Safe Exploration in Reinforcement Learning

A project report submitted

in partial fulfillment for the award of the degree of

Master of Technology

in

Machine Learning and Computing

by

Durgesh Kalwar

Department of Mathematics
Indian Institute of Space Science and Technology
Thiruvananthapuram, India

June 2021

Certificate

This is to certify that the project report titled ***Safe Exploration in Reinforcement Learning*** submitted by **Durgesh Kalwar**, to the Indian Institute of Space Science and Technology, Thiruvananthapuram, in partial fulfillment for the award of the degree of **Master of Technology** in **Machine Learning and Computing** is a bona fide record of the original work carried out by him under my supervision. The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. Vineeth B.S. Assistant Professor Department of Avionics (Supervisor)	Dr. Sumitra S. Associate Professor Department of Mathematics (Department-guide)	Dr. Nicholas Sabu Professor & Head Department of Mathematics

Place: Thiruvananthapuram
Date: June 2021

Declaration

I declare that this project report titled ***Safe Exploration in Reinforcement Learning*** submitted in partial fulfillment for the award of the degree of **Master of Technology** in **Machine Learning and Computing** is a record of the original work carried out by me under the supervision of **Dr. Vineeth B.S.**, and has not formed the basis for the award of any degree, diploma, associateship, fellowship, or other titles in this or any other Institution or University of higher learning. In keeping with the ethical practice in reporting scientific information, due acknowledgments have been made wherever the findings of others have been cited.

Place: Thiruvananthapuram
Date: June 2021

Durgesh Kalwar
(SC19M077)

Acknowledgements

I would like to take this opportunity to sincerely thank my faculty guide, Dr. Vineeth B.S., for offering me an opportunity to gain exposure in the research domain and for helping me throughout with his constant guidance and encouragement. His enormous support and valuable feedback helped me throughout the duration of my project and the writing of this thesis.

I would also like to thank my co-advisor Dr. Sumitra S. for her guidance during my project.

At last, I would like to extend my heartfelt thanks to my parents because without their help this journey would not be so easy. Finally, I would like to thank my dear friends who have been with me all the time and made this journey very joyful.

Durgesh Kalwar

Abstract

Reinforcement learning has seen significant advances over the last decade in simulated or controlled environments. This lead to an interest in deploying these methods in real-world application problems, where they face the following combination of challenges- learning in safety-critical environments, learning in partially observable and non-stationary environments, learning with limited samples, learning in high dimensional environments, system delay, real-time inference at control frequency of the system and learning from off-logs.

In this dissertation, we consider learning in non-stationary and safety-critical environments. We know that the classical reinforcement learning methods are often able to achieve high-level performance but they don't guarantee the safety of the environment during the learning process so they can't be deployed in a safety-critical environment without posing safety constraints to both learner and environment.

We first consider the safe exploration in the switching environment for which we have proposed an Adaptive Safe-Explore policy that incorporates Bayesian inference and change detection-based scheme. This policy balances two objectives: the first is to expand the safe region from the initial safe seed, and the second to detect the change point. We observe that Adaptive Safe-Explore is able to expand the safe region as well as detect the change-points if environment switches and we also observe that a major challenge in adapting to the switching change is to identify safe decisions when the change-point is detected and prevent attraction to local optima in unsafe region.

Secondly, we consider the safe sequential optimization in the switching environments for which we have proposed an Adaptive-SafeOpt policy that is an extension of the Adaptive Safe-Explore policy to also include optimization. This policy balances three objectives: the first is to expand the safe region, the second is to obtain the high reward region, and the third is to detect the change-point. We note that the proposed Adaptive-SafeOpt converges but since the safe set that they explore is limited in size, the convergence is to a local maxima. Here also we observe the same challenge of finding the safe seed at the time when change-point detected and preventing to local maxima attraction in unsafe region.

In the above-mentioned problems, we have considered the objective/reward and safety function as a non-linear and time-variant function. Thirdly, we consider a simplified problem in which the reward function is linear and time time-invariant and the objective is to maximize this reward function subject to linear safety constraints. This problem setup is known as safe optimization in linear bandit setting. We proposed a Thompson sampling based algorithm for this problem which has one parameter to tune (excluding prior parameters) compare to the Adaptive SafeOpt algorithm.

Contents

1 Introduction **1**

1.1 Prior Work 2

1.2 Our contribution 4

1.3 Thesis Outline 4

2 Background and Notation **6**

2.1 Reinforcement Learning Preliminaries 6

2.2 Bayesian Optimization 10

2.3 Gaussian Process Regression 11

2.4 Reinforcement learning for Dynamically Varying Environments 22

3 Safe Exploration in Switching Environment **25**

3.1 System Model and Problem Statement 25

3.2 Adaptive Safe-Explore 27

3.3 Baseline Algorithms 28

3.4 Simulations and Performance Analysis 30

3.5 Conclusions 33

4 Safe Sequential Optimization for Switching Environments **34**

4.1 System Model and Problem Statement 34

4.2 Background: Safe-Opt Algorithm 36

4.3 Adaptive SafeOpt 37

4.4 Baseline Algorithms 40

4.5 Simulations and Performance Analysis 41

4.6 Conclusions 49

5 Safe Thompson Sampling for Linear Bandits **50**
5.1 System Model & Problem Statement 50
5.2 Bayesian Linear Regression 52
5.3 Bayesian Safe-LTS Algorithm 53
5.4 Baseline Algorithm 54
5.5 Simulation and Performance Analysis 54
5.6 Conclusions 58

6 Conclusion **59**
6.1 Future Work 60

Bibliography **60**

Chapter 1
Introduction

Reinforcement learning (Sutton and Barto, 1998 [1]) has become the popular paradigm to learn optimal policies by directly interacting with an a priori unknown dynamical environment. For example, the reinforcement learning methods can control complex video games like Atari directly from the image perceptions (Mnih et al., 2015 [2]) and can defeat the best human players at board games like GO (Silver, Huang, et al., 2016 [3]). For these kinds of environments, reinforcement learning methods can achieve human-level performance or better because they are trained on simulation for thousands of hours. The data-driven nature of these methods leads to an interest in deploying these methods to real-world application problems. Unlike in simulation, in the physical world actions have real consequences. So these methods can't be deployed in a safety-critical environment without posing safety constraints to both learner and environment.

Guaranteeing safety for real-world systems like robotics and healthcare system is a very critical issue, especially when the system or the environment is non-stationary. For example, Krause et al. [4] considered the problem of an autonomous rover exploring the surface of Mars. The rover does not know the height or gradient of the surface which it is exploring. Since the rover has physical limitations with respect to the gradients it can move over, it has to safely explore the surface. In addition, the rover should explore those locations on the surface which maximize scientific insight. The safe exploration in the above problem becomes more difficult when the altitude of the Mars surface changes with time due to asteroid hit or sand coagulation which affects the safety constraints. For another example, Berkenkamp et al. [5] considered the problem of optimizing the controller parameters of a flying quad-copter which becomes more difficult when the environmental conditions change with time, the optimal parameter for the controller needs to be adapted to the changes in the environment.

Prior work [6], [7] has used the framework of Bayesian optimization in order to propose sequential decision making agents for the above safe exploration and optimization problem but mainly for a function that is not time-varying. In this dissertation, we consider the important extension of the above problem to the case where the unknown function is time varying and exhibits a discontinuous *switching* to another unknown function at a change-point epoch. The non-stationarity of the environments that agents have to contend with is an important challenge for real-world problems [8]. We propose heuristic policies Adaptive Safe-Explore for safe exploration and Adaptive Safe-Opt for safe optimization (which is an extension of the algorithm in [6] with change detection for the unknown functions) and evaluate the performance of these policies using simulations. We observe that an important challenge in this problem is the re-initialization of an estimate of the safe set once the change has been detected and propose a solution. And we also propose a heuristic policy Bayesian Safe-LTS (linear Thompson sampling) for safe optimization in linear bandit problem which is special case of above problem.

1.1 Prior Work

Bayesian optimization has been used for addressing the problem of safe exploration and optimization in which the unknown objective function and/or safety constraints are modelled using Gaussian processes. The quantification of uncertainty, which is obtained for free with the Bayesian framework, is used to decide a sequence of safe actions with high probability. For bandit setting, Sui et al. 2015 [6] proposed Safe-Opt algorithm (Safe exploration for Optimization), in which an a priori unknown safety function is modelled using Gaussian Processes (GP) and it's confidence interval is used to decide whether a sequence of decisions taken during exploration is safe or not. If at a decision time the value of the safety function is more than a threshold then it is safe. In their setting the safety and objective function are identical. The proposed Safe-Opt algorithm trades off between maximizing the size of the safe set of decisions starting from an initial safe seed and finding the optimal reachable decision in that safe set. We note that the authors considered that the unknown function is stationary with time. In contrast to their work we consider a non-stationary scenario and propose a change point detection based extension to Safe-Opt. We note that in addition to the trade-off between maximising the size of the safe-set and optimal reachable decision we also have another trade-off in the exploration required to detect the change-point.

For Markov Decision Process (MDP) setting, Krause et al. 2017 [4] proposed a safe exploration algorithm called Safe-MDP. In their work they assumed that the transition model is known and safety function is unknown. The safety function is then modelled using GP and also follows some regularity assumption such as similar values for *similar* states. In their work they considered exploration in stationary MDP setting. Wachi et al. 2018 [7] proposed a safe exploration with optimization algorithm for finite deterministic MDP and provides theoretical guarantees on the policy being safe with high probability during the exploration process, but this policy may not get the near-optimal cumulative reward. Wachi and Sui 2020 [9] proposed a safe RL algorithm for finite deterministic MDP that guarantees that the acquired policy will get near-optimal cumulative reward while being satisfying the safety constraint as well during the learning process. In their work, they also assumed the transition model is known and both reward and safety function is modelled using GP. Wachi, Kajino and Munawar 2018 [10] extended [4] to the case of time-variant safety functions. They assumed that the time-variation of safety functions satisfies a Lipschitz continuity condition.

We note that optimization of unknown time varying (switching) functions without safety constraints has been addressed by many authors. Mellor and Shapiro [11] had proposed a Bayesian online change point detection based method for switching bandits. A similar approach was also used by [12]. Recently, Ghatak [13] had proposed a change detection based Thompson sampling framework for non-stationary bandits. Padakandla et al. [14] provides a survey of reinforcement learning algorithms for dynamically varying environments. We note that our work incorporates the notion of safety in addition to the non-stationarity considered in the above papers.

We note that there are multiple approaches to ensuring *safety* for agents which include the one summarized above. For example, the risk sensitive approach consists of a modification of the optimality criterion. In such work, the long-term reward maximization is transformed to include some notion of risk (risk-sensitive reinforcement learning) related to the variance of the return or its worst-outcome (Coraluppi et al. 1999 [15]). However, these approaches only minimize risk and do not treat safety as a hard constraint. In other works, the optimization criterion is transformed to include the probability of visiting error states (Geibel and Wysotzki 2005 [16]) define risk as the probability of driving the system to a previously known set of undesirable states. Garcıa and Fernández, 2015 [17] provides a succinct survey on safe reinforcement learning. In other work on safe reinforcement learning, Moldovan and Abbeel (2012) [18] consider the problem of safe exploration in MDPs. They ensure safety by restricting policies to be ergodic with high probability, i.e., able to

recover from any state visited. This is computationally demanding even for small state spaces and doesn't provide convergence guarantees. Biyik et al. 2019 [19], consider the problem of safe exploration in deterministic MDPs with unknown transition models. They considered safety criterion similar to that in [18]. Roderick et al. 2019 [20], consider the problem of safe exploration in PAC (probably approximately correct)-MDP with unknown, stochastic dynamics. They assume the environment has a fixed initial states which is safe and the reward function is known a-priori and bounded between -1 and 1; the rewards that are negative denote dangerous state-actions and by using a analogous function they define the similarity between sate-action pairs.

1.2 Our contribution

In this dissertation our contribution is that-

- We formulated the problems of safe exploration and safe sequential optimization for switching environments.
- In safe exploration problem we identified a two way trade-off for exploration for safety and exploration for change points. For that we proposed a safe exploration algorithm called Adaptive Safe-Explore.
- In safe sequential optimization problem we identified a three way trade-off exploitation for the maximization, exploration for safety, exploration for change points. For that we proposed a safe optimization algorithm called Adaptive SafeOpt.
- We considered a simplified problem of safe optimization in linear bandit setting and proposed a Thompson sampling based algorithm called Bayesian SafeLTS.
- We evaluated the performance of proposed algorithms via simulation and also compared their performance with baseline algorithms.

1.3 Thesis Outline

- **Chapter** 2 In this chapter we discuss the necessary background and notation of reinforcement learning, Bayesian optimization framework setting, Gaussian process regression, and dynamically varying environments.

- **Chapter** 3 In this chapter we first state the system model and problem statement for the safe exploration problem in the switching environments and propose an Adaptive Safe-Explore policy to address this problem and evaluate its performance via simulation and compare with baseline policies.
- **Chapter** 4 In this chapter we first state the system model and problem statement for the safe optimization problem in the switching environments and propose an Adaptive SafeOpt policy to address this problem and evaluate its performance via simulation and compare with baseline policies.
- **Chapter** 5 In this chapter we first state the system model and problem statement for the safe optimization problem in the linear bandit setting and propose a Bayesian Safe-LTS policy to address this problem and evaluate its performance via simulation and compare with baseline policies..
- **Chapter** 6 In this chapter we conclude all the results and observations. And we also discuss the future scope ot this work.

Chapter 2

Background and Notation

In this chapter we will introduce the mathematical formulation of reinforcement learning [1] in section 2.1, In Section 2.2 we will discuss the Bayesian optimization framework setting to optimize a unknown function, and a detailed discussion on Gaussian process regression in section 2.3 which is a very useful tool to model the unknown function and gives the uncertainty of the function in terms of the probability distribution. In section 2.4 we formally define the dynamically varying environments. We define our notation in each section of this chapter which we will use in this dissertation.

2.1 Reinforcement Learning Preliminaries

Reinforcement learning provides a mathematical framework to design a optimal policy for sequential decision making process.

Definition 2.1. (Markov Decision Process). The Markov Decision Process (MDP) provides a mathematical framework to model the sequential decision making problems. The MDP is defined as a tuple $M =< S, A, T, R, \gamma >$, where S is set of states $s \in S$ (discrete or continuous), A is set of actions $a \in A$ (discrete or continuous), T transition function defines a conditional probability distribution of the form $T(s_{t+1}|s_t, a_t)$ that describes the dynamics of the system, $R : S \times A \to \mathbb{R}$ is reward function, $\gamma \in (0, 1]$ is a scalar discount factor.

