K_3 = \frac{d\delta C_3}{d\delta} \simeq \delta\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{1}{\sqrt{\log\left(\frac{ad}{\delta}\right)}}\right) = \frac{2\log^2\left(\frac{ad}{\delta}\right)+\log\left(\frac{ad}{\delta}\right)+1}{2\log^{\frac{3}{2}}\left(\frac{ad}{\delta}\right)} > 0.$$

Note that for K_1 we are not including $\varepsilon^{1/5}$ which is a constant that is independent of δ. If we are located at an optimum (wrt δ) then these derivatives should sum to zero:

$$U' = \frac{dU}{d\delta} = \varepsilon^{1/5}K_1 + K_2 + K_3 = 0$$

Now if we increase ε the first term will increase while the other terms remain the same. But this then implies that the derivative wrt U is negative:

$$\frac{dU}{d\delta} < 0$$

since we want to minimize U this implies we should increase δ until this gradient is again equal to zero. Increasing δ leads to a smaller β; as we expected a larger ε has lead to a smaller optimal β. While we can't explicitly solve for the relation between ε and the β at the optimal values of δ and τ this provides some motivation for how the ε will affect the optimal β via $\tilde{\gamma}_T$. This is an advantage when comparing this regret bound to the one presented in [2] (as their β was independent of ε). In chapter 5 we perform simulation studies to investigate if this theoretical result can be supported by data.

4.7. Comparison with Previous Regret Bound

The benefits of this new regret bound are listed as follows:

- Scaling w.r.t. T: The previous regret bounds showed that $\frac{R_T}{T} = \mathcal{O}(\sqrt{\log(T)})$. The new bounds achieve linear cumulative regret which results in $\frac{R_T}{T} = \mathcal{O}(1)$.
- Probabilistic bounds: The previous result required the user to pick δ and choose β_t accordingly before running the algorithm. This would then result in a bound on the regret which holds with probability $(1-\delta)$. The new bounds allow the user to pick

the optimal β regardless of the desired probability. After running the algorithm the user can compute an upper bound on the regret for any desired probability $(1-\omega)$ according to equation (4.40).

- Algorithm dependence on ε: In the previous regret bounds β_t was chosen based on the desired probability of the bound holding according to equation (3.6). For new regret bounds we choose β such that equation (4.4) is minimized. This optimal value will depend on ε which means our algorithm will act differently depending on the rate of change of the function f_t, as described in section 4.6.

Now, let us consider how the algorithm performs as $\varepsilon \to 0$. For $\varepsilon = 0$ we simply have the static case which has been studied in detail and allows us to achieve sub-linear regret [28]. However, it is interesting to consider how well our algorithm might perform for small ε. Let us consider how cumulative regret scales as a function of T and ε:

$$R_T = \mathcal{O}(L(T,\varepsilon)) \quad \text{as} \quad T \to \infty \quad \text{and} \quad \varepsilon \to 0.$$

It has been shown (in Theorem 4.1 [2]) that a lower bound on this scaling is $T\varepsilon$. When designing an algorithm we aim to achieve regret bound of the form $\mathcal{O}(T\varepsilon^{\alpha})$ with the highest possible α.

Let us consider the square exponential kernel to compare the two bounds. Using the corrected version of the maximum information gain for the time varying function presented in theorem 3.2 we find that $\tilde{\gamma}_T = \mathcal{O}(T\varepsilon^{2/5})$ by choosing $\tilde{N} = \varepsilon^{-2/5}$. Then, for the regret bounds from [2] given in (3.7) we have:

$$R_T = \mathcal{O}(T\sqrt{\log(T)}\varepsilon^{1/5}).$$

Now, recall that our new regret bounds are of the from:

$$(1-\delta)\left(\sqrt{C_1 T\beta\tilde{\gamma}_T} + TC_2\right) + T\delta C_3 \leq \sqrt{C_1 T\beta\tilde{\gamma}_T} + TC_2 + T\delta C_3$$

with:

$$
\begin{aligned}
C_1 &= \frac{8}{\log(1+\sigma^{-2})} \\
C_2 &= \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau} \\
C_3 &= rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}}\right).
\end{aligned}
$$

The first term is clearly $\mathcal{O}(T\varepsilon^{1/5})$ based on $\tilde{\gamma}_T$. Unfortunately, the other two terms seem to be independent of ε. This would imply that for our regret bound we can only achieve $\mathcal{O}(T)$ which is independent of ε. However, recall that δ and τ were variables which could be chosen. If for example we choose:

$$
\begin{aligned}
\delta &= \varepsilon^{2/5} \\
\tau &= \left\lceil \varepsilon^{-2/5} \right\rceil.
\end{aligned}
$$

We observe that when we consider C_2 and δC_3 as $\varepsilon \to 0$ we obtain:

$$
\begin{aligned}
C_2 &= \mathcal{O}(\frac{\sqrt{\log(\varepsilon^{-2/5})}}{\varepsilon^{-2/5}}) = \mathcal{O}(\varepsilon^{-1/5}) \\
\delta C_3 &= \mathcal{O}(\varepsilon^{2/5}\sqrt{\log(\varepsilon^{-2/5})}) = \mathcal{O}(\varepsilon^{-1/5}).
\end{aligned}
$$

Therefore we have $R_T = \mathcal{O}(T)\varepsilon^{1/5}$. While these choices of τ and δ might not be optimal, it does confirm that with this bound we are able to achieve at least as good scaling (w.r.t. to ε) as the previous bound. Our regret bound improves on the scaling w.r.t. T and matches the scaling w.r.t. ε.

It is interesting to note that, while we might not be using the optimal values for δ and τ, this result could also give us some intuition for how the optimal β might scale as a function of ε (especially for small values of ε):

$$
\beta = 2(\log(\frac{3}{2\delta}) + d\log(\tau)) = \mathcal{O}(-\log(\varepsilon)) \quad \text{as} \quad \varepsilon \to 0
$$

where we used the values of $\delta = \varepsilon^{2/5}$ and $\tau = \left\lceil \varepsilon^{-2/5} \right\rceil$.

5
Simulation Study

In chapter 4 have provided theoretical results which imply that using the TV-GP-UCB algorithm 2 with a constant value of β will result in lower cumulative regret compared to the traditional scaling of the form $\beta_t = \mathcal{O}(\log(t))$. In this chapter we will support these theoretical results by testing the algorithm on synthetic data. We consider the two dimensional case with a grid of 50 x 50 points. We have $D = [0, 1]^2$ and we generate the data using the squared exponential kernel (2.1) with the length-scale parameter set equal to $\sqrt{0.2}$. For the case of increasing β_t we set values according to the heuristic used in past research ([28], [15], [2]); $\beta_t = 0.8 \log(4t)$. We test this against various constant values of β sampled in an evenly spaced manner on a log scale.

In figure 5.1 we present the results of testing the algorithm in this setting with rate of change $\varepsilon = 0.09$ and observation noise $\sigma = 0.1$. We observe that there is indeed a constant value of β which outperforms the $\beta_t = 0.8 \log(4t)$ heuristic. In fact, the estimated average regret falls below the average regret of the increasing β_t for all $\beta \in [0.5, 5.0]$ and we achieve the best performance for $\beta = 2.0$.

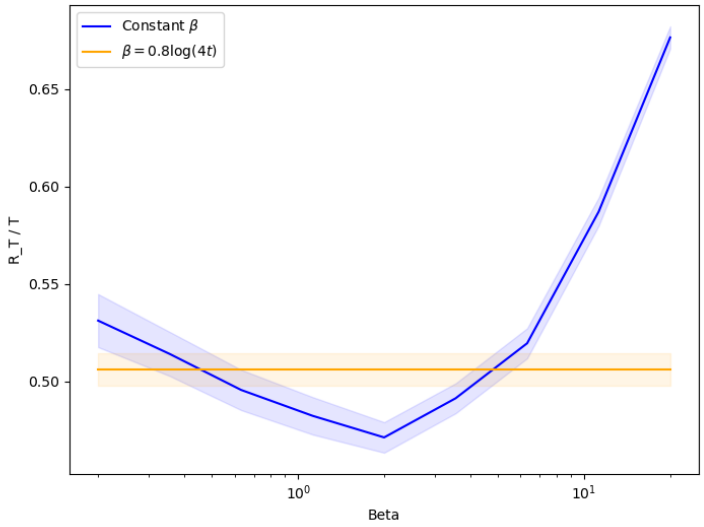


Figure 5.1: Average regret after 200 timesteps vs β. We compare constant values of β (shown on the horizontal axis with log scale) to $\beta_t = 0.8 \log(4t)$ including the 95% confidence interval of the mean after 200 trials. [$\varepsilon = 0.09$ and $\sigma = 0.1$]

In figure 5.2 the results of a similar test with lower ε are presented. We first observe that the average regret for this setting is lower. This is as a result of a slower changing function f_t which allows the algorithm to put more weight on older data which results in a better estimate of the current function. Additionally, we observe that the confidence intervals are larger, this supports the result of (lemma 4.9):

$$\text{Var}\left(\frac{R_T}{T}\right) = \mathcal{O}\left(\frac{1+(1-\varepsilon)^{1/2}}{T(1-(1-\varepsilon)^{1/2})}\right) \quad \text{as} \quad T \to \infty.$$

Which shows that the variance of $\frac{R_T}{T}$ will increase with decreasing ε. We also observe that there is a smaller range in which our constant value of β outperforms. In this case the range is $\beta \in [1.0, 4.0]$.

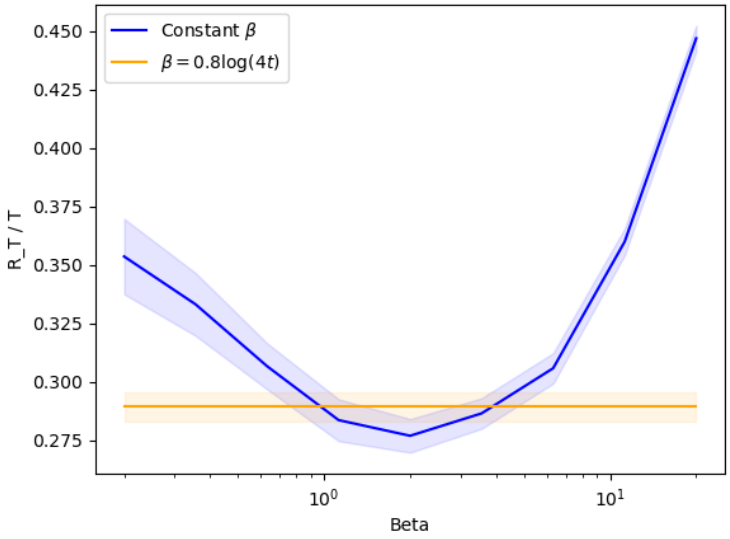
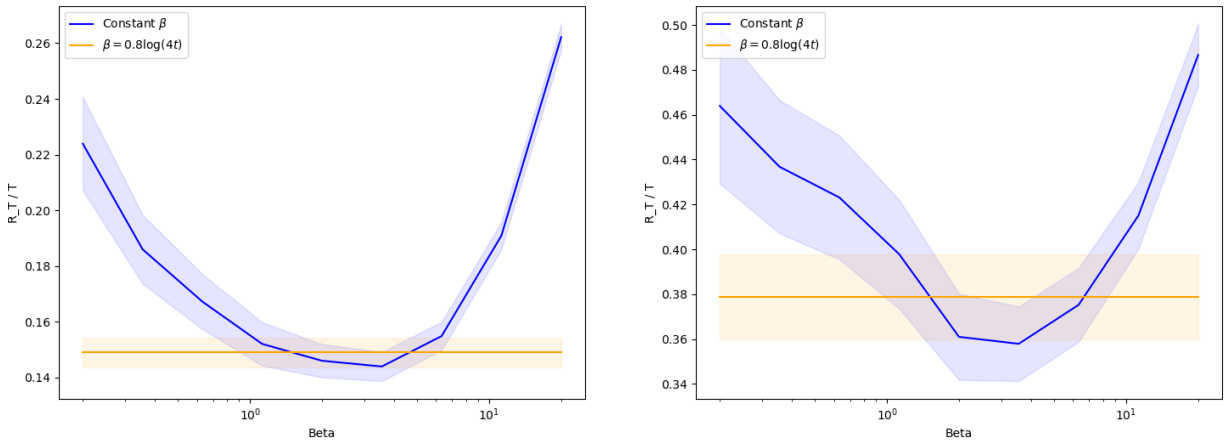


Figure 5.2: Average regret after 200 timesteps vs β. We compare constant values of β (shown on the horizontal axis with log scale) to $\beta_t = 0.8\log(4t)$ including the 95% confidence interval of the mean after 200 trials. [$\varepsilon = 0.03$ and $\sigma = 0.1$]

Next, we investigate the impact of the variance of the noise σ^2 from our samples $y_t = f(x_t) + z_t$ with $z_t \sim \mathcal{N}(0, \sigma^2)$. We present figure 5.3 which are tests performed with very small and very large noise variance respectively. We first observe that the average regret in the high variance case is significantly higher than the low variance case. This is to be expected as the samples are less informative about the underlying function. Additionally, we see that the confidence intervals for the high variance case are quite large. This is a result of the algorithm's choices of x_t becoming more random due to the high amount of noise. Due to the large confidence intervals (and possible small or non-existent effect) the results are inconclusive about the impact of σ on the optimal choice of β. Although there is

a high uncertainty in the estimates, in both cases the expected cumulative regret is lower for constant $\beta \in [1.5, 5.0]$.