In reinforcement setting the state transition probability $T(s_{t+1}|s_t, a_t)$ and the reward function R is not known. Figure 2.1 illustrate the reinforcement learning framework in which agent-environment interact with each other at a sequence of discrete time steps, $t = 0, 1, 2, 3, ...H$, where H is time horizon. Where agent is the decision maker and other than that are included in the environment. In context of the control theory the agent corresponds

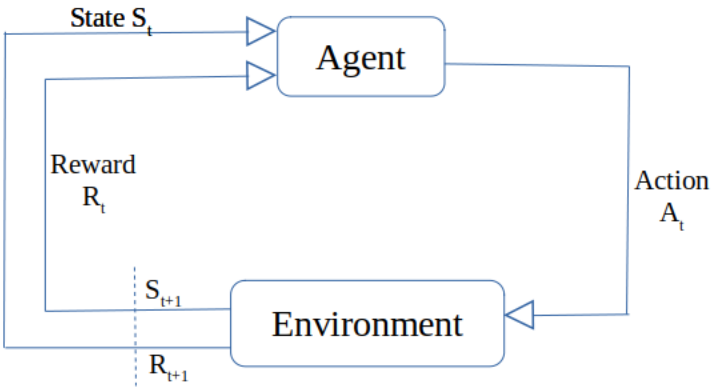


Figure 2.1: The agent–environment interaction.

to controller and the environment corresponds to system or plant. The agent observes the state ($s_t \in S$) of the environment at time t and takes an action ($a_t \in A$), as a consequence the environment evolves to a state $s_{t+1} \in S$ according to the transition function T, and the agent gets a reward R_{t+1} for taking action a_t in state s_t. The environment and agent together thereby give rise to a trajectory τ, which is a sequence of states and actions of length H, given by $\tau = (s_0, a_0, ..., s_H, a_H)$, where H can be finite or infinite based on the task, according to this MDP is categorized as finite MDP or infinite MDP respectively. Here the reward value at every time the agent receive for taking action in current state quantify how good or bad that action is. So informally the goal of the agent is to maximize this reward over the time.

Policy- At every time step the agent take an action $a \in A$ according to a policy, which defines a distribution over actions conditioned on states, $\pi(a_t|s_t)$.

Formally, the agent's goal is to learn a policy π, which maximize the expected cumulative reward over the trajectory distribution $p_\pi(\tau)$. where $p_\pi(\tau)$ is given by-

$$p_\pi(\tau) = d_0(s_0) \prod_{t=0}^{H} \pi(a_t|s_t) T(s_{t+1}|s_t, a_t) \tag{2.1}$$

where $d_0(s_0)$ is initial state distribution.

The learning objective, $J(\pi)$, can then be written as an expectation under this trajectory distribution:

$$J(\pi) = \mathbb{E}_{\tau \sim p_\pi(\tau)}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t)] \tag{2.2}$$

In equation 2.2, if we don't account the scalar discount factor γ then in case of infinite MDP ($H = \infty$) the objective function $J(\pi)$ value will not be finite. So in case of infinite MDP discount factor $\gamma \in (0, 1)$. Or in case of finite MDP we can put $\gamma = 1$.
Almost all reinforcement learning algorithms involve estimating value functions which are state-value function ($V^\pi(s)$) and state-action value function $Q^\pi(s, a)$ under a policy π. $V^\pi(s)$ gives an idea about how good for agent to be in state s under the policy π. And $Q^\pi(s, a)$) give an idea about how good taking action a in state s for given policy. By finding an optimal value of these functions, we can find optimal policy π^* from them. These function are defined as

Definition 2.2. State value function $V^\pi(s_t)$ is defined as an estimate of the expected cumulative discounted reward that will be obtained by following some policy $\pi(a_t|s_t)$ when starting from a given state s_t.

$$V^\pi(s_t) = \mathbb{E}_{\tau \sim p_\pi(\tau|s_t)}[\sum_{t'=t}^{H} \gamma^{t'-t} R(s_t, a_t)] \tag{2.3}$$

Definition 2.3. State-action value function $Q^\pi(s, a)$ is defined as an estimate of the expected cumulative discounted reward that will be obtained by following some policy $\pi(a_t|s_t)$ when starting from a given state s_t and taking action a_t.

$$Q^\pi(s_t, a_t) = \mathbb{E}_{\tau \sim p_\pi(\tau|s_t, a_t)}[\sum_{t'=t}^{H} \gamma^{t'-t} R(s_t, a_t)] \tag{2.4}$$

From this, we can derive recursive definitions for these value functions, which are given as

$$V^\pi(s_t) = \mathbb{E}_{a_t \sim \pi(a_t|s_t)}[Q^\pi(s_t, a_t] \tag{2.5}$$

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \in T(s_{t+1}|s_t, a_t)}[V^\pi(s_{t+1})] \tag{2.6}$$

$$Q^\pi(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \in T(s_{t+1}|s_t, a_t), a_{t+1} \in \pi(a_{t+1}|s_{t+1})}[Q^\pi(s_{t+1}, a_{t+1})] \tag{2.7}$$

Optimal policy and optimal value function- A policy π is better than other policy π' if its expected discounted cumulative reward is greater than or equal to that of π' for all states means $V^\pi(s) \geq V^{\pi'}(s)$ for all $s \in S$. There is always at least one policy that is better than

or equal to all other policies. This is an optimal policy π^* which is defined as

$$V^*(s) = \max_{\pi} V^{\pi}(s), \forall s \in S \tag{2.8}$$

Optimal policies also share the same optimal action-value function, denoted Q^* , and defined as

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a), \forall s \in S \& a \in A \tag{2.9}$$

And the optimal deterministic policy can be given as-

$$\pi^*(.|s) = \operatorname*{argmax}_{a} Q^*(s,a) \tag{2.10}$$

Definition 2.4. (Reinforcement Learning). Reinforcement learning (RL) provides a computational framework for learn to control autonomously from interactions with the surrounding world and concerned with sequential decision making in unknown environments.

2.1.1 Types of RL algorithms

Here we only discuss model-free and model-based RL algorithms briefly. In Model-based RL algorithms agent learn the optimal policy by estimating T and R from state and reward samples. In Model-free algorithms the agent do not estimate T and R functions, instead of that it these directly either estimate value function of a policy and then improve the policy (value based methods) or directly find the optimal policy (policy gradient based methods).

2.1.1.1 Policy Gradient Based

In this type of algorithm parametrized policy $\pi(\theta)$ is used. And the reinforcement learning objective $J(\pi(\theta))$ is optimized wrt parameter θ to obtain the optimal policy $\pi^* = \pi(\theta^*)$.

$$\theta^* \leftarrow \operatorname*{argmax}_{\theta} J(\pi(\theta)) \tag{2.11}$$

2.1.1.2 Value Based

In this type of algorithm the optimal policy π^* is obtained explicitly by obtaining the near optimal state-action value function $Q^{\pi}(s,a)$. The approximate value of $Q^{\pi}(s,a)$ can be estimated by using Monte-Carlo or temporal difference methods. This kind of algorithm

involve two steps first is policy evaluation-

$$Q_{k+1}^{\pi_k}(s,a) = r(s,a) + E_{s',a'}[Q_k^{\pi_k}(s',a')]$$

and second step is policy improvement-

$$\pi_{k+1} \leftarrow \underset{a}{\operatorname{argmax}}\, Q_{k+1}^{\pi_k}(s,a)$$

2.1.1.3 Model Based

In this type of algorithm we explicitly learn the model of the system (transition function and reward function) from some regression based methods. This learned model can be utilized for planning at test time, often by means of model-predictive control (MPC), or can be combined with model free methods (value based and policy gradients based) to find the optimal policy like Dyna algorithm.

2.2 Bayesian Optimization

It is an approach to optimize a priori unknown objective function $f(x)$, by modelling the function using Bayesian regression methods which captures the uncertainty over the function. So it is a technique, focused on solving the problem

$$\max_{x \in \mathcal{X}} f(x)$$

where $\mathcal{X} \subseteq \mathcal{R}^d$, which is in literature known as design space or parameter space or action space.

The formal problem setting of Bayesian optimization can be formulated by by agent-environment interaction (see Figure 2.1) with a fixed state or stateless setting which is also known as bandit setting. So at each time agent select $x_t \in \mathcal{X}$, and interact with the environment to evaluate the noisy observation y_t of the function $f(x)$ at x_t. And add (x_t, y_t) to the buffer or the train set from which it update the model of the function which maintain the Bayesian belief or posterior over what this unknown objective function $f(x)$ might be given the observations it has collected so far.

There is two main ingredient in Bayesian optimization, first is the model class for that we use Gaussian process (which is discussed in next section 2.3) to model the function which captures the structural and measurement uncertainty. Second is sampling criteria

from which we sample the next x and we evaluate the function $f(x)$ at x.

2.2.1 Safe Bayesian Optimization

In safe Bayesian optimization instead of optimizing the unknown objective function $f(x)$ globally, it finds the maximizer within the safe set that is defined by safety constraints which not known to agent. So This safe set is not known initially, but estimated after each safety constraints evaluation. So it is a technique, focused on solving the problem

$$\max_{x \in \mathcal{X}} f(x) \text{ such that } g(x_t) \geq 0, \ \forall t \text{ with probability} \geq 1 - \delta$$

The formal problem setting of the safe Bayesian optimization can be explained by agent-

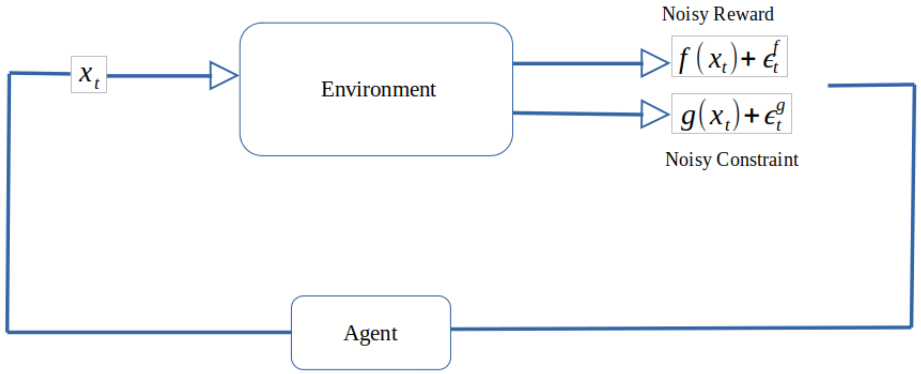


Figure 2.2: The agent–environment interaction in safe Bayesian optimization.

environment interaction (see Figure 2.2, only one safety constrain is considered could be multiple). So at time t agent select a action x_t from the estimated safe set (every x in the estimated safe set should satisfy the safety constrain with high probability.) and interact with the environment and observe the noisy observations y_t and z_t of the objective function $f(x)$ and safety function $g(x)$ at x_t respectively. And add (x_t, y_t, z_t) to the buffer or train set from which it update the Bayesian belief about the unknown objective and safety function.

2.3 Gaussian Process Regression

Regression is the problem of fitting a function f given a set of feature vectors $x_i \in \mathbb{R}^d$ and target vector $y_i = f(x_i) + \epsilon_i \in \mathbb{R}$ which corresponds to the function values f_i, where ϵ_i is a noise term. In the context of the reinforcement learning, regression problem arises frequently. For example, in model-based reinforcement learning estimating the unknown

transition function (which defines the dynamics of the system) and unknown reward function, or in safety-critical system estimating the safety functions comes under in regression problem. Due to the structural and measurement uncertainty, the estimate of the function f is uncertain. To capture this uncertainty in terms of a probability distribution, the Bayesian inference-based Gaussian process (GP) framework is widely used.

Definition 2.5. A Gaussian process is defined as a distribution over functions and can be generalized by the infinite-dimensional normal distribution. Rasmussen and Williams [21] define the Gaussian process as a collection of random variables which have a joint Gaussian distribution.

A Gaussian process is characterized by its mean function $m(x)$ and covariance function $k(x, x')$, which are defined as

$$m(x) = \mathbb{E}[f(x)],$$

$$k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \tag{2.12}$$

and will write the Gaussian process as

$$f(x) \sim \mathbb{GP}(m(x), k(x, x')) \tag{2.13}$$

without loss of generality mean function $m(x)$ can be assumed to zero.

In our case the random variable represent the value of the function $f(x)$ at location x, those joint distribution is given by multivariate normal distribution and it's mean can be assumed to be zero and covariance matrix is given by the covariance function which define the smoothness of the function. In the Gaussian process regression, we denote $(X, \mathbf{y})$ as training set where $X := [x_1, ..., x_n]$is a design matrix and, $\mathbf{y} := [y_1, ..., y_n]^\top$ is the target vector. And y_i is generated according to $y_i = f(x_i) + \epsilon_i$, where $f : \mathbb{R}^d \to \mathbb{R}$ and $\epsilon_i \sim N(0, \sigma_\epsilon^2)$ is independent Gaussian measurement noise. GPs consider f as a random function and infer a posterior distribution $p(f|X, \mathbf{y})$ over f from the GP prior $p(f)$. The posterior is used to make predictions about function values $f(x^*)$ at arbitrary inputs $x^* \in \mathbb{R}^d$.

2.3.1 Bayesian Inference within GP framework

To find the posterior distribution over the function f we use the Bayesian inference technique which can be considered as a three-step procedure:

- First, define a prior on the function f.
- Second, observe the data points.
- Third, calculate the posterior distribution over f by refining the prior distribution using the information from the data points.

2.3.1.1 GP Prior

For the given hyper-parameter θ GP prior is defined as -

$$p(f|\theta) = \mathbb{GP}(m_f, k_f) \tag{2.14}$$

where mean function m_f can be assumed to zero with loss of generality and the covariance function is define by $k_f = k_{SE}(x_p, x_q) + \delta_{pq}\sigma_\epsilon^2$. Where k_{SE} is a squared exponential function which is defined by

$$k_{SE}(x_p, x_q) = \alpha^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^T(x_p - x_q)\right) \tag{2.15}$$

where α is the signal variance of the function f, l is the length scale and σ_ϵ^2 is the noise variance. These parameters combined vector is defined as hyper-parameter θ.

2.3.1.2 GP Posterior

After n observation the training set $(X, \mathbf{y})$ is $X := [x_1, ..., x_n]$, $\mathbf{y} := [y_1, ..., y_n]$. From this information the prior belief over the function f is updated to get the posterior distribution over f using Bay's theorem

$$p(f|X, y, \theta) = \frac{p(y|f, X, \theta)p(f|\theta)}{p(y|X, \theta)} \tag{2.16}$$

where $p(y|X, \theta)$ is the normalizing constant, also known as the marginal likelihood or evidence is independent of f and given by

$$p(y|X, \theta) = \int p(y|f, X, \theta)p(f|\theta)\, df \tag{2.17}$$

We assume that the observations y_i are conditionally independent given X. Therefore, the likelihood of f factors according to

$$p(y|f, X, \theta) = \prod_{i=1}^{n} p(y_i|f(x_i), \theta) = \prod_{i=1}^{n} N(y_i|f(x_i), \sigma_\epsilon^2),$$

$$= N(y|f(X), \sigma_\epsilon^2 I) \tag{2.18}$$

For given hyper-parameters θ, the Gaussian likelihood $p(y|X, f, \theta)$ in equation (2.18) and the GP prior $p(f|\theta)$ lead to the GP posterior

$$p(f|X, y, \theta) \propto (N(y|f(X), \sigma_\epsilon^2 I))(N(0, k_f)) \tag{2.19}$$

$$p(f|X, y, \theta) = \mathbb{GP}(\mathbb{E}_f[f(\tilde{x})|X, y, \theta], cov_f[f(\tilde{x}), f(x')|X, y, \theta]) \tag{2.20}$$

So, the posterior distribution is also given by Gaussian process, whose mean function and a covariance function are given by

$$\mathbb{E}_f[f(\tilde{x})|X, y, \theta] = k_f(\tilde{x}, X)(k_f(X, X) + \sigma_\epsilon^2 I)^{-1} y \tag{2.21}$$

$$cov_f[f(\tilde{x}), f(x')|X, y, \theta] = k_f(\tilde{x}, x') - k_f(\tilde{x}, X)(k_f(X, X) + \sigma_\epsilon^2 I)^{-1} k_f(X, x') \tag{2.22}$$

where $\tilde{x}, x' \in \mathbb{R}^d$ are the test inputs. we write $k_f(X, x')$ for $[k_f(x_1, x'), ..., k_f(x_n, x')] \in \mathbb{R}^{n \times 1}$. Note that $k_f(x', X) = k_f(X, x')^T$.

2.3.1.3 Predictive Distribution

To make predictions for a test case we average over all possible functions, weighted by their posterior probability. Thus the predictive distribution for $f_* \triangleq f(x_*)$ at x_* is given by averaging the output of all possible function values w.r.t. the GP posterior.

$$p(f_*|x_*, X, y) = \int p(f_*|x_*, f) p(f|X, y, \theta) \tag{2.23}$$

So the predictive distribution is also given by Gaussian process whose mean function and covariance function is given by in equations (2.25) and (2.28) respectively.

The function values for test inputs and training inputs are jointly Gaussian, that is,

$$P(f, f_*|X, X_*) \sim N\left(\begin{bmatrix} m_f(X) \\ m_f(X_*) \end{bmatrix}, \begin{bmatrix} K & k_f(X, X_*) \\ k_f(X_*, X) & k_f(X_*, X_*) \end{bmatrix} \right) \tag{2.24}$$

where K the kernel matrix with $K_{ij} = k_f(x_i, x_j)$.
Univariate Predictions: $x_* \in \mathbb{R}^D$, $y_* \in \mathbb{R}$

$$\mu_* := m_f(x_*) := \mathbb{E}_f[f(x_*)|X, y] = k_f(x_*, X)(K + \sigma_\epsilon^2 I)^{-1} y \tag{2.25}$$

$$= k_f(x_*, X)\beta = \sum_{i=1}^{n} \beta_i k(x_i, x_*) \tag{2.26}$$

$$\sigma_*^2 := \sigma_f^2(x_*) := var_f[f(x_*)|X, y] \tag{2.27}$$

$$= k_f(x_*, x_*) - k_f(x_*, X)(K + \sigma_\epsilon^2 I)^{-1} k_f(X, x_*) \tag{2.28}$$

2.3.2 Pseudo Code

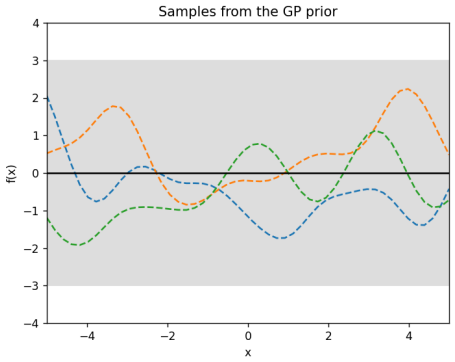
Algorithm 2.1: Gaussian Process Regression [21]

Input: training set $(X, \mathbf{y})$, k (covariance function), σ_n^2 (noise variance) , x_* (test input)

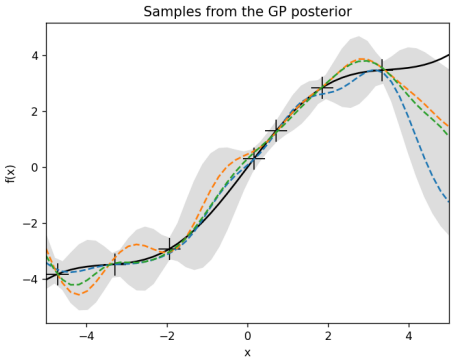
Output: f_* (mean), $\mathbb{V}[f_*]$ (variance), $\log p(y|X)$ (log marginal likelihood)

1 L := cholesky($K + \sigma_n^2 I$)
2 α := L^T /(L /y)
3 $\bar{f}_*$ = $k_f(x_*, X)^T \alpha$ ▷ predictive mean eq. (2.25)
4 v := L /$k_f(x_*, X)$
5 $\mathbb{V}[f_*] := k_f(x_*, x_*) - v^T v$ ▷ predictive variance eq. (2.28)
6 $\log p(y|X) := -\frac{1}{2} y^T \alpha - \sum_i \log L_{ii} - \frac{D}{2} \log 2\pi$

Explanation of algorithm 2.1 : to find the function value at test point x_*, $f(x_*) = k_f(x_*, X)^T K_y^{-1} y$, where $K_y = K + \sigma_n^2 I$. Compute the lower triangular matrix L from the cholesky decomposition of $K_y = LL^T$ (Line 1). In line 2 compute the $\alpha = K_y^{-1} y = (LL^T)^{-1} y$. Predictive mean at test point x_* is given by $\bar{f}_*$ = $k_f(x_*, X)^T \alpha (Line 3)$. At line 5 we are computing Variance at x_*.



(a) The colored dotted curves are the functions sampled from the GP prior, without observing any data points so the uncertainty about the true function is uniform everywhere.



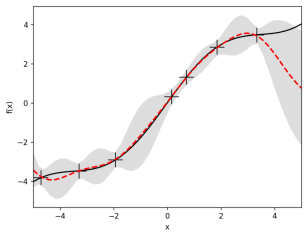
(b) The colored dotted curves are the functions sampled from the GP posterior after observing some data points (black crosses) and the solid black curve is the true function and the gray shaded area shows the uncertainty about it which is less near to observed data points and high elsewhere.

Figure 2.3: Samples from GP prior and GP Posterior.

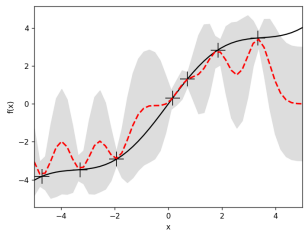
2.3.3 Effect of hyper-parameters on smoothness of the function

In section 2.3.1.1 we define the prior distribution over the f as Gaussian process with zero mean function and covariance function is defined by squared-exponential function (see eqn. 2.15) which is specified by parameters α and l that controls the smoothness of the function.

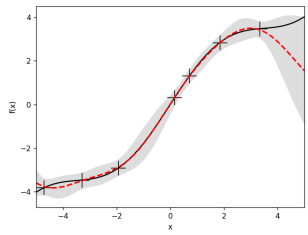
2.3.3.1 Effect of length scale (l) variation.



(a) $l^2 = 1$



(b) $l^2 = 0.2$



(c) $l^2 = 2.0$

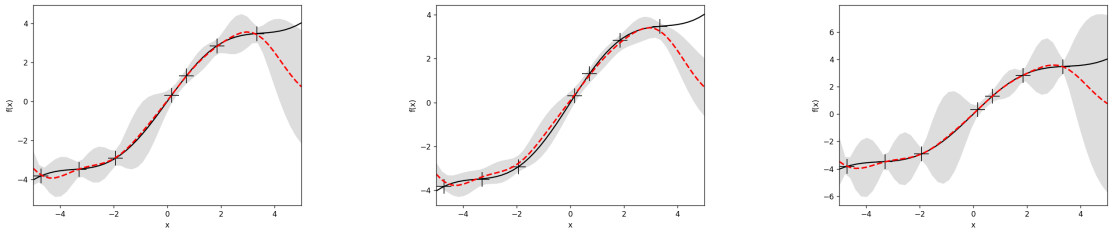
Figure 2.4: Effect of length scale variation.

In fig. 2.4, black crosses are the training points, gray shaded area is the confidence interval, the red doted line is the predicted mean function and the black solid line is the

actual function.

When the length scale l is small, the nearby data-points are also less correlated and the function varies drastically with x as we can see in fig. (2.4b) the red dot line changes drastically with x. And for large l the data-points are highly correlated so the function (red dot line in (2.4a), (2.4c))) changes smoothly with x. So we say l controls the horizontal length-scale.

2.3.3.2 Effect of signal variance α variation.



(a) $\sigma^2 = 1$ **(b)** $\sigma^2 = 0.2$ **(c)** $\alpha^2 = 5.0$

Figure 2.5: Effect of signal variance variation.

For small value of α the shaded gray area is also small (2.5b)), and for large value of α the shaded area is also large (2.5c)). So we say α controls the vertical length-scale.

2.3.4 Learning Hyper-parameters via Evidence Maximization

We have calculated the GP posterior over f for the setting where we assume that the hyper parameter θ is known. In this section we take hyper-parameter θ as a latent variable because we don't know their values a priori. We place a hyper-prior $p(\theta)$ on the hyper-parameter. So the two-level Bayesian inference is given by: [22]

Level-1 Inference :

The GP posterior on the function f for given hyper-parameter is

$$p(f|X, y, \theta) = \frac{p(y|X, f, \theta)p(f|\theta)}{p(y|X, \theta)} \tag{2.29}$$

Level-2 Inference :

The posterior on the hyper-parameters is

$$p(\theta|X, y) = \frac{p(y|X, \theta)p(\theta)}{p(y|X)} \tag{2.30}$$

The posterior distribution over θ is proportional to the marginal likelihood-

$$p(\theta|X, y) \propto p(y|X, \theta) \tag{2.31}$$

$$\log(p(y|X, \theta)) = \log \int p(y|f, X, \theta)p(f|\theta)df$$

$$= -\frac{1}{2}y^T(K_\theta + \sigma_\epsilon^2 I)^{-1}y - \frac{1}{2}|K_\theta + \sigma_\epsilon^2 I| - \frac{D}{2}\log(2\pi) \tag{2.32}$$

by maximizing the log marginal likelihood in equation (2.32) over the hyper-parameters, we find the best θ

$$\hat{\theta} \leftarrow \underset{\theta}{\operatorname{argmax}} \log p(y|X, \theta) \tag{2.33}$$

2.3.5 GP based Safe Exploration Algorithms

A GP is well suited for safety-critical problem settings in which the unknown safety functions are modelled using GP, and the uncertainty of these functions are given by the confidence interval (i.e. variance) of their respective Gps. The confidence interval are used to take an action which is safe with high probability. Previous work on safe exploration using GPs has addressed both stateless settings (Sui et al. 2015 [6]) and stateful settings (Turchetta, Berkenkamp, and Krause 2016 [23]; Wachi et al. 2018 [7]). In the following section we will explain the SAFEOPT (safe exploration in Bandit or stateless setting) and SafeMDP algorithm (safe exploration in MDP or stateful setting).

2.3.5.1 SAFEOPT Algorithm

To guarantee safety in bandit setting problems Sui et al. [6] proposed SAFEOPT algorithm. The objective is to optimize a unknown function $f(x)$ subject to safety constraint. For that they have used safe Bayesian optimization framework (see section 2.2.1). In their setting they considered the safety as, for chosen action/parameter x_t at time t the function value $f(x_t)$ should greater than a safety threshold h. Thus the authors approach the problem of estimating the maximum of the unknown function $f(x)$ by estimating $f(x)$ using Gaussian process (GP) regression. In GP regression, the unknown function is assumed to be modelled by a sample function from a GP prior [21]. The GP prior is completely characterized by its mean function $\mu(x)$ (without loss of generality $\mu(x) = 0$) and covariance function $k(x, x')$ where $x, x' \in \mathcal{X}$. At every time t, the Safe-Opt policy chooses a point x_t and receives an observation $y_t = f(x_t) + n_t$ where n_t is an independent sample from

Gaussian noise with mean 0 and variance σ^2. Based on y_t a posterior distribution for the unknown function can be derived. This posterior distribution is again Gaussian and characterized completely by a mean function $\mu_t(x)$ and covariance function $k_t(x, x')$. In order to satisfy the safety constraints, Safe-Opt computes upper and lower confidence bounds on the function using this posterior. The upper $u_t(x)$ and lower $l_t(x)$ confidence bounds are defined as

$$\begin{aligned} u_t(x) &= \mu_t(x) + \beta_t \sigma_t(x), \\ l_t(x) &= \mu_t(x) - \beta_t \sigma_t(x). \end{aligned}$$

And it uses the confidence interval $Q_t(x) = [l_t(x), u_t(x)]$ to estimate a safe set $S_t \subset \mathcal{X}$, from which agent select a next query point x_{t+1} for which the function value $f(x_{t+1})$ is greater than safety threshold with high probability. Safe Set S_t is defined as:

$$S_t \leftarrow \cup_{x \in S_{t-1}} \{x' \in \mathcal{X} | l_t(x) - Ld(x, x') \geq h\}$$

where L is lipschitz constant. SafeOpt assume at $t = 0$ it is provided with a initial safe seed S_0. Since the safe seed may not achieve the maxima of $f(x)$ it need to explore safely. Safe-Opt maintains a set $G_t \subseteq S_t$ of candidate decisions that, upon potentially repeated selection, have a chance to expand S_t. The set G_t is defined as

$$G_t = \{x \in S_t | \psi_t(x) > 0\} \tag{2.34}$$

where

$$\psi_t(x) = |\{x' \in \mathcal{X} \setminus S_t | u_t(x) - Ld(x, x') \geq h\}|.$$

In order to find the maxima SafeOpt maintain a another set $M_t \subseteq S_t$ of decisions that are potential maximizers of f.

$$M_t = \{x \in S_t | u_t(x) \geq \max_{x' \in S_t} l_t(x')\} \tag{2.35}$$

Safe-Opt policy then chooses points x_t according to

$$x_t = \operatorname*{argmax}_{x \in G_t \cup M_t} w_t(x). \tag{2.36}$$

where $w_t(x) = u_t(x) - l_t(x)$.

The pseudo code for SafeOpt algorithm is given below. For high level description of SafeOpt please refer the Sui et. al. paper [6].

Algorithm 2.2: SAFEOPT [6]

Input: Function domain $\mathcal{X}$, GP prior (μ, k), signal variance parameter σ_0, seed set S_0, safety threshold h,Lipschitz constant L.