(a) [$\varepsilon = 0.01$ and $\sigma = 0.01$]

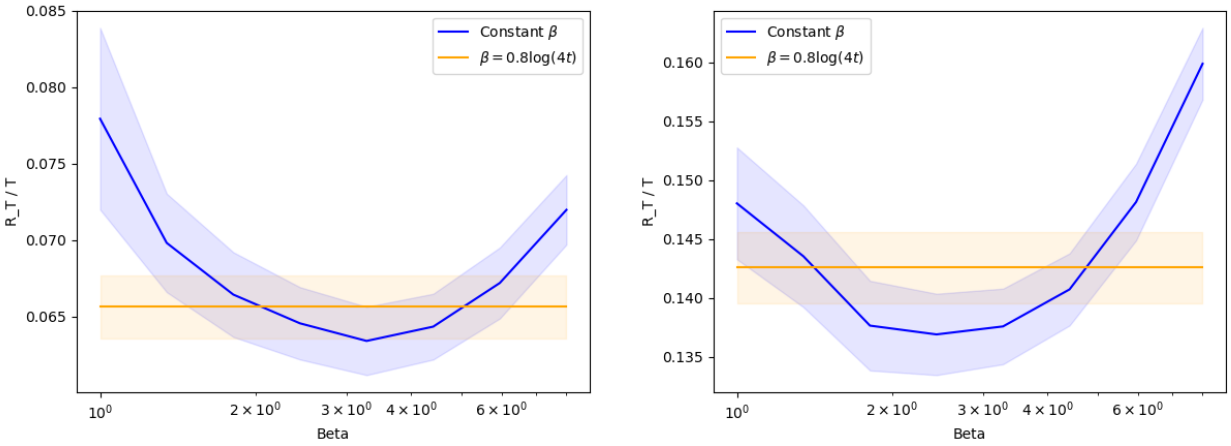
(b) [$\varepsilon = 0.01$ and $\sigma = 1.0$]

Figure 5.3: Average regret after 200 timesteps vs β. We compare constant values of β (shown on the horizontal axis with log scale) to $\beta_t = 0.8 \log(4t)$ including the 95% confidence interval of the mean after 200 trials.

These results support the theoretical result that a constant value for β is able to achieve lower cumulative regret. The optimal value of β will depend on the problem. However in these experiments we found that a value of $\beta \in [2.0, 4.0]$ is a good heuristic when knowledge of the underlying function is limited. In fact, values of β in this range consistently outperform the heuristic $\beta_t = 0.8 \log(4t)$ which has been used in past research.

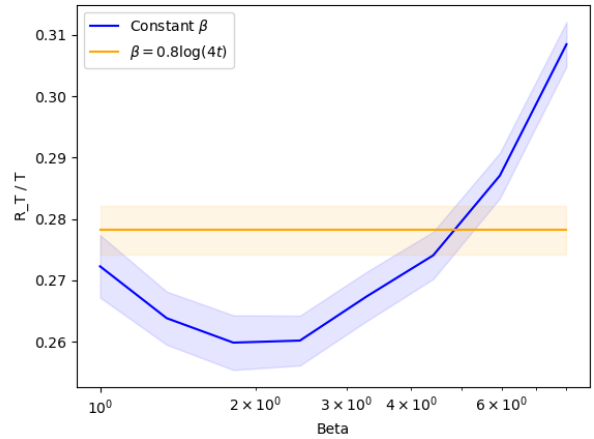
Finally, we perform a more detailed test to determine the impact of ε on β. In this case we choose a low observation noise ($\sigma = 0.01$) to reduce the variance of the final estimate. In previous tests we sampled β in the range $[0.2, 20]$ and observed that in most cases the optimal value was attained in the range $[2.0, 4.0]$. For this test we instead sample 8 values of β in $[1.0, 8.0]$ in order to get a more detailed resolution of the average regret around the optimal point. We also increase the total number of timesteps of the simulation to 300 to obtain a better approximation of how the algorithm performs for long time horizons. Performing long simulations is computationally costly as the compute time of the GPR scales according to $\mathcal{O}(t^3)$ (discussed in more detail in chapter 7). However, for the case of $\varepsilon = 0.003$ we increase the time horizon of the simulation to $T = 600$ since the effects of the time-varying function with such a low rate of change can only be seen on long time horizons.

In figure 5.4 we present the results of this test. The results indicate that the optimal value of β does depend on ε and there is an inverse correlation. As we increase ε the optimal value of β decreases. This is in line with the results of theorem 4.1 and the discussion in section 4.6.

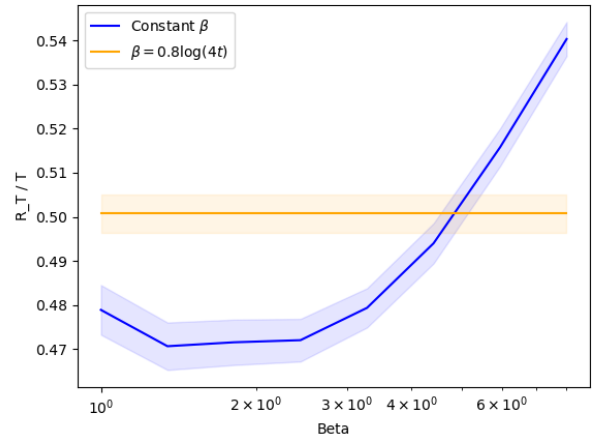


(a) [$\varepsilon = 0.003$ and $T = 600$]

(b) [$\varepsilon = 0.01$ and $T = 300$]



(c) [$\varepsilon = 0.03$ and $T = 300$]



(d) [$\varepsilon = 0.09$ and $T = 300$]

Figure 5.4: Average regret vs β. We compare constant values of β (shown on the horizontal axis with log scale) to $\beta_t = 0.8 \log(4t)$ including the 95% confidence interval of the mean after 400 trials. [$\sigma = 0.01$]

Alternative Temporal Covariance

In our analysis so far we have assumed that the function f_t changes over time according to equation (3.1). In the existing literature there are various types of temporal covariances considered for Gaussian process optimization. In [2] it is assumed that the temporal covariance decays exponentially with time. In [3] the case of change-points is considered where the underlying function changes instantaneously at some timestep and remains static at the remaining timesteps. For real world applications it is useful to consider alternative temporal covariance structures. In chapter 6 we introduce deep reinforcement learning (DRL) which forms some of the motivation for developing the new temporal models which are introduced in the following chapters.

It is important to note that in practice we often do not know the true kernel parameters of the function generating process. When this knowledge is limited we must estimate these parameters from the observations of the function $\{(x_1, y_1), ..., (x_t, y_t)\}$. This is discussed in more detail in chapter 7. One of the implications of fitting the kernel parameters to the data is that a covariance kernel with many parameters quickly becomes intractable which we will show to be relevant in chapter 10. In other words, it is not feasible to use a temporal covariance kernel which is too flexible. With this in mind we introduce and test two new temporal covariance structures which we expect to be relevant for real world applications.

First, in chapter 8 we consider the case where the perturbations which influence the function f_t at every timestep also have a decaying effect on the changes of the function at future timesteps. This causes the variations of the function f_t to be positively correlated with the changes of the function in previous timesteps $\{f_{t-1}, f_{t-2} \ldots\}$. This can be seen as a form of momentum; we expect the function to continue to increase at points where it has increased in recent timesteps. We introduce a new algorithm to deal with this type of temporal dependence called Momentum Time Varying Gaussian Process Upper Confidence Bound (MTV-GP-UCB). Returning to the bird scientist example from the introduction this could occur if heavy rainfall causes worms to emerge from the ground which increases the bird density around a clearing in the forest. This will not only affect the bird behaviour on the following day but could have a continued effect on the changes in multiple following days.

In chapter 9 we examine the case where the function transitions from one static state to another (similarly to [3]). However, rather than assuming this transition happens instantly we consider the case where this change occurs over a larger time frame. For example we could have a function which is equal to h_1 at $t = 0$ and equal to h_2 at $t = 100$. In between $t = 0$ and $t = 100$ the function is some combination of h_1 and h_2. To tackle this problem we introduce an algorithm called Transition Time Varying Gaussian Process Upper Confidence Bound (TTV-GP-UCB). If we again return to the bird example in the introduction this would be the case if there was a new predator introduced into the ecosystem. This causes the birds the adapt to this change in their environment, for example they might learn to avoid areas with lower trees. This can be seen as a transition from h_1 to h_2.

6 Online Hyperparameter Optimization

We will now describe some applications with real data where the data generating model is completely unknown. Consider a hypothetical deep reinforcement learning (DRL) model which is learning (within a simulation) to control a robot body with the goal of learning how to run. Initially the model learns to how to stand up and avoid falling over. This transitions into the model learning how to walk which then finally transitions into the model learning to run. This transitions are not necessarily abrupt; the model can already start learning how to walk while it is still learning how to stand in a stable manner. It is quite likely that each of these three tasks are learned most efficiently with different training hyperparameters (learning rate, batch size etc.). In this sense one can imagine that the underlying function (with hyperparameters as domain) is initially quite constant, as the model is still learning how to balance, but once the transition to walking starts to happen the function changes significantly.

The TVBO model has been successfully applied the problem of online hyperparameter optimization of DRL models [20]. It was shown to outperform other methods such as BO for a static function and genetic algorithms for this task. This implies that the optimal hyperparameters are indeed changing throughout the learning process and can be modelled by a Gaussian process. We expect that the the new models; MTV-GP-UCB and TTV-GP-UCB are well suited for this task and will outperform TV-GP-UCB in more complex environments. In this chapter we provide some motivation for this belief.

6.1. Deep Reinforcement Learning

We will give a brief introduction into DRL which will serve as background knowledge for the following sections. DRL algorithms are deployed in order to solve complex Markov decision problems. This is a problem in which an agent observes a state $S_t \in \mathcal{S} \subset \mathbb{R}^d$ and must choose some action (represented by a vector) $a_t \in \mathcal{A} \subset \mathbb{R}^m$. Following this the agent receives a reward g_{t+1} and observes the following state S_{t+1}. Again, the agent must choose $a_{t+1} \in \mathcal{A}$ and so forth. A flowchart which illustrates this is show in figure 6.1.

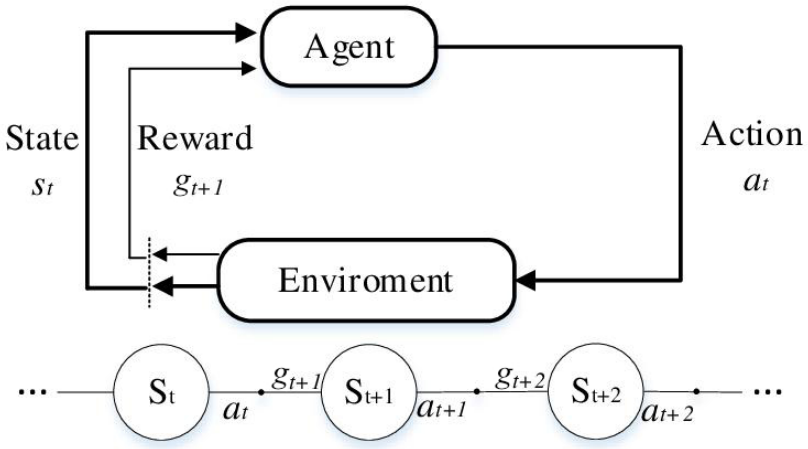


Figure 6.1: Flowchart of simple DRL training environment

The goal of the agent is to choose $a_t \in \mathcal{A}$ as to maximize the expected discounted future rewards:

$$\mathbb{E}\left[G_t\right] = \mathbb{E}\left[g_t + \gamma g_{t+1} + \gamma^2 g_{t+1} + \ldots\right] = \mathbb{E}\left[\sum_{i=0}^{n} \gamma^i g_{t+i}\right]$$

where the discount factor $\gamma \in (0, 1]$ ensures that the agent prefers rewards sooner rather than later which often helps with stability of the model. Note that depending on the environment; n can be finite or $\mathbb{E}\left[G_t\right]$ could involve an infinite sum ($n = \infty$). Some environments have a maximum number of timesteps. Alternatively, there could be a condition such as; the environment stops if some terminal state is reached ($S_t \in \mathcal{S}_\mathcal{T} \subset \mathcal{S}$). The action is sampled according to a policy $\pi(a)$:

$$a_t \sim \pi(a|S = S_t)$$

which is a function that maps states S_t to a specific action $a \in \mathcal{A}$ or a probability distribution over actions. The "deep" part of DRL refers to the fact that these algorithms use deep neural networks (NNs) to approximate this policy π.

Note that there are some similarities between this topic and the online learning we have covered so far in this thesis. In both cases we are optimizing some function (which is unknown to us). We have limited information and must choose between exploring the domain or exploiting the function where we have already gained knowledge. The key difference between DRL and online learning problems (such as Bayesian optimization) arises from the fact that in these Markov decision problems the chosen action $a_t \in \mathcal{A}$ will influence future states

$S_{t+1}, S_{t+2}, \ldots$. This is fundamentally different from the assumptions we have made so far, we have always assumed our choice of $x_t \in D$ did not impact the function f_t.