$C_0(x) \leftarrow [h, \infty), \forall x \in S_0$

$C_0(x) \leftarrow \mathbb{R}, \forall x \in \mathcal{X} - S_0$

$Q_0(x) \leftarrow \mathbb{R}, \forall x \in \mathcal{X}$

for $t = 1, ..$ **do**

  $C_t(x) \leftarrow C_{t-1}(x) \cap Q_{t-1}(x)$

  $S_t \leftarrow \cup_{x \in S_{t-1}} \{x' \in \mathcal{X} | l_t(x) - Ld(x, x') \geq h\}$

  $G_t \leftarrow \{x \in S_t | g_t(x) > 0\}$

  $M_t \leftarrow \{x \in S_t | u_t(x) \geq \max_{x' \in S_t} l_t(x')\}$

  $x_t \leftarrow \text{argmax}_{x \in G_t \cup M_t}(w_t(x))$

  $y_t \leftarrow f(x_t) + n_t$

  Compute $Q_t(x), \forall x \in S_t$

  if $max_{x \in G_t \cup M_t} w_t(x) \leq \epsilon$ then Break

end

2.3.5.2 SafeMDP Algorithm

To guarantee safe exploration in finite MDP with known transition function and unknown safety function Krause et al. 2016 [4]) proposed the SAFEMDP algorithm. The safety function $f(.)$ is modelled using the Gaussian process regression. Then SafeMDP uses the posterior distribution over $f(.)$ to estimate S_t the set of states which are safe. And maintain a set $\hat{S}_t \subseteq S_t$ which consider those states from S_t that are reachable from and returnable to previous set $\hat{S}_{t-1}$. The Safe set S_t at time t is defined by:

$$S_t \leftarrow \{x \in \mathcal{S} | \exists s' \in \hat{S}_{t-1} : l_t(s) - Ld(s, s') \geq h\}$$

Even if the all states in set S_t satisfies the safety constraints with high probability but they might not be reachable from and returnable to previous safe set. So they defined reachable set as, a set of states which are reachable by agent from a given set $\mathcal{X}$ in one step-

$$R_{reach}(\mathcal{X}) = \mathcal{X} \cup \{s \in \mathcal{S} | \exists s' \in \mathcal{X}, a \in A : s = f(s', a)\}$$

And from the set $\mathcal{X}$ the set of states from which the agent can return to $\bar{\mathcal{X}}$.

$$R_{ret}(\mathcal{X}, \bar{\mathcal{X}}) = \bar{\mathcal{X}} \cup \{s \in \mathcal{X} | \exists a \in A : f(s, a) \in \bar{\mathcal{X}}\}$$

So, the returnable set is defined as a set of states that can reach $\bar{\mathcal{X}}$ through agent in multiple steps within $\mathcal{X}$ is $\bar{R}_{ret}(\mathcal{X}, \bar{\mathcal{X}})$. By using these definition set $\bar{S}_t$, is defined as

$$\hat{S}_t \leftarrow \{s \in S_t | s \in \mathcal{R}^{reach}(\hat{S}_{t-1}) \cap \bar{\mathcal{R}}^{ret}(S_t, \hat{S}_{t-1})\}$$

SafeMDP maintain another set $G_t \subseteq \bar{S}_t$, by choosing next visiting state from it could increase the safe region.

$$G_t \leftarrow \{s \in \hat{S}t | g_t(s) > 0\}$$

where

$$g_t(s) = |\{s' \in \mathcal{S}/S_t | u_t(s) - Ld(s, s') \geq h\}|$$

And the next state s_t is sampled according to-

$$s_t \leftarrow \underset{s \in G_t}{\operatorname{argmax}}(w_t(xs))$$

The pseudo code for SafeMDP algorithm is given below. For high level description of SafeMDP please refer the krause et. al. paper [4].

Algorithm 2.3: SAFEMDP [23]

Input: states $\mathcal{S}$, actions A, transition function $f(s, a)$, kernel $k(s, s')$, Safety threshold h, Lipschitz constant L, Safe seed S_0 .

$C_0(x) \leftarrow [h, \infty), \forall x \in S_0$
$C_0(x) \leftarrow \mathbb{R}, \forall x \in \mathcal{S} - S_0$
$Q_0(x) \leftarrow \mathbb{R}, \forall x \in \mathcal{S}$
for $t = 1, ..$ **do**
  $C_t(x) \leftarrow C_{t-1}(x) \cap Q_{t-1}(x)$
  $S_t \leftarrow \{x \in \mathcal{S} | \exists s' \in \hat{S}_{t-1} : l_t(s) - Ld(s, s') \geq h\}$
  $\hat{S}_t \leftarrow \{s \in S_t | s \in \mathcal{R}^{reach}(\hat{S}_{t-1}), s \in \bar{\mathcal{R}}^{ret}(S_t, \hat{S}_{t-1})\}$
  $G_t \leftarrow \{s \in \hat{S}t | g_t(s) > 0\}$
  $s_t \leftarrow \text{argmax}_{s \in G_t}(w_t(xs))$
  Safe Dijkstra in S_t from s_{t-1} to s_t
  Update GP with s_t and $y_t \leftarrow f(x_t) + n_t$
  Compute $Q_t(x), \forall s \in \hat{S}_t$
  if $G_t = \emptyset$ or $max_{s \in G_t \cup M_t} w_t(s) \leq \epsilon$ then Break
end

2.4 Reinforcement learning for Dynamically Varying Environments

In dynamically varying environment the model/context of the environment changes with time, which comprises of the state transition probability ($T(s_{t+1}|s_t, a_t)$) and reward function R. The environment dynamism can be continuous and discontinuous/switching. Figure 2.6 illustrate the time varying environment in which agent observe the reward value and next state according to the current context of the environment.

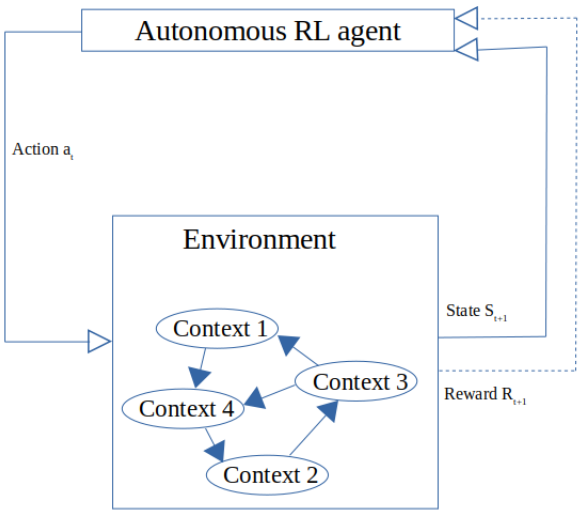


Figure 2.6: Reinforcement learning with dynamically varying environments.

Definition 2.6. [14] Formally the dynamically varying environment can be defined as a group of MDPs $\{M_k\}_{k\in\mathbb{N}^+}$, where $M_k =< S, A, T_k, R_k, \gamma >$ with usual notation defined in (2.1).

Change-point is the time instance at which the context of the environment changes. let say H_1 is the first time instance at which context changes means model of the environment changes from M_{k_0} to M_{k_1}. And the model of the environment changes from M_{k_1} to M_{k_2} at change-point H_2 and so on. where $H1 < H2$.... With respect to these model changes, the

non-stationary dynamics function for $t \geq 0$ will be

$$T(s_{t+1} = s' | s_t = s, a_t = a) = \begin{cases} T_{k_0}(s'|s,a) & \text{for } t \leq H_1 \\ T_{k_1}(s'|s,a) & \text{for } H_1 \leq t \leq H_2 \\ . \\ . \end{cases} \tag{2.37}$$

and the reward for $(s_t, a_t) = (s, a)$ will be

$$R(s,a) = \begin{cases} R_{k_0}(s,a) & \text{for } t \leq H_1 \\ R_{k_1}(s,a) & \text{for } H_1 \leq t \leq H_2 \\ . \\ . \end{cases} \tag{2.38}$$

if we incorporate the safety function ($g : S \times A \to \mathbb{R}$) in this setting, it's also changes with the change in the context, and given by

$$g(s,a) = \begin{cases} g_{k_0}(s,a) & \text{for } t \leq H_1 \\ g_{k_1}(s,a) & \text{for } H_1 \leq t \leq H_2 \\ . \\ . \end{cases} \tag{2.39}$$

According to how the context of the environment is changing, based on that dynamism of the environment is defined. If the context of the environment is changing at every time instance ($H_{i+1} = H_i + 1$) means that dynamism is continuous. If $H_{i+1} = H_i + N_i$ than the dynamism is discontinuous or switching type, where N_i is the number of time instance in between change-point H_{i+1} and H_i. If $H_1 = \infty$ then environment is stationary, not changing with time. Depending on the horizon H, the number of such changes will be either finite or infinite. And these change-points can be detected by using some change point detection algorithm like Bayesian online change point detection (BOCD) [24],etc. (if we have sufficient data points in between a change-point).

In this chapter we have seen the mathematical formulation of reinforcement learning problems and types of reinforcement learning algorithms to solve these problems and dis-

cussed about safe Bayesian optimization framework to optimize the unknown function safely by using Gaussian process to model the uncertainty of the function. In the following chapter we will use the dynamically varying environment definition to define our problem statement for safe exploration and optimization in switching/discontinuous dynamism environment.

Chapter 3

Safe Exploration in Switching Environment

In this chapter we consider the problem of safe exploration in switching environments for which we have proposed an Adaptive Safe-Explore policy that incorporates Bayesian inference and change detection-based scheme, which is an extension of SafeOpt algorithm [4] to switching environments but doesn't incorporate optimization. This policy balances two objectives: the first is to expand the safe region from the initial safe seed, and the second to detect the change point. We evaluate and compare its performance with baseline policies via simulation.

3.1 System Model and Problem Statement

We consider a discrete time model for the safe exploration problem with time indexed by $t \in \mathbb{Z}_+$. The objective of the safe exploration problem is to find the maximum safely reachable region from the initial safe seed S_0 (which is known) of a time varying unknown function $f(x,t)$ where $x \in \mathcal{X} \subseteq \mathbb{R}$. We define the true safe region at every time t by

$$S_{true}^t = \{x \in \mathcal{X} : f(x,t) \geq h\}, \tag{3.1}$$

where $h \in \mathbb{R}$ is chosen to be less than $\max_x f(x,t), \forall t$, is a safety threshold. We assume that $f(x,t)$ is a Lipschitz continuous function of x on a compact set $\mathcal{X} \subseteq \mathbb{R}$ for every t. The Lipschitz constant is assumed to be L. In this chapter we consider functions which model switching of the environment characteristics, the function $f(x,t)$ is assumed to be $f_1(x)$ until a time t_c (i.e., $f(x,t) = f_1(x), \forall t \in \{0,1,\ldots,t_c-1\}$ and

$f(x,t) = f_2(x), \forall t \in \{t_c, t_c+1, \cdots\}$. We consider the problem with only one change[1]. We also assume that $\forall x$, $|f_1(x) - f_2(x)| \leq B$, where B is a constant.

We note that S^t_{true} is unknown, since $f(x,t)$ is not known. However, we consider a setting in which, we estimate the S^t_{true} at each time t by observing the noisy value of the function at points $x_t \in \mathcal{X}$ which are chosen for every time t. The observed value at time t is denoted as y_t. At each time t the x_t is chosen in such a way that the probability $p(x_t \in S^t_{true}) \geq 1-\delta$,$\forall t$.

We define policy π as a sequence of x_t chosen by the agent. The sequence x_t could be chosen as a function of the history of choices, i.e., $(x_0, x_1, \ldots, x_{t-1})$ as well as the observations $(y_0, y_1, \ldots, y_{t-1})$. We first define the following metrics and formulate the problem in terms of them.

We define $C_\pi(t)$ as percentage coverage of true safe set S^t_{true} at time t from the initial safe seed S_0 by

$$C_\pi(t) = \frac{|S_t \cap S^t_{True}|}{|S^t_{True}|} \times 100$$

where S_t is estimated safe set at time t. We also define the cumulative unsafe evaluations over a horizon T as

$$U_\pi(T) = \sum_t \mathbb{I}\{f(x_t,t) < h\},$$

where $\mathbb{I}$ is the indicator function. Our objective is to find a policy π such that the $C_\pi(t)$ is maximized for all t subject to a safety constraint.

$$\max_\pi C_\pi(T) \text{ such that } \frac{U_\pi(T)}{T} \leq \delta. \tag{3.2}$$

where $\delta \in (0,1)$.

Ideally, we would want a policy π that achieves the above maximum for any choice of f_1 and f_2. We evaluate the performance of a policy by considering the average of the above metrics over choices of the function pair (f_1, f_2) and we also use another metric MAE (mean absolute error) over test points to compare between the policies.

[1]We note that the algorithms proposed in this chapter can be extended to the case of multiple changes without any change.

3.2 Adaptive Safe-Explore

In this section, we propose a heuristic policy (Adaptive Safe-Explore) that extends Safe-Opt [6] to adapt to the switches in $f(x,t)$. In contrast to Safe-Opt, Adaptive Safe-Explore is not optimizing the function but balances two objectives: the first is to expand the safe region from the initial safe seed, and the second to detect the change point. For the former objective it maintain a set $G_t \subseteq S_t$ of candidate decisions that, upon potentially repeated selection, have a chance to expand the safe set S_t . The definition of S_t and G_t are considered same as of SafeOpt (see section 2.3.5.1. The safe set S_t is defined by

$$S_t = \{x \in \mathcal{X} | l_t(x) \geq h\}$$

The expander set G_t is defined by

$$G_t = \{x \in S_t | \psi_t(x) > 0\}$$

where

$$\psi_t(x) = |\{x' \in \mathcal{X} \setminus S_t | u_t(x) - Ld(x,x') \geq h\}|.$$

In order to balance between the need to detect a change as well as maximizing the safely reachable region, we use an ϵ-greedy approach for Adaptive Safe-Explore. At every time t we choose

$$x_t = \begin{cases} \text{argmin}_{x \in S_t} \, w_t(x) & \text{with } \epsilon \text{ probability} \\ \text{argmax}_{x \in G_t} \, w_t(x) & \text{with } 1-\epsilon \text{ probability} \end{cases} \tag{3.3}$$

We note that an intuitive approach to adapting to the change in the function $f(.)$ is to detect whether a change has happened and then start exploring safely again. The challenges here are therefore to quickly detect the change as well as to find a safe seed. We note that the following is a candidate rule which can be used to detect a change. At each time step t we observe a noisy observation of function f, $y_t = f(t, x_t) + n_t$, from which we update the GP model of function, where x_t is sampled according to the above sampling criteria. To detect the change-point, at every time step we check the condition that the observed y_t is within the current confidence interval Q_t or not. If $y_t \in Q_t(x_t)$ then the algorithm decides that the function has not changed. If $y_t \notin Q_t(x_t)$ then Adaptive Safe-Explore declares that the change-point has detected and the function has changed.

Suppose a change has been detected, then we also need to estimate a new safe set S_t. If the y_t at the declared change time is safe, then the new safe seed is x_t itself. On the other

hand if $y_t < h$, then we initialize a safe-set estimate defined as

$$S_t = \{x \in \mathcal{X} | l_{t-1}(x) - B \geq h\}$$

Here we make use of the assumption that $|f_1(x) - f_2(x)| \leq B, \forall x$. It may turn out that $S_t = \emptyset$ or not. If $S_t \neq \emptyset$ then we have a safe set and we continue with Safe-Opt as before. However, if $S_t = \emptyset$ according to the above rule then we pick a x_{t+1} from $\operatorname{argmax} l_t(x)$ where we note that x_t although unsafe has been used to update the GP. At change-point we reinitialize the GP. The complete algorithm is given in Algorithm 3.1. We note that a few practically motivated modifications are introduced in Algorithm 3.1. First of all, consider that case in which we have some prior information about the inter-change duration, e.g., we know that the inter-change duration is at least some number of slots. Then, we note that the ϵ-greedy policy need not be used immediately after a change-point. We incorporate this by not using the above ϵ-greedy policy until a counter expires. In order to control the number of data points which need to be stored at every instant to update the GP posterior, we also introduce a data window. The data window size is incremented by one until a maximum window size (window_max) is reached. We note that an intuitive method to handle a time-variant environment is to consider data only in the immediate past. In order to evaluate how the Adaptive Safe-Explore policy compares with such a policy we also consider a FixedWindow Safe-Explore policy defined as follows. The FixedWindow-SafeOpt policy has a parameter *window*. For FixedWindow Safe-Explore, the GP model for $f(x,t)$ is updated at every time t using $(x_{t-window+1}, \ldots, x_t)$ and $(y_{t-window+1}, y_t)$. Then the sets $Q_t(x)$ and G_t are computed and x_{t+1} is chosen as

$$x_t = \operatorname*{argmax}_{s \in G_t} w_t(x).$$

3.3 Baseline Algorithms

In this section we discuss "Genie" algorithms which have access to *extra* or *side* information. Genie policies are not practically implementable since they assume the availability of such information, but are used to obtain insights into the best possible performance of policies in the case of switching functions.

Genie-CP-SS: This is a policy that has knowledge of the time t_c at which change point happens as well as the true safe seed set for f_2 after switching. We note that a function (f_1 or f_2) may have multiple disjoint intervals in the true safe set. We assume that a single

Algorithm 3.1: Adaptive Safe-Explore

Input: Function domain $\mathcal{X}$, GP prior (μ, k), signal variance parameter σ_0, seed set S_0, safety threshold h, $window\_min$, $window\_max$, $delaychangedetection\_flag$ = True, $changepoint\_flag$ =False, $changedetection\_delay$, $counter = 0$, $changepoint\_index = 1$, B, ϵ.

Initialize GP with safe seed points S_0 and compute Q_0
for $t = 1, ..$ **do**
  if *changepoint_flag = false* **then**
    $S_t \leftarrow \{x \in \mathcal{X} | l_t(x) \geq h\}$
    $G_t \leftarrow \{x \in S_t | \psi_t(x) > 0\}$
  else
    $S_t \leftarrow \{x \in \mathcal{X} | l_{t-1}(x) - B \geq h\}$
    $G_t \leftarrow \{x \in S_t | \psi_{t-1}(x) > 0\}$
    $changepoint\_flag = False$
  end
  if *delaychangedetection_flag = True* **then**
    $x_t \leftarrow \begin{cases} \text{argmax}_{x \in G_t}(w_t(x)) & \text{if } S_t \neq \emptyset \\ \text{argmax}_{x \in \mathcal{X}}(l_t(x)) & \text{if } S_t = \emptyset \end{cases}$
    $y_t \leftarrow f(x_t) + n_t$
    $window = window + window\_increment$
    $counter = counter + 1$
    if *window > window_max* **then**
      $window = window\_max$
    end
    if *counter = changedetection_delay* **then**
      $counter = 0,$
      $delaychangedetection\_flag = False.$
    end
  else
    $x_t \leftarrow \begin{cases} \text{argmin}_{x \in S_t} w_t(x) & \text{with } \epsilon \text{ probability} \\ \text{argmax}_{x \in G_t} w_t(x) & \text{with } 1 - \epsilon \text{ probability} \\ \text{argmax}_{x \in \mathcal{X}}(l_t(x)) & \text{if } S_t = \emptyset \end{cases}$
    $y_t \leftarrow f(x_t) + n_t$
    if $y_t < l_t(x_t)$ *or* $y_t > u_t(x_t)$ **then**
      $window = window\_min$
      $changepoint\_index = t$
      $delaychangedetection\_flag = True$
      $changepoint\_flag = True$
    else
      $window = window + window\_increment$
      if *window > window_max* **then**
        $window = window\_max$
      end
    end
  end
  $start = t - window$
  if *start < changepoint_index* **then**
    $start = changepoint\_index$
  end
  update GP using $(x_{start}, \ldots, x_t)$ and $(y_{start}, \ldots, y_t)$.
  Compute $Q_t(x), \forall x \in \mathcal{X}$
end

point from each of these disjoint intervals is given as part of the safe seed set to Genie-CP-SS. Then, for $t < t_c$ Genie-CP-SS uses Safe-Opt which is initialized with the safe seed, and for $t \geq t_c$ Safe-Opt can be re-initialized with the new safe seed and used. This policy chooses $x_t = \text{argmax}_{x \in G_t}(w_t(x))$. We note that since t_c as well as the safe-seed set is known, Genie-CP-SS should achieve the maximum possible value of $C_\pi(t)$ with the

minimum number of unsafe evaluations and provides an useful baseline for comparing with Adaptive Safe-Explore.

Genie-CP: This policy has side information only about the change point and not about the safe seed when a change happens. At t_c if $y_{t_c} \geq h$, then we re-initialize $S_{t_c} = x_{t_c}$ and then for $t > t_c$, the policy chooses x_t according to $x_t = \text{argmax}_{x \in G_t}(w_t(x))$. Otherwise, we choose x_t as $\text{argmax}_{x \in \mathcal{X}} l_t(x)$. The performance of Genie-CP would indicate the loss in performance due to the non-knowledge of safe seed set.

Genie-SS: This policy has side information of the safe seeds. However, it does not know the change point and uses a change point detection scheme as follows (this is similar to used by Adaptive Safe-Explore). At every time t, if the algorithm is allowed to do change detection (see discussion about incorporating prior information about change point times for Adaptive Safe-Explore), and if the current observation $y_t \notin Q_t(x_t)$ Genie-SS declares that a change has happened. Once a change is declared to have happened, the genie is given one safe seed each from each of the disjoint intervals which makes up the true safe set S_t^*. Similar to Genie-CP, the performance of Genie-SS would indicate the loss in performance due to non-knowledge of the change point.

3.4 Simulations and Performance Analysis

For comparing the performance of the algorithms proposed above, we consider one-dimensional functions $f_1(x)$ and $f_2(x)$ which are sampled from a GP prior. The safety threshold h is assumed to be 0 without loss of generality. The mean function $\mu(x)$ is assumed to be 0 and the covariance function is specified by a radial-basis function kernel (parametrized by variance of 4 and length scale of 1). When sampling f_2 we restrict to those samples such that $\forall x, |f_1(x) - f_2(x)| \leq B$, where B is fixed to be 2.5. We also sample f_1 and f_2 such that both $f_1(0) > 0$ and $f_2(0) > 0$ so that there is at least one point in the safe set for both functions. In our experiments, we consider one change point at $t_c = 100$. The time horizon is assumed to be 200. In the experiment shown below, we draw 20 samples of function pairs f_1 and f_2 and the performance metrics are averaged over these samples with standard deviation around the mean. For each pair of functions, the initial safe seed is the same for Adaptive-SafeOpt and Genie-CP; also the safe-seed set is the same for Genie-CP-SS and Genie-SS.

In Figure 3.1 we illustrate $C_\pi(t)$ for the different algorithms as a function of time. We plot the average of $C_\pi(t)$ over the 20 samples of (f_1, f_2) with the standard deviation around the mean. We observe that the policies Genie-CP-SS, and Genie-SS are able to cover the

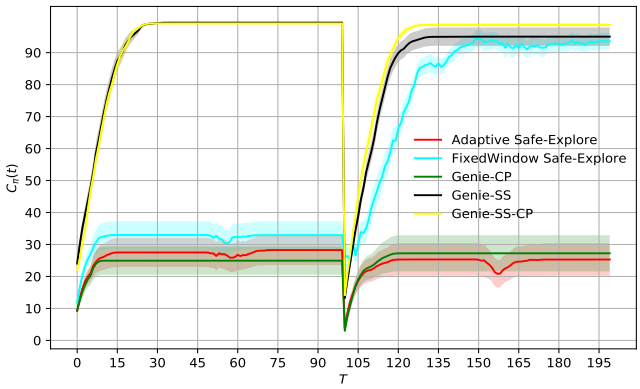
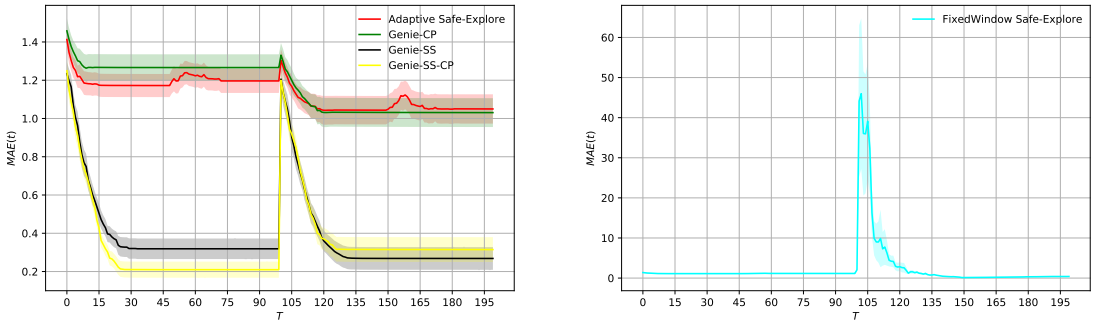


Figure 3.1: Comparison of $C_\pi(t)$ as a function of t for different algorithms. The change point $t_c = 100$. In this illustration, we assume that there is no observation noise.

approx whole true safe set S_{true} of functions f_1 and f_2, before and after the change point t_c. We also note that the proposed Adaptive Safe-Explore as well as Genie-CP are able to cover the 100% of a disjoint set of true safe set S_{true} of f_1 for which initial safe seed S_0 is given and also cover the 100% of a disjoint set of true safe set S_{true} of f_2 for which agent finds the a initial safe point. The FixedWindow-SafeOpt algorithm is able to cover the 100% of a disjoint set of true safe set S_{true} of f_1 for which initial safe seed S_0 is given and cover the approx whole true safe set S_{true} of function f_2 after the change-point.

In Figure 3.2 we illustrate mean absolute error *MAE(t)* between true function value and predicted value from GP at test points (which are uniformly sampled from the input space $\mathcal{X}$) for the different algorithms as a function of time. We plot the average of *MAE(t)* over the 20 samples of (f_1, f_2) with the standard deviation around the mean. We observe that the policies Genie-CP-SS, and Genie-SS are able to achieve the minimum *MAE* on test points for functions f_1 and $f2$ because they are able to explore the whole true safe region for f_1 and f_2 as seen in Figure 3.1. We also note that the *MAE(t)* for the proposed Adaptive Safe-Explore as well as Genie-CP is more compare to the policies Genie-CP-SS, and Genie-SS because they explore only a disjoint set of true safe set of f_1 and f_2. The *MAE(t)* for the FixedWindow Safe-Explore is same as Adaptive Safe-Explore and Genie-CP for function f_1 means before the change point t_c and after t_c for some time *MAE(t)* is large but after that it achieve the minimum *MAE* on test points for function f_2.

We illustrate the cumulative number of unsafe evaluations $U_\pi(T)$ for the different policies in Figure 3.3. Interestingly, we find that on average, $U_\pi(T)$ increases for those policies



(a) **(b)**

Figure 3.2: Comparison of *MAE(t)* as a function of T for different algorithms. The change point $t_c = 100$.

for which the side information about the safe set is not available.

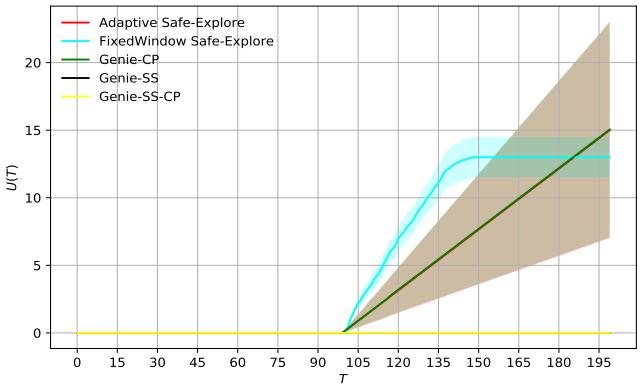
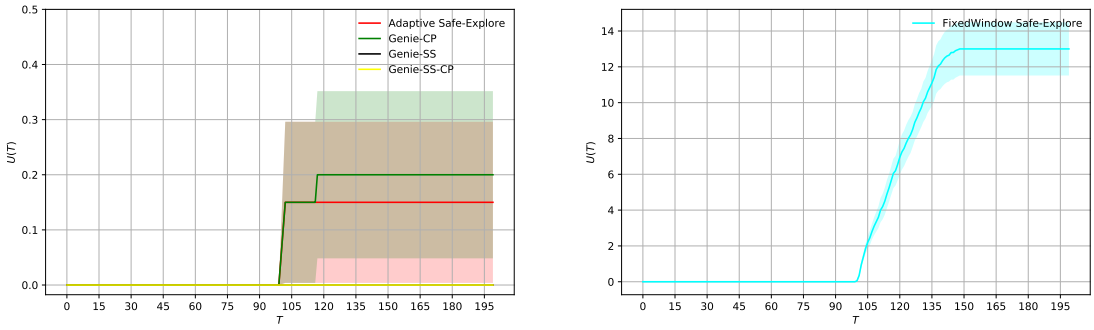


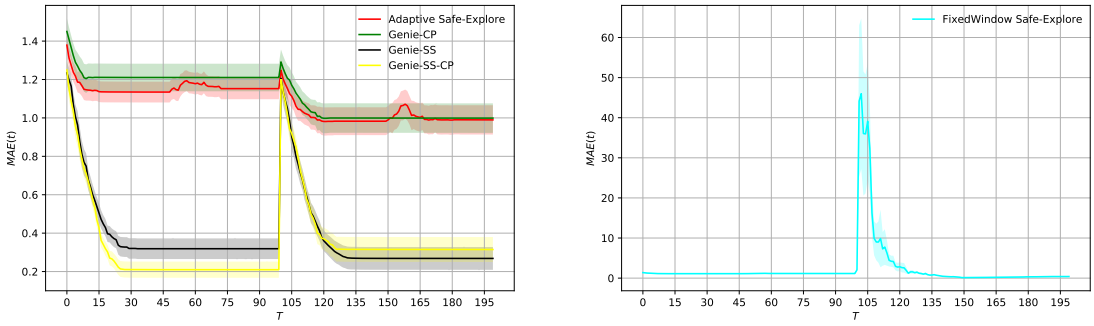
Figure 3.3: Comparison of the cumulative number of unsafe evaluations $U_\pi(T)$ as a function of T for different algorithms. The number of unsafe evaluations increase at the change point t_c.

This is found to happen because the proposed algorithms get attracted to local maxima, which are unsafe. Another set of experiments where the averaging is done by excluding such examples confirm this; see Figures 3.4 and 3.5.



(a) **(b)**

Figure 3.4: Comparison of the cumulative number of unsafe evaluations $U_\pi(T)$ as a function of T for different algorithms. The cases where the local maxima in unsafe region after change-point occurs are excluded in the average.



(a) **(b)**

Figure 3.5: Comparison of *MAE(t)* as a function of T for different algorithms. after excluding the cases in which the local maxima occurs. The change point $t_c = 100$.