The removal of this assumption greatly increases the number of tasks which can be solved by these models. DRL has been applied to many tasks; from solving board games and robotic control problems [19] to autonomous vehicles [29] and even controlling the magnetic fields inside fusion reactors; ensuring the stability of the plasma [26]. However, the learning process for these tasks is significantly more complex than online learning scenarios. It is hard to achieve any useful theoretical guarantees for these complex problems.

Training DRL agents typically takes place in a simulated environment where the neural network (representing $\pi(a|S = S_t)$) is initialized with random weights. These NN weights are updated (using stochastic gradient descent/backpropagation [12]) as the agent interacts with the environment and learns which actions (in which states) result in positive and negative rewards. There is a large body of research [27] on how to optimally train neural networks for these problems. DRL algorithms control this training procedure and how the NN weights are updated. However, most state of the art DRL algorithms have a large number of hyperparamters which influence the rate at which the neural network is able to converge to an effective policy. Additionally, from experiments we observe that the optimal hyperparameters can be vastly different depending on the environment [7]. Since for most environments we do not know beforehand which hyperparameter schedule will result in fast learning this motivates the use of hyperparameter optimization algorithms such as TV-GP-UCB to adjust these parameters during the training process (based on the rate at which the model is learning).

As an example we consider the popular Proximal Policy Optimization (PPO) [25] algorithm. During training this algorithm uses a stochastic policy, which means that actions are drawn from a probability distribution. The probability distribution is represented by the neural network. The goal in training is to increase the probability of taking "good" actions. Without delving into the details of the PPO implementation we can discuss three key hyperparameters [22] which we expect to change over the course of the training session:

- Learning rate: this parameter controls the step-size of the updates during the stochastic gradient descent which updates the NN weights. A larger learning rate will increase the rate at which the policy changes. However, a learning rate which is too large could also cause the policy to "over-correct".
- Batch size: this hyperparameter determines the number of interactions with the environment are included in each update of the NN weights. A lower batch size allows for more frequent updates but might not include a representative sample.
- Entropy coefficient: the entropy coefficient is used to steer the NN towards more exploration. It is a term which is added to the computed loss that encourages the

gradient descent step to adjust weights towards a more "random" policy. If the entropy coefficient is too low it is possible that the agents becomes stuck in a local optimum and doesn't sufficiently explore the environment.

6.2. Hyperparameter Changes

As an example we will consider the Gymnasium Mujoco Humanoid environment [30]. This is a physics simulation in which the agent is tasked with controlling the body of a 3D bipedal robot. With the ultimate goal of the environment; to walk forward as fast as possible without falling over. The NN receives relevant state information as input (such as the position and velocities of the various parts of the body). The policy can choose to activate "muscles" which allow it to control the movement of the robot. A visualization of this environment is presented in figure 6.2.

Figure 6.2: Render of the Gymnasium Mujoco Humanoid environment

The agent receives a positive at each timestep reward if the robot is able to keep it's torso above 1m. An additional reward is provided proportional to the forward velocity of the robot. If the robots torso falls below 1m the environment is reset (into an initial standing position). Let's consider training an agent to learn in this environment using the PPO algorithm (while adjusting the hyperparameters according to figure 6.3, using the TTV model). The NN is initialized with random weights and we selecting random hyperparameters as a starting point. This causes the agent to take random actions which quickly result in the environment being reset (due to the torso passing below 1m) resulting in a negative reward. Over time the GPR model learns which region in the hyperparameter domain is most effective. The agent learns to balance the body in order to avoid the environment resetting and continue collecting the positive rewards. At this point there is a gradual transition as most of the additional reward can be gained by increasing the forward velocity of the robot. This can be seen as a different task and likely has different optimal hyperparameters. For example the learning rate might be reduced to avoid the policy changing too far from the current policy which is able to

balance the robot. The TTV-GP-UCB is designed to handle this type of transition in the underlying function. As the agent starts to learn to walk (while staying balanced) the TTV model adapts the hyperparameters to this new task. Finally, once the NN has converged to a policy which is able to stay balanced and walk a final transition may take place. In this case the agent is transitioning from walking to running. Again, this likely results in a change in the hyperparameters which is captured by the TTV model. For example it might be required to reduce the Entropy coefficient so that the policy becomes less random (thereby allowing to agent to take more risks and walk faster). Note that the temporal model derived in chapter 9 assumed only one single transition will occur in the dataset. However, this could be adapted to handle multiple transitions, either by discarding old data or by adding additional parameters to model additional transitions. In theory, using TTV-GP-UCB to dynamically adapt the choice of hyperparameters allows for efficient learning which reduces the training time.

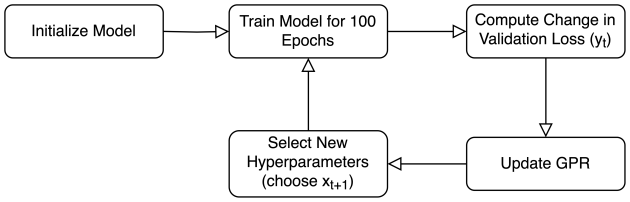


Figure 6.3: Simplified training loop TV-GP-UCB

Alternatively, let's consider the environment where an agent is learning to play the stochastic casino game Blackjack. Again, we initially have a random policy due to the random initialization of the NN. At the beginning of the training process a small batch size and high learning rate might be optimal to rapidly update the policy (and move it roughly in the correct direction). However, throughout the training and as we get closer to the optimal policy we must increasing our batch size. The stochastic nature of the environment can cause the samples to not be representative of the expected reward. In some steps the agent might take optimal actions but still lose the game. It is possible that this is a consistent trend over time (the optimal batch size simply increases over time throughout the entire training process). As we get closer to the final "perfect" policy we continue to increase our batch size. This type of evolution of the optimal hyperparameters can be handled by the MTV-GP-UCB algorithm which is able to capture trends time varying function f_t.

7 Implementation

Before we can introduce the two new models we must discuss some of the technicalities related to the implementation of these algorithms. The Gaussian process regression was implemented using the python library GPy [11] and the functions were sampled from a Gaussian process using the scikit-learn python library [21].

7.1. Kernel Parameters

So far we have assumed that the parameters (k, ε, σ) of the GP prior were known to the algorithm. This has allowed us to derive regret bounds and simulate the performance of the algorithm for various values of β. However, in many real world applications the user may not have exact knowledge of some or all of these parameters. When we are evaluating the performance of these algorithms for real applications we must consider how this lack of knowledge will affect their performance.

Since we must determine the parameters in order to apply the Gaussian process regression we must estimate them based on the available data. A commonly used technique to overcome this is to "fit" these parameters to the data by maximizing the log marginal likelihood (LML) of observing the data [23]:

$$\log(p(y \mid X)) = -\frac{1}{2} y^T (\tilde{\mathbf{K}}_t + \sigma^2 \mathbf{I})^{-1} y - \frac{1}{2} \log \left| \tilde{\mathbf{K}}_t + \sigma^2 \mathbf{I} \right| - \frac{n}{2} \log 2\pi \tag{7.1}$$

where $\tilde{\mathbf{K}}_t$ is the covariance matrix as defined in equation (3.2) for the case of the time-varying function with random perturbations. This is not guaranteed to be a convex optimization problem which introduces additional computational cost. Additionally, it has been empirically observed [14] that having good priors for the parameters is essential to making GPs scale to higher dimensional data.

We will be testing our algorithms on synthetic data and compare the performance with perfect knowledge of the true parameters and fitting the parameters based on the observed data. In order to efficiently maximize (7.1) it is convenient to compute the derivative of log marginal likelihood with respect to the parameters. Therefore, as we introduce the new

temporal models we will also aim to explicitly formulate these derivatives which can directly be implemented in the GPy model in python.

7.2. Computation

One of the major drawbacks of Gaussian process regression is that in general the computational cost scales according to $\mathcal{O}(t^3)$ [23]. This arises from the computationally expensive inversion $(K + \sigma^2 I)^{-1}$, which occurs both during the prediction of $\mu(x)$ and $\sigma^2(x)$, as well as during the learning of the kernel parameters using (7.1). Recall that the mean and variance estimates are computed as:

$$\mu_t(x) = \tilde{\mathbf{k}}_t(x)^T(\tilde{\mathbf{K}}_t + \sigma^2\mathbf{I})^{-1}y_t \tag{7.2}$$

$$\sigma_t^2(x) = \tilde{k}(x,x) - \tilde{\mathbf{k}}_t(x)^T(\tilde{\mathbf{K}}_t + \sigma^2\mathbf{I})^{-1}\tilde{\mathbf{k}}_t(x) \tag{7.3}$$

where $\tilde{\mathbf{k}}_t$ and $\tilde{\mathbf{K}}_t$ were defined in (3.3) and (3.2) respectively. In chapter 11 we will discuss how these algorithms can be improved to achieve better scaling than $\mathcal{O}(t^3)$ for the case of a time-varying function.

This poor scaling limits the extent to which we can tests our algorithms on long time horizons. Additionally, we must choose the amount of compute time used to optimize (7.1). As the optimization problem is non-convex it is possible that we will not attain the global optimal kernel parameters according to the LML. We adjust our algorithm to Algorithm 3 when the true parameters are unknown, this version includes an additional loop which optimizes the kernel parameters. Here, the $n\_restarts$ determines the computational effort put into finding a global optimal set of parameters for the LML.

This Algorithm 3 will be used to test the performance in the case of unknown kernel parameters.

Algorithm 3 TV-GP-UCB algorithm (with kernel parameter estimation)

Require: Domain D, lower and upper bounds for the parameters (k, ε, σ)

for $t = 1, 2, \ldots, T$ **do**
  Choose $x_t = \arg\max_{x \in D} \left[\mu_{t-1}(x) + \beta_t^{1/2} \sigma_{t-1}(x) \right]$
  Sample $y_t = f(x_t) + z_t$
  for $i = 1, 2, \ldots, n\_restarts$ **do**
    if $i = 1$ **then**
      Use the optimal parameters from the previous t as the starting (k, ε, σ)
    else
      Initialize (k, ε, σ) randomly within the given bounds
    end if
    Maximize (7.1) by performing gradient descent steps
  end for
  Select the (k, ε, σ) which achieved the maximum LML in the previous loop
  Perform Bayesian update according to (7.2) and (7.3)
end for

8 Momentum Time-Varying Function

We now aim to derive a model which is able to capture a slightly more complex temporal covariance structure. Suppose that instead of the perturbations g_t having a single instantaneous effect on f_t, they instead also cause (decaying) changes to f at future timesteps $f_{t+1}, ...$ This can be seen as a sort of momentum where previous changes in the function are expected to persist in the future. In order to make this model useful for inference we would like to be able to efficiently fit the model to the data by maximizing the log marginal likelihood (LML) and determining the values of the hyper-parameters of the kernels. Ideally, we would like to have a model which has an explicit function for the covariance kernel and an explicit derivative of the LML w.r.t. these kernel parameters.

8.1. Momentum Time-Varying Gaussian Process

We introduce the following model which is a generalization of the time varying model presented in chapter 3. We call this the momentum time varying Gaussian process model (MTV-GP):

$$f_{t+1}(x) = \varepsilon f_t(x) + \sqrt{\lambda} \sum_{i=0}^{\infty} \alpha^i g_{t+1-i}(x) \tag{8.1}$$

where $\varepsilon \in [0, 1)$ controls the rate of change of the function and $\alpha \in [0, \varepsilon]$ controls the rate of decay of the momentum. The functions g_i are independently sampled from a Gaussian process $g_i \sim \mathcal{GP}(0, k)$. The parameter λ is chosen such that for all t we have the prior $f_t \sim \mathcal{GP}(0, k)$. The correct choice of λ is proven in lemma 8.1. It is interesting to note that this model is a generalization of the time varying model presented in [2]. If we set $\alpha = 0$ the momentum effect is completely removed and the model returns to the simple TV-GP.

By adding an additional parameter to the temporal kernel the model is able to capture more complex dynamics in the time varying function. To give some intuition for this we again compare with the simple TV-GP model. In that model it is assumed that at each time step the underlying function f_t is perturbed by combining it with an independent sample from

a $\mathcal{GP}(0, k)$. This implies that the underlying function changes in a random fashion at each time step. If the change of the function also includes the decaying momentum effect from past perturbations there is a different temporal covariance structure. The MTV-GP model tackles this case by ensuring that the random samples from the $\mathcal{GP}(0, k)$ have a delayed decaying effect on the function. We observe that at some time t the function is perturbed by $\alpha^0 g_t + \alpha^1 g_{t-1} + \alpha^2 g_{t-2} + ...$ a geometric series of past samples from $\mathcal{GP}(0, k)$. This can be interpreted as a form of momentum as each perturbation g_t also has a decaying effect on the future changes of the function f_t.

We want that for all t; $f_t \sim \mathcal{GP}(0, k)$. Thus we must ensure that the variance does not diverge over time. This can be achieved by setting λ correctly. The value of λ is derived in the following lemma.