3.5 Conclusions

we formulated the problem of safe exploration in switching environments. We proposed a heuristic algorithm called Adaptive Safe-Explore for this purpose and evaluated the performance of the algorithm via simulations. We observed that a major challenge for extending safe exploration to switching environments is finding a safe point to continue exploration when there is a change in the unknown function. We find that on average, $U_\pi(T)$ increases for those policies for which the side information about the safe set is not available after the change-point. This is found to happen because the proposed algorithms get attracted to local maxima, which are unsafe.

Chapter 4

Safe Sequential Optimization for Switching Environments

In this chapter we consider the problem of safe sequential optimization of a time varying unknown function for which we have proposed an Adaptive-SafeOpt policy that is an extension of the Adaptive Safe-Explore policy to also include optimization. This policy balances three objectives: the first is to expand the safe region, the second is to obtain the high reward region, and the third is to detect the change-point. We evaluate and compare its performance with baseline policies via simulation.

4.1 System Model and Problem Statement

We consider a discrete time model for the safe optimization problem with time indexed by $t \in \mathbb{Z}_+$. The objective of the safe optimization problem is to find the maximum of a time varying unknown function $f(x,t)$ where $x \in \mathcal{X} \subseteq \mathbb{R}$ subject to safety constraints. We assume that $f(x,t)$ is a Lipschitz continuous function of x on a compact set $\mathcal{X} \subseteq \mathbb{R}$ for every t. The Lipschitz constant is assumed to be L. In this chapter we consider functions which model switching of the environment characteristics, the function $f(x,t)$ is assumed to be $f_1(x)$ until a time t_c (i.e., $f(x,t) = f_1(x), \forall t \in \{0,1,\ldots,t_c-1\}$ and $f(x,t) = f_2(x), \forall t \in \{t_c, t_c+1, \cdots\}$. Here for simplification we consider the problem with only one change[1]. We also assume that $\forall x$, $|f_1(x) - f_2(x)| \leq B$, where B is positive.

We note that since the function is unknown, we optimize the function by observing the value of the function at points $x_t \in \mathcal{X}$ which are chosen for every time t. The observed value at time t is denoted as y_t. The safety constraint that we consider in this chapter is that the fraction of time $y_t \geq h$ is greater than or equal to $1-\delta$ where $\delta \in (0,1)$. Here we note

[1]We extended it to the case of multiple changes which is explained in section 4.5.2.

that $h \in \mathbb{R}$ is chosen to be less than $\max_x f(x,t), \forall t$. So ideally our problem is to obtain a sequence of points x_t^* such that $\forall t, f(x_t^*,t) = \max_{x\in\mathcal{X}} f(x,t)$, and $f(x_t^*) \geq h$. Since the functions are unknown we note that obtaining such as sequence of points would be a tough task. We therefore first define the following metrics and formulate a simpler problem.

A policy π is defined to be a sequence of x_t chosen by the optimizing agent. The sequence x_t could be chosen as a function of the history of choices, i.e., $(x_0, x_1, \ldots, x_{t-1})$ as well as the observations $(y_0, y_1, \ldots, y_{t-1})$. We denote by $\Delta_\pi(t)$ the gap between the maximum value of the function and the observed value y_t at x_t

$$\Delta_\pi(t) = \max_{x\in\mathcal{X}} f(x,t) - y_t$$

We also define the normalized cumulative regret over a horizon T as

$$R_\pi(T) = \frac{\sum_t \Delta_\pi(t)}{T}$$

The cumulative unsafe evaluations over a horizon T is defined as

$$U_\pi(T) = \sum_t \mathbb{I}\{f(x_t,t) < h\},$$

where $\mathbb{I}$ is the indicator function. Our objective is to find a policy π such that the regret is minimized for all T subject to a safety constraint.

$$\min_\pi R_\pi(T) \text{ such that } \frac{U_\pi(T)}{T} \leq \delta. \tag{4.1}$$

We define the true safe set $\mathcal{S}_t^*$ as $\mathcal{S}_t^* = \{x : f(x,t) \geq h\}$. We assume that at $t = 0$, an element of $\mathcal{S}_0^*$ called safe-seed is known to the algorithm[2].

Ideally, we would want a policy π that achieves the above minimum for any choice of f_1 and f_2. However, this may not be possible [25, Chapter 2]. In this chapter, we evaluate the performance of a policy by considering the average of the above metrics over choices of the function pair (f_1, f_2).

[2]For comparison purposes, we introduce genie policies. For these genie policies we note that at $t = 0$, one point each from the disjoint intervals that makes up $\mathcal{S}_0^*$ is given as input. This is called the safe-seed set.

4.2 Background: Safe-Opt Algorithm

In this section, we briefly review the Safe-Opt algorithm proposed by Sui et. al. [6] and introduce some essential notation since our algorithm is an extension of Safe-Opt to switching environments.

We note that in [6] the problem is similar to that in (4.1) but the function $f(x)$ is time-invariant. Thus the authors approach the problem of estimating the maximum of the unknown function $f(x)$ by estimating $f(x)$ using Gaussian process (GP) regression. In GP regression, the unknown function is assumed to be modelled by a sample function from a GP prior [21]. The GP prior is completely characterized by its mean function $\mu(x)$ (without loss of generality $\mu(x) = 0$) and covariance function $k(x, x')$ where $x, x' \in \mathcal{X}$. At every time t, the Safe-Opt policy chooses a point x_t and receives an observation $y_t = f(x_t) + n_t$ where n_t is an independent sample from Gaussian noise with mean 0 and variance σ^2. Based on y_t a posterior distribution for the unknown function can be derived. This posterior distribution is again Gaussian and characterized completely by a mean function $\mu_t(x)$ and covariance function $k_t(x, x')$. In order to satisfy the safety constraints, Safe-Opt computes upper and lower confidence bounds on the function using this posterior. The upper $u_t(x)$ and lower $l_t(x)$ confidence bounds are defined as

$$\begin{aligned} u_t(x) &= \mu_t(x) + \beta_t \sigma_t(x), \\ l_t(x) &= \mu_t(x) - \beta_t \sigma_t(x). \end{aligned}$$

By an appropriate choice of β_t and by choosing x_{t+1} such that $l_t(x_{t+1}) \geq h$, Safe-Opt is able to satisfy the safety constraint with high probability. We denote by $Q_t(x)$ the interval $[l_t(x), u_t(x)]$ as a function of $x \in \mathcal{X}$. We also denote the length of the confidence interval as $w_t(x) := u_t(x) - l_t(x)$. We note that on the basis of the confidence bounds an estimate $S_t \subseteq \mathcal{X}$ of the safe set can be maintained which is defined as

$$S_t = \{x \in \mathcal{X} | l_t(x) \geq h\} \tag{4.2}$$

We note that at $t = 0$ we are given a safe seed $\mathcal{S}_0^*$ (in the context of Safe-Opt $\mathcal{S}_t^*$ is time-invariant). Since the safe seed may not achieve the maxima of $f(x)$ we need to explore safely. Safe-Opt maintains a set $G_t \subseteq S_t$ of candidate decisions that, upon potentially repeated selection, have a chance to expand S_t. The set G_t is defined as

$$G_t = \{x \in S_t | \psi_t(x) > 0\} \tag{4.3}$$

where

$$\psi_t(x) = |\{x' \in \mathcal{X} \setminus S_t | u_t(x) - Ld(x, x') \geq h\}|.$$

We note that SafeOpt assumes Lipschitz continuity for the function $f(x)$ with Lipschitz constant L over $x \in \mathcal{X}$. We also note that in order to find the maxima, we need to consider candidate points which are chosen from a set $M_t \subseteq S_t$ of decisions that are potential maximizers of f.

$$M_t = \{x \in S_t | u_t(x) \geq \max_{x' \in S_t} l_t(x')\} \tag{4.4}$$

Safe-Opt policy then chooses points x_t according to

$$x_t = \operatorname*{argmax}_{x \in G_t \cup M_t} w_t(x). \tag{4.5}$$

4.3 Adaptive SafeOpt

In this section, we propose a heuristic policy (Adaptive SafeOpt) that extends Safe-Opt [6] to adapt to the switches in $f(x, t)$. In contrast to Safe-Opt, Adaptive SafeOpt balances three objectives: the first is to expand the safe region, the second is to obtain the high reward region, and the third is to detect the change-point.

We note that an intuitive approach to adapting to the change in the function $f(.)$ is to detect whether a change has happened and then restart the Safe-Opt algorithm. The challenges here are therefore to quickly detect the change as well as to find a safe seed for restarting Safe-Opt. We note that the following is a candidate rule which can be used to detect a change. At each time step t we observe a noisy observation of function f, $y_t = f(t, x_t) + n_t$, from which we update the GP model of function, where x_t is sampled according to the above sampling criteria. To detect the change-point, at every time step we check the condition that the observed y_t is within the current confidence interval Q_t or not. If $y_t \in Q_t(x_t)$ then the algorithm decides that the function has not changed. If $y_t \notin Q_t(x_t)$ then Adaptive-SafeOpt declares that the change-point has detected and the function has changed. In order to balance between the need to detect a change as well as maximize the function safely, we use an ϵ-greedy approach for Adaptive SafeOpt. At every time t we choose

$$x_t = \begin{cases} \operatorname{argmin}_{x \in S_t} w_t(x) & \text{with } \epsilon \text{ probability} \\ \operatorname{argmax}_{x \in G_t \cup M_t} w_t(x) & \text{with } 1 - \epsilon \text{ probability} \end{cases} \tag{4.6}$$

Suppose a change has been detected, then we also need to estimate a new safe set S_t. If the

y_t at the declared change time is safe, then the new safe seed is x_t itself. On the other hand if $y_t < h$, then we initialize a safe-set estimate defined as

$$S_t = \begin{cases} \{x \in \mathcal{X} | l_{t-1}(x) - B \geq h\} & \text{if being pessimistic} \\ \{x \in \mathcal{X} | u_{t-1}(x) + B \geq h\} & \text{if being optimistic} \end{cases} \tag{4.7}$$

Here we make use of the assumption that $|f_1(x) - f_2(x)| \leq B, \forall x$. So at that time t,we can assume $f_2(x) = l_{t-1}(x) - B$ to estimate the safe set S_t which is being pessimistic way or we can assume $f_2(x) = u_{t-1}(x) + B$ to estimate the safe set S_t which is being optimistic way. It may turn out that $S_t = \emptyset$ or not. If $S_t \neq \emptyset$ then we have a safe set and we continue with Safe-Opt as before. However, if $S_t = \emptyset$ according to the above rule then we pick a x_{t+1} from

$$x_{t+1} = \begin{cases} \operatorname{argmax} l_t(x) & \text{if being pessimistic} \\ \operatorname{argmax} u_t(x) & \text{if being optimistic} \end{cases} \tag{4.8}$$

where we note that x_t although unsafe has been used to update the GP. At change-point we reinitialize the GP which can be done in two ways. The one way is to use the previous function observations knowledge to define the mean function $\mu(x)$ by posterior mean function of the previous function and covariance function $k(x, x')$ by the posterior covariance function of the previous function plus a radial basis kernel. In second way we don't utilize the previous function observation information to re-initialize the GP so we take mean function $\mu(x) = 0$ and covariance function $k(x, x')$ is defined by a radial basis kernel.

The complete algorithm is given in Algorithm 4.1. We note that a few practically motivated modifications are introduced in Algorithm 4.1. First of all, consider that case in which we have some prior information about the inter-change duration, e.g., we know that the inter-change duration is at least some number of slots. Then, we note that the ϵ-greedy policy need not be used immediately after a change-point. We incorporate this by not using the above ϵ-greedy policy until a counter expires. In order to control the number of data points which need to be stored at every instant to update the GP posterior, we also introduce a data window. The data window size is incremented by one until a maximum window size (window_max) is reached.

We note that an intuitive method to handle a time-variant environment is to consider data only in the immediate past. In order to evaluate how the Adaptive-SafeOpt policy compares with such a policy we also consider a FixedWindow-SafeOpt policy defined as follows. The FixedWindow-SafeOpt policy has a parameter *window*. For FixedWindow-SafeOpt, the GP model for $f(x, t)$ is updated at every time t using $(x_{t-window+1}, \ldots, x_t)$

Algorithm 4.1: Adaptive-SafeOpt

Input: Function domain $\mathcal{X}$, GP prior (μ, k), signal variance parameter σ_0, seed set S_0, safety threshold h, $window\_min$, $window\_max$, $delaychangedetection\_flag$ = True, $changepoint\_flag$ =False, $changedetection\_delay$, $counter = 0$, $changepoint\_index = 1$, B, ϵ, $use\_modifiedkernel$=True or False.

Initialize GP with safe seed points S_0 and compute Q_0
$X = \{x | x \in S_0\}, Y = \{f(x) | x \in S_0\}$
for $t = 1, ..$ **do**
  if *changepoint_flag = false* **then**
    $S_t \leftarrow \{x \in \mathcal{X} | l_t(x) \geq h\}$
    $M_t \leftarrow \{x \in S_t | u_t(x) \geq \max_{x' \in S_t} l_t(x')\}$
    $G_t \leftarrow \{x \in S_t | \psi_t(x) > 0\}$
  else
    $S_t \leftarrow \begin{cases} \{x \in \mathcal{X} | l_{t-1}(x) - B \geq h\} & \text{if being pessimistic.} \\ \{x \in \mathcal{X} | u_{t-1}(x) + B \geq h\} & \text{if being optimistic} \end{cases}$
    $M_t \leftarrow \{x \in S_t | u_{t-1}(x) \geq \max_{x' \in S_t} l_{t-1}(x')\}$
    $G_t \leftarrow \{x \in S_t | \psi_{t-1}(x) > 0\}$
    $changepoint\_flag = False$
  end
  if *delaychangedetection_flag = True* **then**
    $x_t \leftarrow \begin{cases} \text{argmax}_{x \in G_t \cup M_t}(w_t(x)) & \text{if } S_t \neq \emptyset \\ \text{argmax}_{x \in \mathcal{X}}(l_t(x)) & \text{if } S_t = \emptyset \text{ and being pessimistic} \\ \text{argmax}_{x \in \mathcal{X}}(u_t(x)) & \text{if } S_t = \emptyset \text{ and being optimistic} \end{cases}$
    $y_t \leftarrow f(x_t) + n_t$
    $window = window + window\_increment$
    $counter = counter + 1$
    if *window > window_max* **then**
      $window = window\_max$
    end
    if *counter = changedetection_delay* **then**
      $counter = 0, \quad delaychangedetection\_flag = False.$
    end
  else
    $x_t \leftarrow \begin{cases} \text{argmin}_{x \in S_t} w_t(x) & \text{with } \epsilon \text{ probability} \\ \text{argmax}_{x \in G_t \cup M_t} w_t(x) & \text{with } 1 - \epsilon \text{ probability} \\ \text{argmax}_{x \in \mathcal{X}}(l_t(x)) & \text{if } S_t = \emptyset \text{ and being pessimistic} \\ \text{argmax}_{x \in \mathcal{X}}(u_t(x)) & \text{if } S_t = \emptyset \text{ and being optimistic} \end{cases}$
    $y_t \leftarrow f(x_t) + n_t$
    if $y_t < l_t(x_t)$ *or* $y_t > u_t(x_t)$ **then**
      $window = window\_min$
      $changepoint\_index = t$
      $delaychangedetection\_flag = True$
      $changepoint\_flag = True$
    else
      $window = window + window\_increment$
      if *window > window_max* **then**
        $window = window\_max$
      end
    end
  end
  $start = t - window$
  if *start < changepoint_index* **then**
    $start = changepoint\_index$
  end
  $\begin{cases} \text{update GP using } (x_{start}, \ldots, x_t) \text{ and } (y_{start}, \ldots, y_t). & \text{if changepoint\_flag = False} \\ \text{reinitialize GP with } (x_t, y_t) \text{ without modified kernel} & \text{if changepoint\_flag = True and } use\_modifiedkernel = False \\ \text{reinitialize GP with } (x_t, y_t) \text{ with modified kernel} & \text{if changepoint\_flag = True and } use\_modifiedkernel = True \end{cases}$
  Compute $Q_t(x), \forall x \in S_t$
end

and $(y_{t-window+1}, y_t)$. Then the sets $Q_t(x), G_t$, and M_t are computed and x_{t+1} is chosen as in SafeOpt (see Section 4.2).

4.4 Baseline Algorithms

In this section we discuss "Genie" algorithms which have access to *extra* or *side* information. Genie policies are not practically implementable since they assume the availability of such information, but are used to obtain insights into the best possible performance of policies in the case of switching functions.

Genie-CP-SS: This is a policy that has knowledge of the time t_c at which change point happens as well as the true safe seed set for f_2 after switching. We note that a function (f_1 or f_2) may have multiple disjoint intervals in the true safe set. We assume that a single point from each of these disjoint intervals is given as part of the safe seed set to Genie-CP-SS. Then, for $t < t_c$ Genie-CP-SS uses Safe-Opt which is initialized with the safe seed, and for $t \geq t_c$ Safe-Opt can be re-initialized with the new safe seed and used. Thus, the policy chooses $x_t = \text{argmax}_{x \in G_t \cup M_t}(w_t(x))$. We note that since t_c as well as the safe-seed set is known, Genie-CP-SS should achieve the minimum possible value of regret with the minimum number of unsafe evaluations and provides an useful baseline for comparing with Adaptive-SafeOpt.

Genie-CP: This policy has side information only about the change point and not about the safe seed when a change happens. At t_c if $y_{t_c} \geq h$, then we re-initialize $S_{t_c} = x_{t_c}$ and then for $t > t_c$, the policy chooses x_t according to $x_t = \text{argmax}_{x \in G_t \cup M_t}(w_t(x))$. Otherwise, we choose x_t as $\text{argmax}_{x \in D}\, l_t(x)$. The performance of Genie-CP would indicate the loss in performance due to the non-knowledge of safe seed set.

Genie-SS: This policy has side information of the safe seeds. However, it does not know the change point and uses a change point detection scheme as follows (this is similar to used by Adaptive Safe-Opt). At every time t, if the algorithm is allowed to do change detection (see discussion about incorporating prior information about change point times for Adaptive-SafeOpt), and if the current observation $y_t \notin Q_t(x_t)$ Genie-SS declares that a change has happened. Once a change is declared to have happened, the genie is given one safe seed each from each of the disjoint intervals which makes up the true safe set S_t^*. Similar to Genie-CP, the performance of Genie-SS would indicate the loss in performance due to non-knowledge of the change point.

GP-UCB-CP: Srinivas et al. [26] had proposed GP-UCB algorithm which does not consider the safety constraint. Here we consider GP-UCB endowed with side information

about when the change occurs so as to compare the regret with our policy. We note that GP-UCB-CP uses $x_t = \text{argmax}_{x \in D} u_t(x)$ and re-initializes the algorithm at the change point t_c. We note that GP-UCB-CP would inform us about the global maxima without any regard to the safety constraint. We expect that Genie-CP-SS and GP-UCB-CP would perform similarly as the global maxima is also safe; however we note that the exploration methodology for both of these policies are different.

4.5 Simulations and Performance Analysis

4.5.1 Experiment setup for single change-point case

For comparing the performance of the algorithms proposed above, we consider one-dimensional functions $f_1(x)$ and $f_2(x)$ which are sampled from a GP prior. The safety threshold h is assumed to be 0 without loss of generality. The mean function $\mu(x)$ is assumed to be 0 and the covariance function is specified by a radial-basis function kernel (parametrized by variance of 2 and length scale of 1). When sampling f_2 we restrict to those samples such that $\forall x, |f_1(x) - f_2(x)| \leq B$, where B is fixed to be 1. We also sample f_1 and f_2 such that both $f_1(0) > 0$ and $f_2(0) > 0$ so that there is at least one point in the safe set for both functions. In our experiments, we consider one change point at $t_c = 150$. The time horizon is assumed to be 300. In the experiment shown below, we draw 500 samples of function pairs f_1 and f_2. For each pair of functions, the initial safe seed is the same for Adaptive-SafeOpt and Genie-CP; also the safe-seed set is the same for Genie-CP-SS and Genie-SS. In Figure 4.1 we illustrate $\Delta_\pi(t)$ for the different algorithms as a function of time. We plot the average of $\Delta_\pi(t)$ over the 500 samples of (f_1, f_2) with the standard deviation around the mean. We observe that GP-UCB-CP, Genie-CP-SS, and Genie-SS converge to the minimum possible $\Delta_\pi(t)$ after an initial exploration phase. We also note that the proposed Adaptive-SafeOpt as well as Genie-CP converges but since the safe set that they explore is limited in size, the convergence is to a local maxima. The FixedWindow-SafeOpt algorithm is observed not to converge. We also consider a case with observation noise variance of 0.2 in Figure 4.2. We observe that in this case Genie-CP-SS and Genie-SS are limited by their ability to explore the safe sets completely and have larger gaps from the optimal value, in comparison to the GP-UCB-CP algorithm which is able to achieve $\Delta_\pi(T) = 0$ on average. Again, the proposed Adaptive-SafeOpt converges to the local maxima corresponding to the safe seed that it finds, which is shown by the match with the Genie-CP policy.

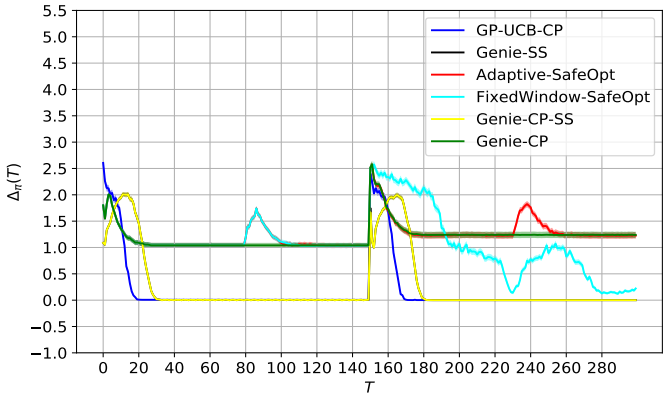


Figure 4.1: Comparison of $\Delta_\pi(T)$ as a function of T for different algorithms. The change point $t_c = 150$. In this illustration, we assume that there is no observation noise.

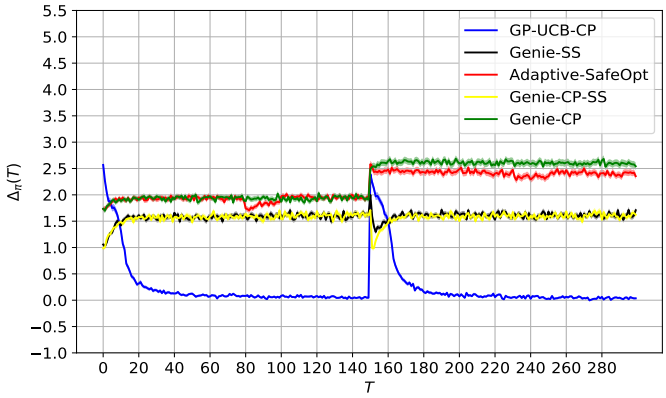


Figure 4.2: Comparison of $\Delta_\pi(T)$ as a function of T for different algorithms. The change point $t_c = 150$. Observations are noisy with noise variance of 0.2.

The time normalized regret $R_\pi(T)$ for these policies without and with observation noise variance are shown in Figures 4.3 and 4.4. We observe that Genie-CP-SS and Genie-SS overlap with each other due to zero observation noise variance. Also, in this case, Genie-SS able to detect the change point accurately at t_c without any delay. We illustrate the cumulative number of unsafe evaluations $U_\pi(T)$ for the different policies in Figure 4.5. Interestingly, we find that on average, $U_\pi(T)$ increases for those policies for which the side information about the safe set is not available. This is found to happen because the proposed algorithms get attracted to local maxima, which are unsafe. Another set of experiments where the averaging is done by excluding such examples confirm this; see Figures 4.6 and 4.7. We then observe that GP-UCB has traded off unsafe evaluations with achieving

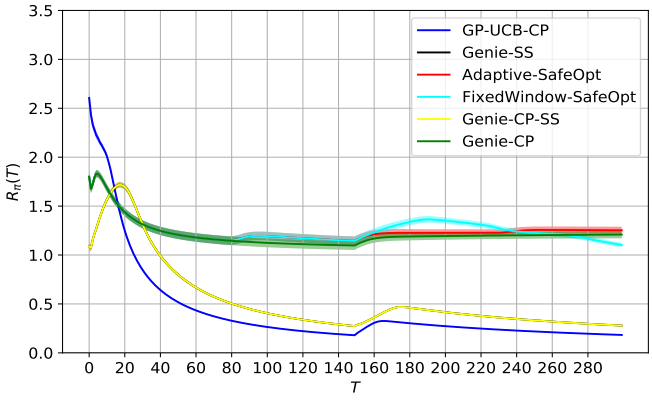


Figure 4.3: Comparison of $R_\pi(T)$ as a function of T for the different algorithms. The change point $t_c = 150$.

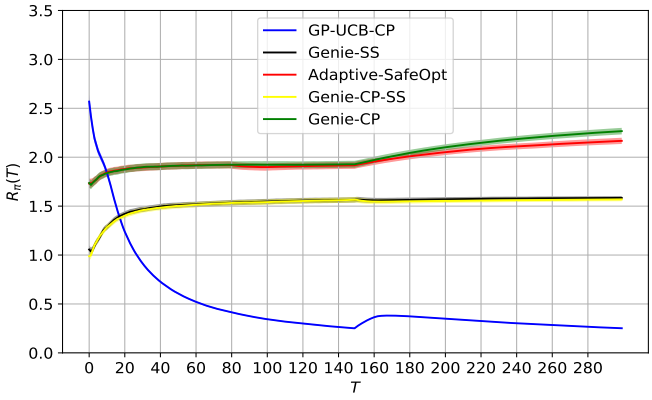


Figure 4.4: Comparison of $R_\pi(T)$ as a function of T for the different algorithms. The change point $t_c = 150$. Observations are noisy with noise variance of 0.2.

the global maxima. We note that the performance of Adaptive-SafeOpt depends critically on the safe-set initialization at the change point. Although Adaptive-SafeOpt is able to converge to a local safe maxima, it could still be larger than the global safe maxima which is achieved by Genie-CP-SS or Genie-SS. It has also been observed that instead of choosing x_{t+1} as $\operatorname{argmax} l_t(x)$ when $S_t = \emptyset$, the GP-UCB choice of $x_{t+1} = \operatorname{argmax} u_t(x)$ leads to lower $R_\pi(T)$.

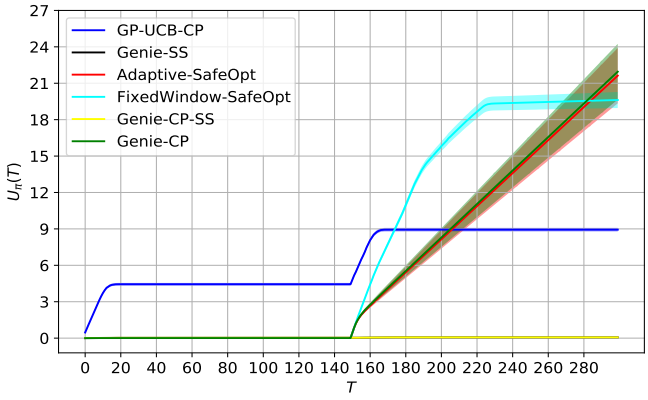


Figure 4.5: Comparison of the cumulative number of unsafe evaluations $U_\pi(T)$ as a function of T for different algorithms. The number of unsafe evaluations increase at the change point t_c.

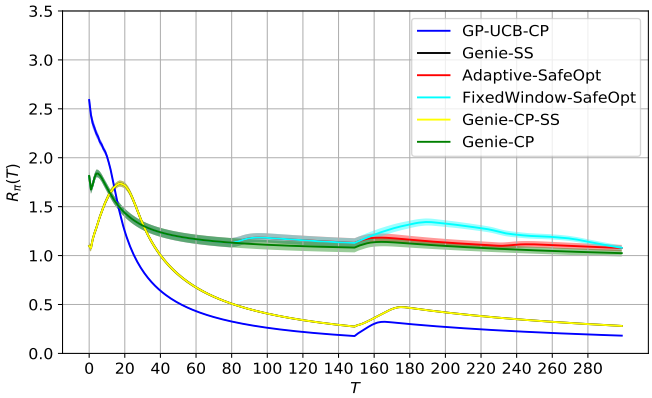


Figure 4.6: Comparison of $R_\pi(T)$ as a function of T for the different algorithms after excluding the cases in which the local maxima occurs. The change point $t_c = 150$.