Lemma 8.1 If a time varying function is generated according to (8.1) with $\varepsilon \in [0, 1]$, $\alpha \in [0, \varepsilon]$ and $k(x, x) = 1$ and we choose:

$$\lambda = \frac{(\varepsilon - \alpha)^2}{\frac{\alpha^2}{1-\alpha^2} + \frac{\varepsilon^2}{1-\varepsilon^2} - 2\frac{\alpha\varepsilon}{1-\alpha\varepsilon}} \qquad \text{if} \quad \alpha < \varepsilon$$

and

$$\lambda = \frac{(1-\varepsilon^2)^3}{\varepsilon^2 + 1} \qquad \text{if} \quad \alpha = \varepsilon$$

then we will have a $\mathcal{GP}(0, k)$ prior for f_t which results in:

$$\text{Var}(f_t(x)) = 1 \qquad \forall x \in D \quad , \quad \forall t \in \mathbb{R}$$

Proof. In the current formulation of (8.1) the term εf_t contains previous terms g_i and hence is not independent of the second term. We will rewrite f_t as a sum of independent g_i terms which will allow us to compute the variance. Let t be arbitrary and fix some $x \in D$, then:

$$\begin{aligned} f_{t+1}(x) &= \varepsilon f_t(x) + \sqrt{\lambda} \sum_{i=0}^{\infty} \alpha^i g_{t+1-i}(x) \\ &= \varepsilon^2 f_{t-1}(x) + \varepsilon\sqrt{\lambda} \sum_{i=0}^{\infty} \alpha^i g_{t-i}(x) + \sqrt{\lambda} \sum_{i=0}^{\infty} \alpha^i g_{t+1-i}(x) \\ &= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \sum_{i=0}^{j} \alpha^i \varepsilon^{j-i}. \end{aligned} \tag{8.2}$$

The final step follows from recursively unpacking $f_t, f_{t-1}, ...$ It should be noted that due to the nested sum g_{t+1} occurs one time, g_t occurs twice, etc. This is a result of the fact that the perturbation g_i is included in the computation of f_t at every timestep $t \geq i$. We will first consider the case of $\alpha < \varepsilon$, we can now rewrite this sum as:

$$\begin{aligned}
f_{t+1}(x) &= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \sum_{i=0}^{j} \alpha^i \varepsilon^{j-i} \\
&= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \varepsilon^j \sum_{i=0}^{j} \left(\frac{\alpha}{\varepsilon}\right)^i \\
&= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \varepsilon^j \left(\frac{1 - (\frac{\alpha}{\varepsilon})^{j+1}}{1 - \frac{\alpha}{\varepsilon}}\right) \\
&= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \left(\frac{\varepsilon^{j+1} - \alpha^{j+1}}{\varepsilon - \alpha}\right) \\
&= \frac{\sqrt{\lambda}}{\varepsilon - \alpha} \sum_{j=0}^{\infty} g_{t+1-j}(x) \left(\varepsilon^{j+1} - \alpha^{j+1}\right)
\end{aligned}$$

Recall that each $g_i(x) \sim \mathcal{N}(0, 1)$ and every g_i is independent. We can now compute the variance of a sum of independent random variables (which is equal to the sum of the variances):

$$\begin{aligned}
\text{Var}(f_t(x)) &= \text{Var}\left(\frac{\sqrt{\lambda}}{\varepsilon - \alpha} \sum_{j=0}^{\infty} g_{t+1-j}(x) \left(\varepsilon^{j+1} - \alpha^{j+1}\right)\right) \\
&= \frac{\lambda}{(\varepsilon - \alpha)^2} \sum_{j=0}^{\infty} \text{Var}(g_{t+1-j}(x)) \left(\varepsilon^{j+1} - \alpha^{j+1}\right)^2 \\
&= \frac{\lambda}{(\varepsilon - \alpha)^2} \left[\sum_{j=0}^{\infty} \varepsilon^{2j+2} + \alpha^{2j+2} - 2\varepsilon^{j+1}\alpha^{j+1}\right] \\
&= \frac{\lambda}{(\varepsilon - \alpha)^2} \left[\frac{\alpha^2}{1 - \alpha^2} + \frac{\varepsilon^2}{1 - \varepsilon^2} - 2\frac{\alpha\varepsilon}{1 - \alpha\varepsilon}\right]
\end{aligned}$$

where the final step follows from the sum of an infinite geometric series. Now it should be clear that by choosing:

$$\lambda = \frac{(\varepsilon - \alpha)^2}{\frac{\alpha^2}{1-\alpha^2} + \frac{\varepsilon^2}{1-\varepsilon^2} - 2\frac{\alpha\varepsilon}{1-\alpha\varepsilon}}$$

we obtain that the variance is equal to 1. Since t was arbitrary and this holds for any $x \in D$ this completes the proof for $\alpha < \varepsilon$.

We will now consider the case of $\alpha = \varepsilon$. We return to our representation of $f_{t+1}(x)$ given in equation (8.2). Note that this now becomes:

$$f_{t+1}(x) = \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \sum_{i=0}^{j} \varepsilon^j \tag{8.3}$$

$$= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \varepsilon^j \sum_{i=0}^{j} 1 \tag{8.4}$$

$$= \sqrt{\lambda} \sum_{j=0}^{\infty} g_{t+1-j}(x) \varepsilon^j (j+1) \tag{8.5}$$

$$= \frac{\sqrt{\lambda}}{\varepsilon} \sum_{j=1}^{\infty} g_{t+2-j}(x) \varepsilon^j j \tag{8.6}$$

where (8.3) follows from equation (8.2) first since $\alpha = \varepsilon$. Equation (8.4) follows because ε^j is independent of i. Equations (8.5) and (8.6) follow from evaluating the inner sum and re-indexing the outer sum.

Now in order to compute the variance of $f_{t+1}(x)$ we again take the sum of the variances:

$$\begin{aligned}
\mathrm{Var}(f_t(x)) &= \mathrm{Var}\left(\frac{\sqrt{\lambda}}{\varepsilon} \sum_{j=1}^{\infty} g_{t+2-j}(x) \varepsilon^j j \right) \\
&= \frac{\lambda}{\varepsilon^2} \sum_{j=1}^{\infty} \mathrm{Var}(g_{t+2-j}(x)) \varepsilon^{2j} j^2 \\
&= \frac{\lambda}{\varepsilon^2} \sum_{j=1}^{\infty} (\varepsilon^2)^j j^2.
\end{aligned}$$

We must still compute the sum of the form:

$$\begin{aligned}
\sum_{k=1}^{\infty} k^2 r^k &= \frac{1}{1-r} \sum_{k=1}^{\infty} (2k-1) r^k \\
&= \frac{1}{1-r} \left[\frac{2r}{(1-r)^2} - \frac{r}{1-r} \right] \\
&= \frac{(1+r)r}{(1-r)^3}.
\end{aligned}$$

Where the first step follows from [8] and the second step uses the common arithmetico-geometric (Gabriel's staircase). This means that our sum can be evaluated as:

$$\sum_{j=1}^{\infty}(\varepsilon^2)^j j^2 = \frac{\varepsilon^4+\varepsilon^2}{(1-\varepsilon^2)^3}.$$

Which implies that we want to choose λ such that:

$$\frac{\lambda}{\varepsilon^2}\frac{\varepsilon^4+\varepsilon^2}{(1-\varepsilon^2)^3} = 1.$$

This yields:

$$\lambda = \frac{(1-\varepsilon^2)^3}{\varepsilon^2+1}$$

for the case of $\alpha = \varepsilon$ □

8.2. Momentum Time-Varying GP-UCB

We now introduce the algorithm which will be used to optimize a function which is changing in this manner. The algorithm builds on the methods presented in chapter 2 and chapter 3. We use the GP-UCB algorithm with an adjusted covariance kernel in order to handle the time varying function. In order to define this covariance kernel $\tilde{k}(x,x)$ for this new setting we must derive the temporal covariance function.

Lemma 8.2 Consider a time varying function which is generated according to (8.1), with $k(x,x) = 1$. If we fix some $x \in D$ then the covariance of the function at two different times (t and $t+k$) is given by:

$$\mathrm{Cov}(f_t(x), f_{t+k}(x)) = \frac{(\varepsilon^2-1)\,\alpha^{k+1} + (1-\alpha^2)\varepsilon^{k+1}}{(\varepsilon-\alpha)\,(\varepsilon\alpha+1)} \qquad \text{if} \quad \alpha < \varepsilon$$

$$\mathrm{Cov}(f_t(x), f_{t+k}(x)) = \varepsilon^k\left[1 + k\frac{1-\varepsilon^2}{1+\varepsilon^2}\right] \qquad \text{if} \quad \alpha = \varepsilon$$

Proof. Similarly to Lemma 8.1 it is convenient to decompose f_t into a sum of independent g_i terms. From the proof of Lemma 8.1 we have the following equation for f_t:

$$f_t(x) = \frac{\sqrt{\lambda}}{\varepsilon - \alpha} \sum_{j=0}^{\infty} g_{t-j}(x) \left(\varepsilon^{j+1} - \alpha^{j+1}\right).$$

For f_{t+k} we simply shift the subscript of g:

$$f_{t+k}(x) = \frac{\sqrt{\lambda}}{\varepsilon - \alpha} \sum_{j=0}^{\infty} g_{t+k-j}(x) \left(\varepsilon^{j+1} - \alpha^{j+1}\right).$$

We can now start to compute the desired covariance. Note that every g_i is i.i.d. sampled from the Gaussian process. And for a fixed $x \in D$ we have $g_i(x) \sim \mathcal{N}(0,1)$. Then we can use the property of covariance (for any constant a,b):

$$\begin{aligned} \mathrm{Cov}(ag_i(x), bg_j(x)) &= 0 \qquad i \neq j \\ \mathrm{Cov}(ag_i(x), bg_j(x)) &= ab \qquad i = j. \end{aligned}$$

This, combined with the properties of geometric series allows us to explicitly compute the covariance. For the case of $\alpha < \varepsilon$:

$$\begin{aligned} \mathrm{Cov}(f_t(x), f_{t+k}(x)) &= \frac{\lambda}{(\varepsilon-\alpha)^2} \mathrm{Cov}\left(\sum_{j=0}^{\infty} g_{t-j}(x)\left(\varepsilon^{j+1} - \alpha^{j+1}\right), \sum_{j=0}^{\infty} g_{t+k-j}(x)\left(\varepsilon^{j+1} - \alpha^{j+1}\right)\right) \\ &= \frac{\lambda}{(\varepsilon-\alpha)^2} \sum_{j=0}^{\infty} \left(\varepsilon^{j+1} - \alpha^{j+1}\right)\left(\varepsilon^{j+k+1} - \alpha^{j+k+1}\right) \\ &= \frac{\lambda}{(\varepsilon-\alpha)^2} \sum_{j=0}^{\infty} \left(\varepsilon^{2j+k+2} + \alpha^{2j+k+2} - (\varepsilon^k + \alpha^k)(\alpha\varepsilon)^{j+1}\right) \\ &= \frac{\lambda}{(\varepsilon-\alpha)^2} \left[\frac{\varepsilon^{k+2}}{1-\varepsilon^2} + \frac{\alpha^{k+2}}{1-\alpha^2} - \frac{(\varepsilon^k + \alpha^k)(\alpha\varepsilon)}{1-\alpha\varepsilon}\right] \end{aligned}$$

Now replacing the λ we defined in Lemma 8.1:

$$\begin{aligned} \mathrm{Cov}(f_t(x), f_{t+k}(x)) &= \frac{\lambda}{(\varepsilon-\alpha)^2} \left[\frac{\varepsilon^{k+2}}{1-\varepsilon^2} + \frac{\alpha^{k+2}}{1-\alpha^2} - \frac{(\varepsilon^k + \alpha^k)(\alpha\varepsilon)}{1-\alpha\varepsilon}\right] \\ &= \frac{\frac{\varepsilon^{k+2}}{1-\varepsilon^2} + \frac{\alpha^{k+2}}{1-\alpha^2} - \frac{(\varepsilon^k+\alpha^k)(\alpha\varepsilon)}{1-\alpha\varepsilon}}{\frac{\alpha^2}{1-\alpha^2} + \frac{\varepsilon^2}{1-\varepsilon^2} - 2\frac{\alpha\varepsilon}{1-\alpha\varepsilon}} \\ &= \frac{\varepsilon^{k+2}(1-\alpha^2)(1-\alpha\varepsilon) + \alpha^{k+2}(1-\varepsilon^2)(1-\alpha\varepsilon) - (\varepsilon^k+\alpha^k)(\alpha\varepsilon)(1-\alpha^2)(1-\varepsilon^2)}{\varepsilon^2(1-\alpha^2)(1-\alpha\varepsilon) + \alpha^2(1-\varepsilon^2)(1-\alpha\varepsilon) - 2(\alpha\varepsilon)(1-\alpha^2)(1-\varepsilon^2)} \\ &= \frac{(\varepsilon-\alpha)\left[\varepsilon^{k+1}(1-\alpha^2) - \alpha^{k+1}(1-\varepsilon^2)\right]}{\varepsilon^2(1-\alpha^2)(1-\alpha\varepsilon) + \alpha^2(1-\varepsilon^2)(1-\alpha\varepsilon) - 2(\alpha\varepsilon)(1-\alpha^2)(1-\varepsilon^2)} \\ &= \frac{(\varepsilon^2-1)\,\alpha^{k+1} + (1-\alpha^2)\varepsilon^{k+1}}{(\varepsilon-\alpha)\,(\varepsilon\alpha+1)}. \end{aligned}$$

This is the desired result and thus completes the proof for $\alpha < \varepsilon$.

Now for the case of $\alpha = \varepsilon$. Based on equation (8.6) we have:

$$\begin{aligned}
\mathrm{Cov}(f_t(x), f_{t+k}(x)) &= \frac{\lambda}{\varepsilon^2}\mathrm{Cov}\left(\sum_{j=1}^{\infty} g_{t+1-j}(x)\varepsilon^j j, \sum_{j=1}^{\infty} g_{t+1+k-j}(x)\varepsilon^j j\right) \\
&= \frac{\lambda}{\varepsilon^2}\sum_{j=1}^{\infty} \varepsilon^j j \varepsilon^{j+k}(j+k) \\
&= \lambda\varepsilon^{k-2}\sum_{j=1}^{\infty} \varepsilon^{2j} j^2 + \varepsilon^{2j} jk \\
&= \lambda\varepsilon^{k-2}\left[\frac{\varepsilon^4+\varepsilon^2}{(1-\varepsilon^2)^3} + k\frac{\varepsilon^2}{(1-\varepsilon^2)^2}\right] \\
&= \frac{(1-\varepsilon^2)^3}{\varepsilon^2+1}\varepsilon^k\left[\frac{\varepsilon^2+1}{(1-\varepsilon^2)^3} + k\frac{1}{(1-\varepsilon^2)^2}\right] \\
&= \varepsilon^k\left[1 + k\frac{1-\varepsilon^2}{1+\varepsilon^2}\right].
\end{aligned}$$

This is the desired result for $\alpha = \varepsilon$ which completes the proof.

□

We will now restrict to the case of $\alpha < \varepsilon$ and derive the relevant expressions for the implementation of the algorithm. The case of $\alpha = \varepsilon$ can be treated analogously.

The partial derivatives of the covariance w.r.t. α and ε can also be explicitly computed (this is useful when fitting the model using LML maximization):

$$\frac{\partial \mathrm{Cov}(f_t(x), f_{t+k}(x))}{\partial \alpha} = -\frac{\left(\varepsilon^2-1\right)\left(\alpha^k\cdot\left(\left(\varepsilon k-\varepsilon\right)\alpha^2 + \left(1-\varepsilon^2\right)k\alpha - \varepsilon k - \varepsilon\right) + \varepsilon^{k+1}\alpha^2 + \varepsilon^{k+1}\right)}{\left(\alpha-\varepsilon\right)^2\left(\varepsilon\alpha+1\right)^2}. \tag{8.7}$$

Due to symmetry the partial derivative with respect to ε is of the same form and can be obtained by swapping α with ε.

Now we can use this covariance function as the temporal component of our kernel function. Similarly to [2] the kernel is produced by taking the element-wise product of the spatial and temporal covariance kernel as follows:

$$\begin{aligned} \tilde{k}_{MTV}(x,x) &= k(x,x') \cdot d_{MTV}(t,t') \\ \tilde{\mathbf{K}}_t^{MTV} &= \mathbf{K}_t \circ \mathbf{D}_t^{MTV} \\ \tilde{\mathbf{k}}_t^{MTV}(x) &= \mathbf{k}_t(x) \circ \mathbf{d}_t^{MTV} \end{aligned} \tag{8.8}$$

where $\circ$ denotes the Hadamard product and $k(x,x')$ representing the spatial kernels for the Gaussian Process regression (these could for example be the commonly used Squared Exponential kernel or the Matérn kernel). The vector $\mathbf{k}_t(x)$ and matrix $\mathbf{K}_t$ are defined as in (2.4) and (2.5) respectively based on $k(x,x')$. The temporal kernel is defined as:

$$d_{MTV}(t,t') := \frac{(\varepsilon^2-1)\,\alpha^{|t-t'|+1} + (1-\alpha^2)\varepsilon^{|t-t'|+1}}{(\varepsilon-\alpha)\,(\varepsilon\alpha+1)} \tag{8.9}$$

$$\mathbf{D}_t^{MTV} := \left[\frac{(\varepsilon^2-1)\,\alpha^{|i-j|+1} + (1-\alpha^2)\varepsilon^{|i-j|+1}}{(\varepsilon-\alpha)\,(\varepsilon\alpha+1)}\right]_{i,j=1}^{t}$$

$$\mathbf{d}_t^{MTV} := \left[\frac{(\varepsilon^2-1)\,\alpha^{|t+1-i|+1} + (1-\alpha^2)\varepsilon^{|t+1-i|+1}}{(\varepsilon-\alpha)\,(\varepsilon\alpha+1)}\right]_{i=1}^{t}.$$

Note that equation (8.9) follows from lemma 8.2. Recall that the Gaussian process regression is defined as:

$$\mu_t(x) = \tilde{\mathbf{k}}_t^T(\tilde{\mathbf{K}}_t + \sigma^2\mathbf{I})^{-1}y_t \tag{8.10}$$

$$\sigma_t^2(x,x) = \tilde{k}(x,x) - \tilde{\mathbf{k}}_t^T(\tilde{\mathbf{K}}_t + \sigma^2\mathbf{I})^{-1}\tilde{\mathbf{k}}_t \tag{8.11}$$

We can now compute the predicted mean and variance using (8.10) and (8.11) with the new $\tilde{\mathbf{K}}_t^{MTV}$ and $\tilde{\mathbf{k}}$ given in equation (8.9). The algorithm chooses the point to sample according to the UCB rule where, as in the previous algorithms, the exploration-exploitation trade-off is controlled by β_t:

$$x_t = \arg\max_{x\in\mathcal{D}} \mu_{t-1}(x) + \beta_t^{1/2}\sigma_{t-1}(x)$$

For updating the LML in the implementation we would like to compute the partial derivative of the terms of $\mathbf{D}_t^{MTV}$ w.r.t. α and ε. This can be done using equation (8.7) and populating the matrix as follows:

$$\frac{\partial \mathbf{D}_t^{MTV}}{\partial \alpha} = \left[\frac{\partial \text{Cov}(f_t(x), f_{t+|i-j|}(x))}{\partial \alpha}\right]_{i,j=1}^{t}.$$

The partial derivative w.r.t. ε is computed using the same method. We now have all the equations required to implement the UCB algorithm for this setting according to Algorithm 3.

8.3. Regret Bounds

The regret bounds can be derived using the same procedure as the TV-GP model which is described in detail in chapter 4. This yields the following regret bounds:

Theorem 8.1 Let $D \subseteq [0, r]^d$ be compact and convex with $d \in \mathbb{N}$ and $r > 0$. Suppose a time varying function f_t is generated according to:

$$f_{t+1}(x) = \varepsilon f_t(x) + \sqrt{\lambda} \sum_{i=0}^{\infty} \alpha^i g_{t+1-i}(x)$$

where $g_i \sim \mathcal{GP}(0, k)$ and $\varepsilon \in (0, 1]$. Assume that the spatial kernel k generates functions such that there exists some $a, b > 0$ for which it holds that:

$$\mathbb{P}\left\{\sup_{\boldsymbol{x} \in D} \left|\frac{\partial f}{\partial x^{(j)}}\right| > L\right\} \leq a e^{-(L/b)^2}, \quad \forall L \quad j = 1, \ldots, d.$$

At each time step we select one point in the domain x_t to sample the function and observe a noisy evaluation $y_t = f(x_t) + z_t$ with $z_t \sim \mathcal{N}(0, \sigma^2)$. Choose some $\delta \in (0, 1)$ and $\tau \in \mathbb{N}^+$. If we select the point to sample at each timestep according to the UCB rule:

$$\beta = 2(\log(\frac{3}{2\delta}) + d \log(\tau))$$

$$x_t = \arg\max_{x \in D} \left[\mu_{t-1}(x) + \beta^{1/2} \sigma_{t-1}(x)\right].$$

then we will achieve the following bounds on the expected cumulative regret R_T:

$$\mathbb{E}[R_T] \leq (1 - \delta)\left[\sqrt{C_1 T \beta \tilde{\gamma}_T} + T C_2\right] + T \delta C_3.$$

And assuming that conjecture 4.1 holds. We have with probability at least $(1 - \omega)$:

$$R_T \leq \mathbb{E}[R_T] + \sqrt{9\left[T\frac{1+(1-\varepsilon)^{1/2}}{1-(1-\varepsilon)^{1/2}} + \frac{2(1-\varepsilon)^{1/2}}{((1-\varepsilon)^{1/2}-1)^2}\left((1-\varepsilon)^{(T+1)/2}+1\right)\right]\frac{1}{\omega}}$$
$$= \mathcal{O}(T)$$

where $\tilde{\gamma}_T$ is the maximum information gain and:

$$C_1 = \frac{8}{\log(1+\sigma^{-2})}$$
$$C_2 = \frac{rdb\sqrt{\log(\frac{3ad}{\delta})}}{\tau}$$
$$C_3 = rdb\left(\sqrt{\log\left(\frac{ad}{\delta}\right)} + \frac{\sqrt{\pi}}{2\sqrt{\log(\frac{ad}{\delta})}}\right).$$

And in the limit as $T \to \infty$ the average regret will converge in probability to the expected average regret:

$$\frac{R_T}{T} \to \frac{\mathbb{E}[R_T]}{T}.$$

Note that the maximum information gain for this function is not identical to the maximum information gain of the original time varying function equation (3.1). Therefore, we will discuss the maximum information gain for this alternative function in more detail.

8.4. Maximum Information Gain

Theorem 8.2

Consider the setting of theorem 8.1, then the maximum information gain for the time varying function can be bounded as follows:

$$\tilde{\gamma}_T \leq \gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\left(\frac{(1-\alpha^2)\varepsilon}{(\varepsilon-\alpha)(\varepsilon\alpha+1)}(1-\varepsilon) + \frac{(\varepsilon^2-1)\alpha}{(\varepsilon-\alpha)(\varepsilon\alpha+1)}\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)$$

where $\gamma_{\tilde{N}}$ is the maximum information gain for a static function $f \sim \mathcal{GP}(0,k)$ which is sampled $\tilde{N}$ times.

Proof. Similarly to chapter 3, from the chain rule for mutual information and the fact that the noise terms in the observations are independent ([5], Lemma 7.9.2):

$$\tilde{I}(\mathbf{f}_T; \mathbf{y}_T) \leq \sum_{i=1}^{T/\tilde{N}} \tilde{I}(\mathbf{f}_{\tilde{N}}^{(i)}; \mathbf{y}_{\tilde{N}}^{(i)})$$

where (i) refers to the i-th block of $\mathbf{y}_{\tilde{N}}$ and $\mathbf{f}_{\tilde{N}}^{(i)}$. This leads to the bound:

$$\tilde{\gamma}_T \leq \frac{T}{\tilde{N}} \tilde{\gamma}_{\tilde{N}}.$$

So it remains to bound $\tilde{\gamma}_{\tilde{N}}$. Recall that we have the covariance matrix derived previously in equation (9.3) as $\tilde{\mathbf{K}}_t$. We rewrite the covariance matrix for each block as as sum of two matrices:

$$\tilde{\mathbf{K}}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} \circ \mathbf{D}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}}$$

where we define $\mathbf{A}_{\tilde{N}}$ as:

$$\mathbf{A}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} \circ \mathbf{D}_{\tilde{N}} - \mathbf{K}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} \circ (\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})$$

and $\mathbf{1}_{\tilde{N}}$ is the $\tilde{N} \times \tilde{N}$ matrix of ones. Now we want to create an upper bound for the absolute value of the terms in the matrix $(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})$. Without loss of generality (due to symmetry) we consider the case of $\varepsilon \geq \alpha$. We define the following variables to simplify notation:

$$\omega := \frac{(1-\alpha^2)\varepsilon}{(\varepsilon - \alpha)(\varepsilon\alpha + 1)} \qquad \theta := \frac{(\varepsilon^2 - 1)\alpha}{(\varepsilon - \alpha)(\varepsilon\alpha + 1)}$$

Observe that we have $\theta \leq 0$ and $\omega \geq 1$ for all $\varepsilon, \alpha \in [0, 1]$ and $\theta + \omega = 1$. Now we define $|i - j| = k$:

$$|(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})_{i,j}| = 1 - \theta\alpha^k - \omega\varepsilon^k = [\theta - \theta\alpha^k] + [\omega - \omega\varepsilon^k] \tag{8.12}$$

We will now create an upper bound for both terms. $[\omega - \omega\varepsilon^k]$ is concave in k and it is upper bounded by (the functions are equal for $k = 0, 1$):

$$u_1(k) = k\omega(1-\varepsilon) \quad \forall k \in \mathbb{N}^+$$

Now, we upper bound $[\theta - \theta\alpha^k]$ (which is convex because θ is negative) by:

$$u_2(k) = k\theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1} \quad 0 \leq k \leq \tilde{N}-1$$

This now allows us to combine these two upper bounds to bound equation (8.12):

$$|(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})_{i,j}| \leq u_1(|i-j|) + u_2(|i-j|) = k\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right) \tag{8.13}$$

We now will use this to obtain an upper bound on the square Frobenius norm of the matrix $(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})$:

$$\begin{aligned}
\|(\mathbf{D}_{\tilde{N}} - \mathbf{1}_{\tilde{N}})\|_F^2 &\leq \sum_{i,j}(i-j)^2\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)^2 \\
&\leq \frac{1}{6}\tilde{N}^2(\tilde{N}-1)^2\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)^2 \\
&\leq \tilde{N}^4\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)^2
\end{aligned}$$

where the first inequality follows from equation (8.13). The second inequality is a common double summation and the final inequality is simply a removal of the constant factor to simplify notation. We define the following variable:

$$\Delta_i := \lambda_i(\tilde{\mathbf{K}}_{\tilde{N}}) - \lambda_i(\mathbf{K}_{\tilde{N}})$$

where λ_i is the i-th largest eigenvalue. We now use Mirsky's theorem (which has been defined earlier: lemma 3.1) to analyze the eigenvalues of $\tilde{\mathbf{K}}_{\tilde{N}}$. We apply this lemma to $\mathbf{U}_{\tilde{N}} = \mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}}$ and $\mathbf{V}_{\tilde{N}} = \mathbf{K}_{\tilde{N}}$. We choose the Frobenius norm as our norm which gives us; $\lambda_i(\mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}}) = \lambda_i(\mathbf{K}_{\tilde{N}}) + \Delta_i$ for some $\{\Delta_i\}_{i=1}^{\tilde{N}}$ which must satisfy:

$$\sum_{i=1}^{\tilde{N}}\Delta_i^2 \leq \tilde{N}^4\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)^2 \tag{8.14}$$

This now allows us to derive an upper bound on $\tilde{\gamma}_{\tilde{N}}$ (according to the definitions given in [28]):

$$\begin{aligned}
\tilde{\gamma}_{\tilde{N}} &= \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\lambda_i(\mathbf{K}_{\tilde{N}} + \mathbf{A}_{\tilde{N}})\right) \\
&= \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\lambda_i(\mathbf{K}_{\tilde{N}}) + \sigma^{-2}\Delta_i\right) \\
&\leq \gamma_{\tilde{N}} + \sum_{i=1}^{\tilde{N}} \log\left(1 + \sigma^{-2}\Delta_i\right) && (8.15) \\
&\leq \gamma_{\tilde{N}} + \tilde{N}\log\left(1 + \sigma^{-2}\tilde{N}^{3/2}\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)\right) && (8.16) \\
&\leq \gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right) && (8.17)
\end{aligned}$$

(8.15) follows from $\log(1+a+b) \leq \log(1+a) + \log(1+b$ for $a, b \geq 0$ and (8.16) follows from considering the worst case scenario with the knowledge that equation (8.14) must hold. This yields $\Delta_i = \tilde{N}^{3/2}\left(\omega(1-\varepsilon) + \theta\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)$ for every i. Finally, (8.17) follows from $\log(1+a) \leq a$. This leaves us with an upper bound for $\tilde{\gamma}_{\tilde{N}}$, upon replacing the defined ω and θ we obtain the desired result:

$$\tilde{\gamma}_T \leq \gamma_{\tilde{N}} + \sigma^{-2}\tilde{N}^{5/2}\left(\frac{(1-\alpha^2)\varepsilon}{(\varepsilon-\alpha)(\varepsilon\alpha+1)}(1-\varepsilon) + \frac{(\varepsilon^2-1)\alpha}{(\varepsilon-\alpha)(\varepsilon\alpha+1)}\frac{(1-\alpha^{\tilde{N}-1})}{\tilde{N}-1}\right)$$

□

9 Transition Time-Varying Function

In this chapter we introduce a new type of time varying function which we expect will be present in real world applications. We would like to consider the case of a function changing in a more predictable way than what has been discussed so far. Specifically, the case where a function is transitioning from some function h_1 at $t = t_1$ to h_2 at $t = t_2$ (with unknown h_1 and h_2). At the times in between $t \in (t_1, t_2)$ the function is some combination of h_1 and h_2. This can be viewed as a setting with less uncertainty compared to the time varying cases where a new sample from a GP is introduced at every timestep. Since there are only two unknown functions (h_1 and h_2) this case could be considered to be somewhere in between the static function [28] and time-varying function [2] case.

For a real world example of when we would want to optimize a function changing like this (and limit regret) consider the bird scientist example from the introduction who is observing birds in a forest. Every day she chooses a location to observe. Her goal is to observe as many birds as possible every day. If we assume the density of birds throughout the forest can initially be modeled as a static function by a Gaussian process (call it h_1), then she could use the procedure mentioned in chapter 2 to inform her choice of which location to observe each day. However, suppose that a new predator starts hunting in the forest. This causes a gradual change in the birds behavior (for example; they start to avoiding areas with low trees in order to avoid risks). Eventually, the ecosystem settles to a new equilibrium with a new density function (h_2). We would like create a model which can identify and capture this type of change while limiting the regret throughout the transition period.

9.1. Transition Time-Varying Gaussian Process

We will analyse the case where during the transition from h_1 to h_2 the function f_t is a weighted average of the two functions:

$$f_t(x) = (1 - s(t))h_1(x) + s(t)h_2(x) \tag{9.1}$$

where $0 \leq s(t) \leq 1$ is a function which controls the rate of change of f_t and $h_1, h_2 \sim \mathcal{GP}(0, k)$.

We expect $s(t)$ to be a monotonically non-decreasing function which means that the transition occurs in one direction with the weight of $h_2(x)$ non-decreasing over time (and vice versa).

It should be noted that in this case our prior on f_t is not $\mathcal{GP}(0, k)$ for all t (as it was for the previous models). This is a result of taking a weighted average of two independent Gaussian processes. Suppose $k(x, x) = 1$ so $\text{Var}(h_1(x)) = \text{Var}(h_2(x)) = 1$ and we are at $s(t) = \frac{1}{2}$. Then for some fixed $x \in D$ the variance of f_t is less than 1:

$$\text{Var}(f_t(x)) = \text{Var}(\frac{1}{2}h_1(x)) + \text{Var}(\frac{1}{2}h_2(x)) = \frac{1}{4}\left(\text{Var}(h_1(x)) + \text{Var}(h_2(x))\right) = \frac{1}{2} < 1.$$

Using this reasoning we can observe that the variance of our prior f_t is minimized at $s(t) = \frac{1}{2}$ which can be considered to be the "middle" of the transition. While this might not seem intuitive at first, it is a result of using an independent Gaussian process prior for h_1 and h_2 which results in $h_1(x)$ and $h_2(x)$ being a normal random variable. The average of two i.i.d. normal random variables will have a lower variance than each individual normal RV as a result of the central limit theorem.

However, using this weighted average method for the transition is useful as it ensures some properties which we expect our function to have. For example suppose that for some $x \in D$ we have that $h_1(x) = h_2(x) = 1$. In this case we would expect $f_t(x) = 1$ for all t as the function is constant at x. Indeed, we see that this is the case when we assume that f_t is generated according to equation (9.1). Consider, what happens if we generate the function such that the variance stays constant throughout the transition. This can be achieved by:

$$v_t(x) = \sqrt{(1 - s(t))}h_1(x) + \sqrt{s(t)}h_2(x).$$

In this case $\text{Var}(v_t(x)) = 1$ for all t. However, if $h_1(x) = h_2(x) = 1$ we observe that $v_t(x) = \sqrt{2}$ at $s(t) = \frac{1}{2}$. This is undesirable as we would expect a constant point to stay constant throughout the transition. Therefore, we proceed with using equation (9.1) as our transition model.

9.2. Transition Time-Varying GP-UCB

We will now derive an algorithm which can optimize this type of time varying function while minimizing the regret. Again, we will work within the UCB framework and create a new covariance kernel which is able to capture this temporal dependence. In the previous section $s(t)$ was an arbitrary function (with certain requirements). For the rest of this chapter (as well as the experiments in the following chapter) we will use the sigmoid function:

$$s(t) = \frac{1}{1 + e^{\frac{\tau - t}{\rho}}}.$$

the parameter $\tau \in \mathbb{R}$ controls the center of the transition (the point where $s(t) = 0.5$), the parameter $\rho > 0$ controls how rapidly the transition occurs (how steep the function is). In figure 9.1 we present a visualisation of this function for the case of $\tau = 10$ and $\rho = 1$. This is a commonly used function for machine learning applications (often used as an activation function for neural networks).

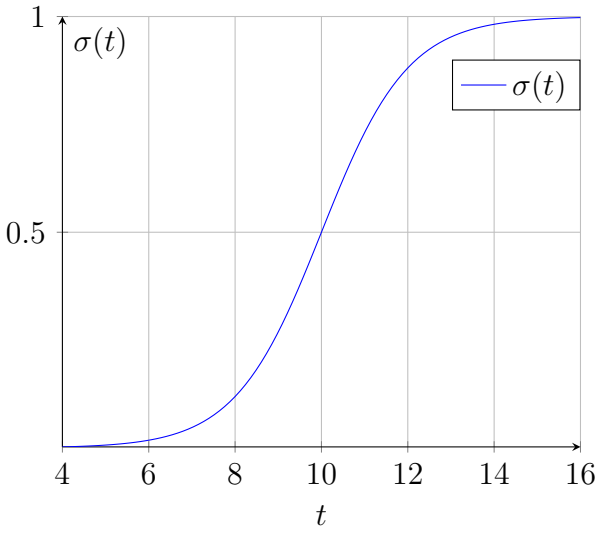


Figure 9.1: Sigmoid function centered at $t = 10$

This function is desirable for this use case as the partial derivatives can be expressed explicitly:

$$\frac{\partial s(t)}{\partial \tau} = -\frac{\exp\left(\frac{t-\tau}{\rho}\right)}{\rho\left(\exp\left(\frac{t-\tau}{\rho}\right) + 1\right)^2} \tag{9.2}$$

$$\frac{\partial s(t)}{\partial \rho} = \frac{(\tau - t)\exp\left(\frac{t-\tau}{\rho}\right)}{\rho^2\left(\exp\left(\frac{t-\tau}{\rho}\right) + 1\right)^2}$$

which allows them to be directly implemented in the python code. Additionally, as the derivative of the function w.r.t. t is non-zero for all finite values of t we expect it to improve the convergence of the LML optimization process.

The following step is to formulate the covariance function for a fixed $x \in D$ as a function of t, t'. We will use the fact that h_1 is independent of h_2. First consider the functions f_t and $f_{t'}$:

$$f_t(x) = (1 - s(t))h_1(x) + s(t)h_2(x) \qquad f_{t'}(x) = (1 - s(t'))h_1(x) + s(t')h_2(x).$$

Now suppose that $k(x, x) = 1$, then:

$$\begin{aligned}\mathrm{Cov}(f_t(x), f_{t'}(x)) &= (1 - s(t))(1 - s(t'))\mathrm{Var}(h_1(x)) + s(t)s(t')\mathrm{Var}(h_2(x)) \\ &= (1 - s(t))(1 - s(t')) + s(t)s(t') \\ &= 2s(t)s(t') - s(t) - s(t') + 1.\end{aligned}$$

We observe for the first time that the temporal covariance is not stationary. This means that the covariance can no longer be computed from the difference in t. For the previous kernels the covariance was only a function of $|t - t'|$ and hence we had $d(t, t') = d(|t - t'|)$. In this transition time-varying setting we see that this is no longer the case. From an implementation perspective this slightly complicates the matter as we can no longer compute the temporal kernel matrix from a matrix of temporal differences.

Our goal is to use the UCB algorithm which requires equation (8.10) and equation (8.11). Similarly to the MTV-GP algorithm we define our new temporal kernel function as:

$$\begin{aligned}\tilde{k}_{TTV}(x, x) &= k(x, x') \cdot d_{TTV}(t, t') \\ \tilde{\mathbf{K}}_t^{TTV} &= \mathbf{K}_t \circ \mathbf{D}_t^{TTV} \\ \tilde{\mathbf{k}}_t^{TTV}(x) &= \mathbf{k}_t(x) \circ \mathbf{d}_t^{TTV}.\end{aligned} \tag{9.3}$$

Based on the covariance function we have that $d(t, t')$ is defined as:

$$d_{TTV}(t, t') := 2s(t)s(t') - s(t) - s(t') + 1.$$

For the implementation we define the two matrices:

$$A = [s(i)]_{i,j=1}^{t} \qquad B = A^T = [s(j)]_{i,j=1}^{t}.$$

Then:

$$\mathbf{D}_t^{TTV} = 1 + 2A \circ B - A - B.$$

And finally the partial derivatives are computed as:

$$\frac{\partial \mathbf{D}_t^{TTV}}{\partial \tau} = 2\frac{\partial A}{\partial \tau} \circ B + 2\frac{\partial B}{\partial \tau} \circ A - \frac{\partial A}{\partial \tau} - \frac{\partial B}{\partial \tau}.$$

Where $\frac{\partial A}{\partial \tau}$ and $\frac{\partial B}{\partial \tau}$ are computed element-wise according to equation (9.2). The derivative w.r.t. ρ is computed in the same manner. This allows us to implement the UCB algorithm (including the LML maximization) according to Algorithm 3.

10 Validation of New Models

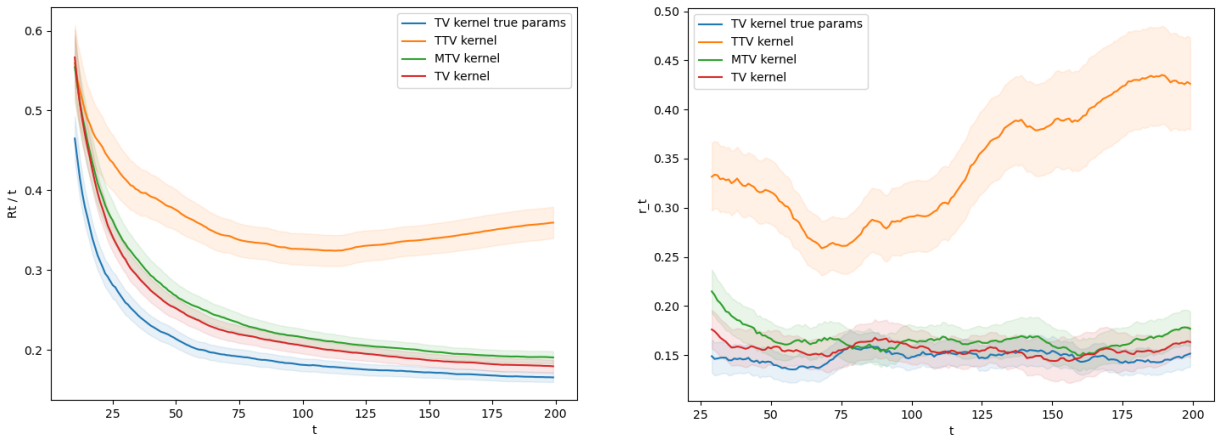
In this chapter we will test the newly developed models to validate their utility. We use a similar setting to chapter 5 where we consider the two dimensional case with a grid of 50 x 50 points. We have $D = [0, 1]^2$ and we generate the data using the spatial square exponential kernel with the length-scale parameter set equal to $\sqrt{0.2}$. We set the noise parameter to $\sigma = 0.1$. In order to produce a fair comparison of how well these models perform in real world settings we have to consider the fact that the new models both have an additional kernel parameter. This could mean that in practice (when the true parameters are unknown) our models under-perform. In order to simulate this we use algorithm 3 which has been described in chapter 7. In this version of the algorithm the kernel parameters are fit using a LML maximization approach along with quasi newton methods. The LML function can be very hard to optimize over which often results in the parameters getting stuck in local optima, hence the starting point is sampled multiple times according to the parameter n_restarts. Based on our testing we found that using algorithm 3 with $n\_restarts = \min(10, 200/t)$ provides a good balance between fitting the kernel parameters to the data and computational efficiency, we will be using this for all tests.

We will be comparing three models; TV-GP-UCB, TV-GP-UCB and TTV-GP-UCB. Each of these models also correspond to methods to generate time-varying functions f_t. For a fair comparison we will generate data according to each method and compare the model performance of each model on each dataset. In each case we will include a perfect model which is given the true parameters and all three models with LML estimation of parameters. The average results (as well as 95% confidence intervals of this average) over 200 trials are presented in the following sections. We include the average cumulative regret R_t/t over time as well a 20 step moving average of the instantaneous regret r_t (the moving average is required to make the graphs readable because r_t is very noisy). The R_t/t graph serves to indicate the overall performance of the algorithm. The r_t graphs gives some indication at which timesteps the algorithms are incurring high regret and when exploration/exploitation is taking place.

10.1. TV Function Data

In figure 10.1 we present the numerical performance of the algorithms on a function generated according to the TV kernel. We observe that the TTV model has quite poor performance in this setting. This is because the model is not able to fit to the dataset. It is assuming the

function is transitioning between two static functions when in fact the function is continuously being perturbed randomly. As to be expected, the best performance is achieved by the TV algorithm with true parameters. This model is given the true parameters at the start and therefore does not have to learn them from the data. The TV model without true parameters also performs well and quite rapidly is able to learn the kernel parameters, it is already achieving similar instantaneous regret r_t as the model with true parameters after 75 timesteps. Finally, the MTV model is a generalization of the TV model. As such, we expect it to perform well in this setting. Indeed, we see that the model obtains only slightly worse performance than the TV model. This is due to the fact that the model is overparameterized for this setting which somewhat limits its ability to learn the correct kernel parameters.



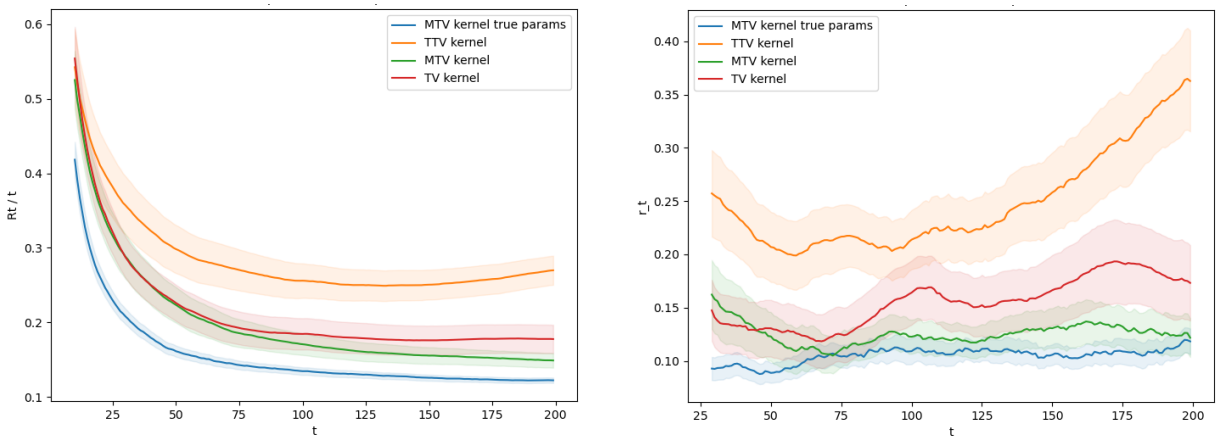
(a) Average regret $\frac{R_t}{t}$

(b) Instantaneous regret r_t (20 step moving average)

Figure 10.1: Numerical performance over time of algorithms on data generated by the TV model [200 trials, $\varepsilon = 0.01$]

10.2. MTV Function Data

In the MTV (figure 10.2) setting we see that similarly to the TV setting, the TTV model performs badly when the function is being continuously (randomly) perturbed. The MTV model with true parameters performs significantly better than all other models. We observe a larger difference between the model with true parameters and with parameter estimation compared to the TV setting. This is likely because it is harder to correctly learn this more complex temporal dependence from limited data. We do see that over time the MTV model is able to learn the parameters and obtain similar r_t to the model with true parameters. The TV model performs worse than the MTV model because it does not have the flexibility to fully capture the temporal dependence of this function f_t.



(a) Average regret $\frac{R_t}{t}$

(b) Instantaneous regret r_t (20 step moving average)

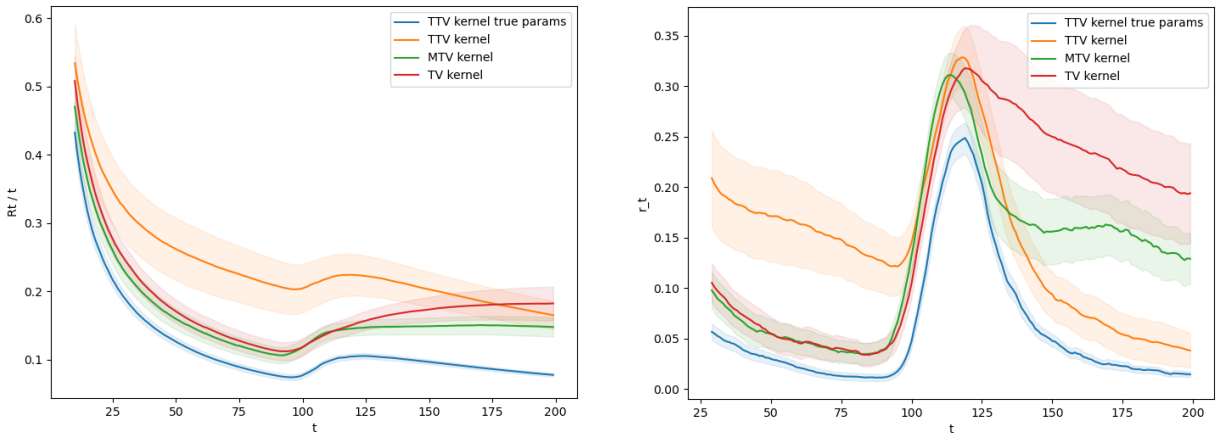
Figure 10.2: Numerical performance of algorithms on data generated by the MTV model [200 trials, $\varepsilon = 0.99$, $\alpha = 0.98$]

10.3. TTV Function Data

We now consider the TTV setting. In this case the function the functions transitions between two static functions. As we have set $\tau = 100$ center of the sigmoid occurs at $t = 100$ which implies that $f_{100}(x) = 0.5h_1(x) + 0.5h_2(x)$. In order to give some indication at how rapidly this transition occurs consider the timesteps at which f_t is 95% h_1 and h_2:

$$f_{85}(x) \approx 0.95h_1(x) + 0.05h_2(x)$$
$$f_{115}(x) \approx 0.05h_1(x) + 0.95h_2(x).$$

So we could say that the majority of the transition occurs between $t = 85$ and $t = 115$. In this case we keep $n\_restarts = 3$ fixed over time as the models must re-fit their kernel parameters after the steepest point of the transition occurs at $t = 100$. In figure 10.3 we present the results for this setting. We observe that the model with true parameters significantly outperforms the rest. This is because the model "knows" that the initial function is almost static and can exploit this. We see that the MTV model slightly outperforms the TV model (due to its increased flexibility). However, it is clear that both of these models were not designed for this setting. After $t = 100$ the models are not capable of discarding the early data which is mostly representative of h_1 while highly weighting the data after $t = 100$. On the other hand the TTV model with parameter estimation is able to reduce its instantaneous regret r_t and fit well to the underlying function. Overall the TTV with LML estimation of parameters still achieves poor cumulative regret in this setting. This implies that the model should only be used if there is a good prior on when we expect the function f_t to transition.



(a) Average regret $\frac{R_t}{t}$

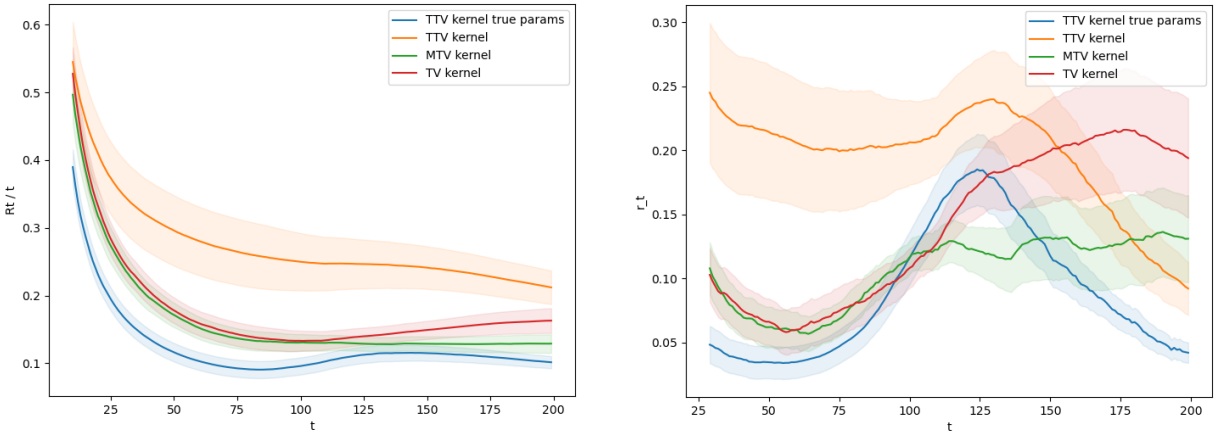
(b) Instantaneous regret r_t (20 step moving average)

Figure 10.3: Numerical performance of algorithms on data generated by the TTV model [200 trials, $\rho = 5$, $\tau = 100$]

We also perform a similar test with a more gradual transition. The results are presented in Figure 10.4. In this case we keep $\tau = 100$ and set $\rho = 20$ which implies that the majority of the transition occurs between timesteps 40 and 160:

$$f_{40}(x) \approx 0.95h_1(x) + 0.05h_2(x)$$
$$f_{160}(x) \approx 0.05h_1(x) + 0.95h_2(x).$$

We observe similar trends in the performance of the algorithms. In general we see that in this slower transition the MTV model is able to perform closer to the TTV model with true parameters. It is clear that the TTV model with parameter estimation performs quite poorly in this setting which reinforces the result that this model should only be used if we have a good prior for τ and ρ.



(a) Average regret $\frac{R_t}{t}$

(b) Instantaneous regret r_t (20 step moving average)

Figure 10.4: Numerical performance of algorithms on data generated by the TTV model [200 trials, $\rho = 20$, $\tau = 100$]

11
Conclusion and Further Research

This thesis has investigated the problem of minimizing the cumulative regret R_T while sampling a time-varying function f_t. This is relevant in problems where we are interested in simultaneously exploring and exploiting (maximizing) some unknown dynamic function. In chapter 1 we introduced the history of this topic as well as some of the applications. The problem falls into the class of online learning problems and can be seen as an extension to the multi-armed bandit problem [24]. In chapter 2 we discuss the progress that has been made for optimizing a static function. In 2012 the first sub-linear regret bounds for this setting were derived in [28]. The algorithm is based on the upper confidence bound (UCB) method where a single parameter (β_t) controls the exploration-exploitation trade-off at each timestep. Sub-linear regret implies that for long time horizons the average instantaneous regret r_t converges to zero. This is desirable as it implies that we are eventually able to sample the maximum point of the function with high certainty. The proof of this regret bound relies on a property of the Gaussian process (which is generating the function) known as the maximum information gain [5]. The maximum information gain allows us to quantify how rapidly we can reduce our uncertainty of the underlying function. It is fundamental property in regret analysis in Gaussian process setting and we see it return in the time-varying case.

In chapter 3, we discuss the literature on time-varying function case. In this setting the function is no longer static between samples. In 2016 Bogunovic et al. [2] introduced a new problem setting in which the function changes over time by being randomly perturbed between each sample. The rate of change of this function is controlled by a single parameter $\varepsilon \in (0, 1)$. The superposition of many random perturbations leads to the function changing significantly over long time horizons. This time-varying setting makes it harder to minimize the regret as old data becomes stale over time and is no longer representative of the current function. In fact, Theorem 4.1 in [2] proved that it is possible to achieve at best linear regret in this setting; sub-linear regret is not feasible. They adjust the UCB method which has previously been used for static function to develop an algorithm which achieves super-linear regret in this time varying setting. This proof relies on β_t increasing with time proportionally to $\log(t)$. In a similar manner to the static case their proof relies on the maximum information gain to bound the regret.

In chapter 4 we consider this same time-varying setting with random perturbations. We modify the algorithm introduced in [2] by using a constant value for $\beta_t = \beta$. We show that

by adjusting the derivation of the regret bound it is possible to achieve linear regret with this modified algorithm. This alternative approach motivates us to investigate the influence of the rate of the change of the function ε on the optimal value of the exploration-exploitation parameter β. This is a connection which was not present in the original regret bound presented in [2]. We hypothesize that a low rate of change ε of the function should lead to more exploration (higher β) as we have more timesteps to exploit the gained information before the data becomes stale. In chapter 5 we perform a simulation study to support these claims. We see that in every tested setting there exists a constant value of β is able to outperform a heuristic from literature where $\beta_t = \mathcal{O}(\log(t))$. Additionally, we observe that the optimal value of β is inversely correlated with ε. We find that choosing a constant value of β in the range $[2.0, 4.0]$ provides a good heuristic if the optimal value is unknown. However, more research is required to test a wider range of settings (higher dimensions, more erratic functions, etc.).

Following this we consider some of the use cases for these algorithms and introduce two new temporal models which we expect to be relevant for real world applications. The algorithm derived in [2] has been successfully applied to the task of hyperparameter optimization for deep reinforcement learning (DRL) in [20]. In chapter 6 we give a surface level introduction to DRL which provides the basis for the following chapters. We describe why time-varying functions are relevant in this problem and provide some motivation for introducing new temporal models. Chapter 7 discusses some of the considerations regarding implementing these algorithms for real world problems. In most applications we do not have full knowledge of the function generating process. Hence, we estimate the parameters of the Gaussian process using log marginal likelihood (LML) maximization. This adds some additional steps to our algorithm and has some implications for how we should design the new temporal models.

Chapter 8 introduces the Momentum Time Varying Gaussian Process Upper Confidence Bound (MTV-GP-UCB) algorithm. We first introduce a new temporal model which allows for the time-varying function to "trend" rather than change through random perturbations. The new temporal model can be seen as a generalization of the time-varying function model with random perturbations. It contains one additional parameter to model the rate at which the momentum decays. This also results in one additional parameter which has to be estimated when using the MTV-GP-UCB algorithm. We derive analogous regret bounds to those presented in chapter 4 and provide a modified upper bound on the maximum information gain for this setting. In chapter 9 we introduce the second new temporal model. In this case we are concerned with a function which is transitioning from one static function h_1 to a new function h_2. We introduce the Transition Time Varying Gaussian Process Upper Confidence Bound (TTV-GP-UCB) algorithm to handle this setting. The current version of the model is designed to handle a single transition in the dataset but can be expanded to handle multiple transitions.

Finally, in chapter 10 we test these new models in various settings. There are three types of temporal models discussed in this thesis; time-varying functions with random perturbations, time-varying functions with momentum and the transition time-varying function. In reality we often do not know the true temporal setting we are in and we might use the incorrect model. Hence, we generate data for all three settings and test all there models in every setting to consider how they perform when there is model misspecification. We observe that, as expected, each model outperforms the other models in the setting it was designed for. For the model misspecification case MTV model performs best overall which can be attributed to the high level of flexibility in the temporal kernel.

11.1. Further Research

There are various directions for further research in this topic. Theoretically, there is 4.1 which we have not been able to prove. We expect it to hold without any additional assumptions. However, proving it seems to be quite complex due to the difficulties in analysing the (conditional) distribution of the maximum of a Gaussian process.

With regards to applications the new algorithms from chapter 8 and chapter 9 still need to be tested on DRL hyperparameter optimization as discussed in chapter 6. This can allow us to identify which temporal structures are most relevant for this task and potentially develop new algorithms based on the results.

One of the major limitations of Gaussian process regression methods is that they scale poorly to large datasets. As discussed in chapter 7, the computational cost of estimating the mean and variance will scale according to $\mathcal{O}(t^3)$ in the general case. This cost arises from the inverse of the covariance matrix. However, the time-varying nature of the function opens the opportunity to greatly speed up the computation for long time horizons. This is a result of the matrix containing the temporal covariances $\mathbf{D}_t$. This matrix has a very specific structure, where entries which are far from the diagonal are by definition close to zero. Hence, it can be approximated by a banded matrix:

$$\mathbf{D}_t \sim \begin{bmatrix} D_{11} & D_{12} & D_{13} & 0 & \cdots & 0 \\ & D_{22} & D_{23} & D_{24} & \ddots & \vdots \\ & & D_{33} & D_{34} & D_{35} & 0 \\ & & & D_{44} & D_{45} & D_{46} \\ & \text{sym} & & & D_{55} & D_{56} \\ & & & & & D_{66} \end{bmatrix}$$

where we choose some bandwidth k and approximate all entries by zero if they fall outside the bandwidth:

$$D_{i,j} = 0 \quad \text{if} \quad |i - j| > k. \tag{11.1}$$

This results in a sparse matrix for large t. There are algorithms available for banded matrices [16] which allow for computing the inverse in $\mathcal{O}(k^2 t)$. We assume that we can choose some fixed bandwidth. Then, when using this approach we can compute the log marginal likelihood according to equation (7.1) in $\mathcal{O}(t)$. Since all of the data which is further in the past than the bandwidth can be disregarded we will be able to compute $\mu_t(x)$ and $\sigma_t^2(x)$ in $\mathcal{O}(1)$. Note that some additional work is required to determine the additional regret we can expect to incur by using the approximation equation (11.1). This computational speedup is one of the reasons that these algorithms are particularly well suited for handling time varying functions as even with large datasets it is still feasible to apply this method.

Another direction of research is the process of fitting the kernel parameters to the data. In this thesis the regret bounds are based on the assumption that the true kernel parameters are known to the user. In chapter 7 the LML maximization procedure is described which is the method used in chapter 10 to test the performance of models in the unknown parameter case. However, there are many alternative approaches available for fitting these parameters. One such approach involves using makov chain monte carlo (MCMC) [18] to sample from the marginal likelihood distribution (rather than simply maximizing it). This approach is quite computationally expensive which might not make it feasible for many applications. The complications around fitting the kernel parameters raises another question. Can we adjust our algorithm to sample points in a manner that allows us to learn these parameters faster? Currently we only consider exploring f_t but for many applications the kernel parameters must also be "explored". The results presented in chapter 10 imply that learning the parameters earlier could lead to significant improvements in the performance.

Finally, it would be interesting to consider a setting in which we can sample multiple points at each timestep. This occurs in applications when we are running models in parallel. For example in DRL we might choose to simultaneously train 8 agents with different combinations of hyperparameters. This requires us to re-define the problem. Rather than choosing a single point x_t we choose a set of n points; $\mathcal{X}_t = \{x_t^1, \dots, x_t^n\}$. The regret in this new setting could be defined as:

$$r_t = \max_{x \in D} \left[f_t(x) \right] - \max_{x \in \mathcal{X}_t} \left[f_t(x) \right].$$

By simply choosing all of the points as x_t according to our current algorithm we would achieve the same regret. However, it is interesting to consider how we could reduce the regret further by using all n points. Can we develop a better algorithm for this setting? And how does it scale with n? These are all relevant questions which can improve the functionality of these algorithms for applications.

References

[1] Robert J Adler and Jonathan E Taylor. *Random fields and geometry*. Springer Science & Business Media, 2009.

[2] Ilija Bogunovic, Jonathan Scarlett, and Volkan Cevher. "Time-varying Gaussian process bandit optimization". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 314–323.

[3] Paul Brunzema et al. "Event-triggered time-varying bayesian optimization". In: *arXiv preprint arXiv:2208.10790* (2022).

[4] Kathryn Chaloner and Isabella Verdinelli. "Bayesian experimental design: A review". In: *Statistical science* (1995), pp. 273–304.

[5] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications and Signal Processing. Wiley, 1991. ISBN: 9780471062592. URL: `https://books.google.nl/books?id=CX9QAAAAMAAJ`.

[6] Dennis D Cox and Susan John. "A statistical method for global optimization". In: *[Proceedings] 1992 IEEE international conference on systems, man, and cybernetics*. IEEE. 1992, pp. 1241–1246.

[7] Zihan Ding et al. "RLzoo: A Comprehensive and Adaptive Reinforcement Learning Library". In: *arXiv preprint arXiv:2009.08644* (2020).

[8] Tom Edgar. "Staircase series". In: *Mathematics Magazine* 91.2 (2018), pp. 92–95.

[9] William Feller. *An introduction to probability theory and its applications, Volume 2*. Vol. 81. John Wiley & Sons, 1991.

[10] Subhashis Ghosal and Anindya Roy. "Posterior consistency of Gaussian process prior for nonparametric binary regression". In: (2006).

[11] GPy. *GPy: A Gaussian process framework in python*. `http://github.com/SheffieldML/GPy`. since 2012.

[12] Robert Hecht-Nielsen. "Theory of the backpropagation neural network". In: *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[13] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, 2012. ISBN: 9781139788885. URL: `https://books.google.nl/books?id=O7sgAwAAQBAJ`.

[14] Carl Hvarfner, Erik Orm Hellsten, and Luigi Nardi. "Vanilla Bayesian Optimization Performs Great in High Dimension". In: *arXiv preprint arXiv:2402.02229* (2024).

[15] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. "High dimensional Bayesian optimisation and bandits via additive models". In: *International conference on machine learning*. PMLR. 2015, pp. 295–304.

[16] Emrah Kılıç and Pantelimon Stanica. "The inverse of banded matrices". In: *Journal of Computational and Applied Mathematics* 237.1 (2013), pp. 126–135.

[17] Achim Klenke. *Wahrscheinlichkeitstheorie*. Vol. 1. Springer, 2006.

[18] Vidhi Lalchand and Carl Edward Rasmussen. "Approximate inference for fully Bayesian Gaussian process regression". In: *Symposium on Advances in Approximate Bayesian Inference*. PMLR. 2020, pp. 1–12.

[19] Yuxi Li. "Deep reinforcement learning: An overview". In: *arXiv preprint arXiv:1701.07274* (2017).

[20] Jack Parker-Holder, Vu Nguyen, and Stephen J Roberts. "Provably efficient online hyperparameter optimization with population-based bandits". In: *Advances in neural information processing systems* 33 (2020), pp. 17200–17211.

[21] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[22] Antonin Raffin et al. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8. URL: `http://jmlr.org/papers/v22/20-1364.html`.

[23] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2005. ISBN: 9780262182539. URL: `https://books.google.nl/books?id=GhoSngEACAAJ`.

[24] Herbert Robbins. "Some aspects of the sequential design of experiments". In: (1952).

[25] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[26] Jaemin Seo et al. "Avoiding fusion plasma tearing instability with deep reinforcement learning". In: *Nature* 626.8000 (2024), pp. 746–751.

[27] Mohit Sewak. *Deep reinforcement learning*. Springer, 2019.

[28] Niranjan Srinivas et al. "Information-theoretic regret bounds for gaussian process optimization in the bandit setting". In: *IEEE transactions on information theory* 58.5 (2012), pp. 3250–3265.

[29] Victor Talpaert et al. "Exploring applications of deep reinforcement learning for real-world autonomous driving systems". In: *arXiv preprint arXiv:1901.01536* (2019).

[30] Mark Towers et al. "Gymnasium: A Standard Interface for Reinforcement Learning Environments". In: *arXiv preprint arXiv:2407.17032* (2024).