4.5.2 Experiment setup for multiple change-points case

We extended the Adaptive-SafeOpt algorithm to multiple change-points case. In multiple change-point case the function $f(x,t)$ switches by

$$f(x,t) = \begin{cases} f(x,t-1) & \text{with } 1-p \text{ probability} \\ f_{new}(x) & \text{with } p \text{ probability} \end{cases}$$

So at each time t with p probability function changes to $f_{new}(x)$ otherwise remain same. In this experiment we assumed $p = 0.01$ and the functions $f(x,t-1)$ and $f_{new}(x)$ are

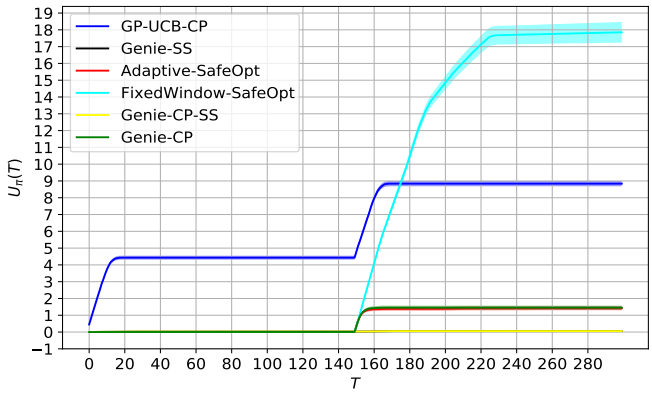
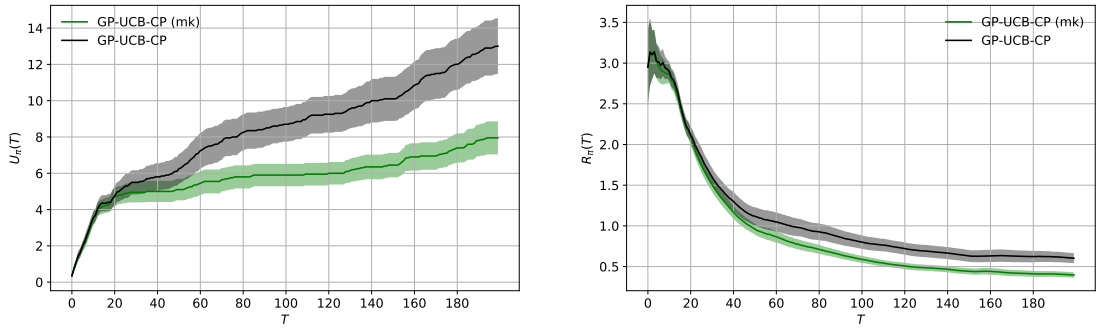


Figure 4.7: Comparison of the cumulative number of unsafe evaluations $U_\pi(T)$ as a function of T for different algorithms. The cases where the local maxima occurs are excluded in the average.

one-dimensional and sampled from GP prior which having zero mean function and covariance function is specified by a radial-basis function kernel (parametrized by variance of 4 and length scale of 1), with boundedness condition $|f(x, t-1) - f_{new}(x)| \leq B$, where B is fixed to be 2.5. The safety threshold h is assumed to be 0 without loss of generality. The time horizon is assumed to be 200 and the observation noise variance is 0.0. In the experiment shown below, the performance metrics are averaged over 20 run. For each run, the initial safe seed is the same for Adaptive-SafeOpt and Genie-CP.

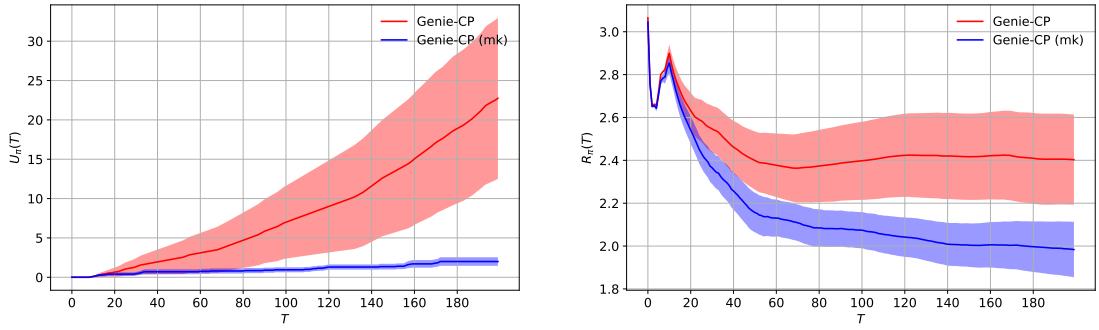
In Figure 4.8 We plot the average of $R_\pi(T)$ and $U_\pi(T)$ over the 20 run with the standard deviation around the mean as a function of T. We observe that GP-UCB-CP with modified kernel perform better compare to GP-UCB-CP without modified kernel and gives less unsafe evaluation and normalized regret. The same results (see Figures 4.9 and 4.10) are observed for other policies also, means if the policy is using modified kernel to reinitialize the GP at the change-point the unsafe evaluation and normalized regret is less compare to, if they don't use modified kernel to reinitialize the GP at the change-point.

In Figure 4.11 We plot the average of $R_\pi(T)$ and $U_\pi(T)$ over the 20 run with the standard deviation around the mean as a function of T. We observe that GP-UCB-CP converge to the minimum possible $R_\pi(t)$ after an initial exploration phase. We also note that the proposed Adaptive-SafeOpt as well as Genie-CP converges but since the safe set that they explore is limited in size, the convergence is to a local maxima. We then observe that GP-UCB-CP has traded off unsafe evaluations with achieving the global maxima. The



(a) **(b)**

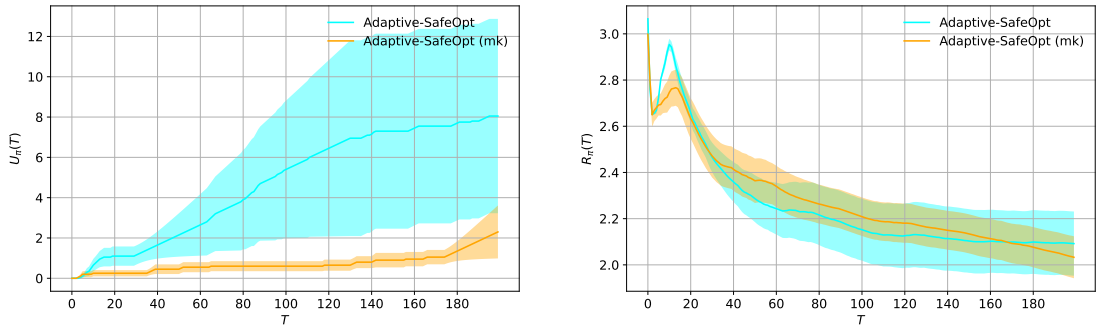
Figure 4.8: Comparison of $U_\pi(T)$ (4.8a) and $R_\pi(T)$ (4.8b) as a function of T for GP-UCB-CP policy for with or without use of modified kernel at the time of change.



(a) **(b)**

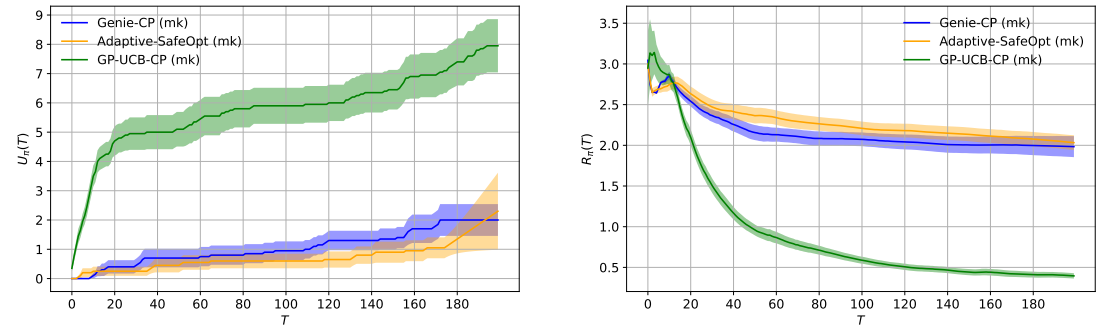
Figure 4.9: Comparison of $U_\pi(T)$ (4.9a) and $R_\pi(T)$ (4.9b) as a function of T for Genie-CP policy for with or without use of modified kernel at the time of change.

proposed Adaptive-SafeOpt as well as Genie-CP have less unsafe evaluation compare to GP-UCB-CP policy. We note that these results are same for both single change-point case and multiple change-points case.



(a) **(b)**

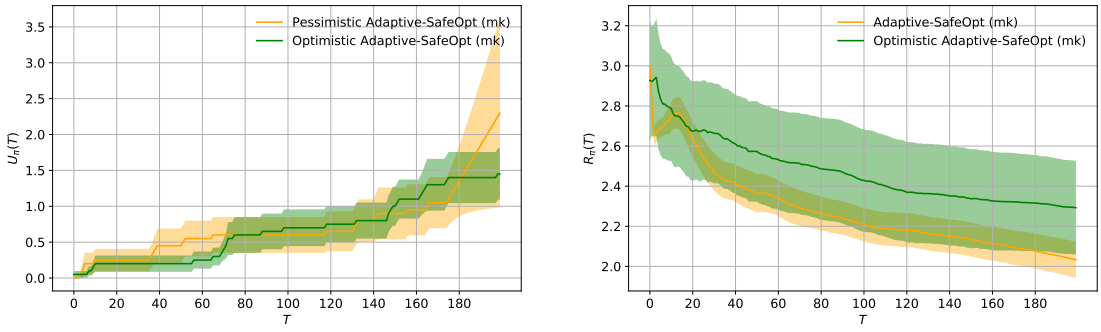
Figure 4.10: Comparison of $U_\pi(T)$ (4.9a) and $R_\pi(T)$ (4.9b) as a function of T for Adaptive-SafeOpt policy for with or without use of modified kernel at the time of change.



(a) **(b)**

Figure 4.11: Comparison of $U_\pi(T)$ (4.11a) and $R_\pi(T)$ (4.11b) as a function of T for different policies with use of modified kernel at the time of change.

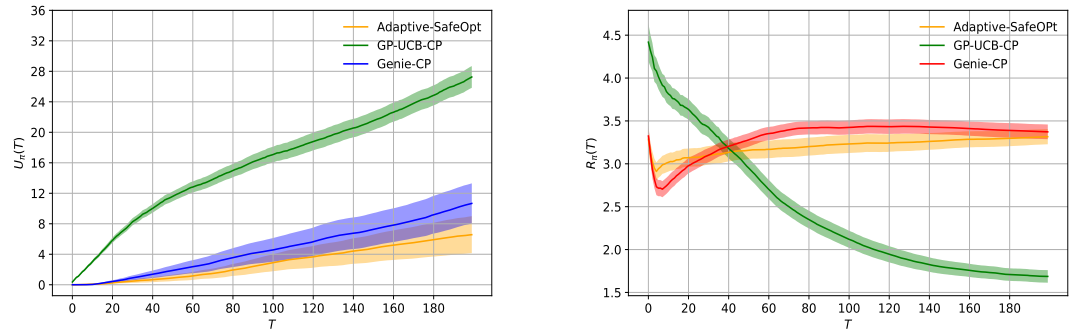
In Figure 4.12 We plot the average of $R_\pi(T)$ and $U_\pi(T)$ over the 20 run with the standard deviation around the mean as a function of T. We observe that Pessimistic Adaptive-SafeOpt and Optimistic Adaptive-SafeOpt policies achieve the same regret but the unsafe evaluation increases for Pessimistic Adaptive-SafeOpt because for some case in stuck to local maxima in unsafe region. In Figure 4.13 We plot the average of $R_\pi(T)$ and $U_\pi(T)$



(a) **(b)**

Figure 4.12: Comparison of $U_\pi(T)$ (4.12a) and $R_\pi(T)$ (4.12b) as a function of T for Pessimistic and Optimistic Adaptive-SafeOpt policies with use of modified kernel at the time of change.

over the 20 run with the standard deviation around the mean as a function of T for different algorithm. In this we have considered $x \in \mathbb{R}^2$ along with multiple change-points case. So we extend the Adaptive-SafeOpt for multi-dimension function also. We not that below result is same as $1d$ case with multiple change-points.



(a) **(b)**

Figure 4.13: [$2d$-case] Comparison of $U_\pi(T)$ (4.13a) and $R_\pi(T)$ (4.13b) as a function of T for different policies.

4.6 Conclusions

In this chapter, we considered the problem of safe optimization of a time varying unknown function using the framework of Bayesian optimization and change point detection. We proposed a heuristic algorithm called Adaptive-SafeOpt and its variants for this purpose and evaluated the performance of the algorithm via simulations. We observed that a major challenge for extending safe optimization to time varying functions is finding a safe point to continue exploration when there is a change in the unknown function. We find that on average, $U_\pi(T)$ increases for those policies for which the side information about the safe set is not available after the change-point. This is found to happen because the proposed algorithms get attracted to local maxima, which are unsafe. The variant Optimistic Adaptive-SafeOpt algorithm doesn't have this issue so it has less unsafe evaluation compare to variant Pessimistic Adaptive-SafeOpt. And these two perform same if we exclude the outlier cases. We note that the proposed Adaptive-SafeOpt converges but since the safe set that they explore is limited in size, the convergence is to a local maxima. We also note that Adaptive-SafeOpt with modified covariance and mean function perform better compare to Adaptive-SafeOpt without modified covariance and mean function.

Chapter 5

Safe Thompson Sampling for Linear Bandits

In the previous chapter 4, we have considered the objective/reward function as a non-linear and time-variant function, and our objective is to optimize that reward function at every time such that the safety constants are also satisfied with high probability (The safety constraint that we considered in the previous chapter is that the function value should be non-negative for the chosen action x at every time). In this chapter, we consider a simplified problem in which the reward function is linear and time time-invariant and the objective is to maximize this reward function subject to linear safety constraints. This problem setup is known as safe optimization in linear bandit setting. We proposed a Thompson sampling based algorithm for this problem which has less parameter (if we exclude prior parameter than there is only scheduling parameter is to tune.) to tune compare to the Adaptive SafeOpt algorithm. There are many real world applications such as online advertising, recommender system that can be formulate as linear bandit problem.

5.1 System Model & Problem Statement

We consider a discrete-time model where time is indexed by $t \in \mathbb{Z}^+$. The expected reward value at time t for chosen action x_t is given by

$$r_t = x_t^\top \theta_* + \epsilon_t, \tag{5.1}$$

where action $x_t \in \mathcal{X} \subseteq \mathbb{R}^d$, parameter $\theta_* \in \mathbb{R}^d$ is fixed and not known to the agent and noise $\epsilon_t \sim \mathcal{N}(0, \sigma_r^2)$. And the agent has to satisfy the safety constraint which also linear

and given by

$$x^\top \mu_* \leq C, \tag{5.2}$$

where parameter $\mu_* \in \mathbb{R}^d$ is fixed but not known to the agent and parameter C is a positive constant which is known to the agent. We define a true safe set S_{true} which satisfy the above constraint (5.2) by-

$$S_{true} = \{x \in \mathcal{X} : x^\top \mu_* \leq C\} \tag{5.3}$$

We note that S_{true} is unknown since μ_* is not known the agent. However we consider a setting in which we estimate the set S_{true} at each time t by observing the noisy evaluation of the safety constraint:

$$w_t = x^\top \mu_* + \zeta_t, \tag{5.4}$$

where noise $\zeta_t \sim \mathcal{N}(0, \sigma_w^2)$. At each time t the x_t is chosen in such a way that the probability $p(x_t \in S_{true}) \geq 1 - \delta$.

A policy π is defined to be a sequence of x_t chosen by the agent. We first define the following metrics and formulate the problem in terms of them. We define $R_\pi^c(T)$ the cumulative regret over a horizon T as,

$$R_\pi^c(T) = \sum_{t=1}^{T} x_*^\top \theta_* - x_t^\top \theta_*$$

where x_* is the optimal safe action i.e.

$$x_* = \underset{x \in S_{true}}{\operatorname{argmax}}\, x^\top \theta_*.$$

We also define the cumulative unsafe evaluations over a horizon T as

$$U_\pi(T) = \sum_{t=1}^{T} \mathbb{I}\left\{x^\top \mu_* > C\right\},$$

where $\mathbb{I}$ is the indicator function. The agent's objective is to find a policy π such that the $R_\pi^c(T)$ is minimized subject to a safety constraint.

$$\min R_\pi^c(T) \text{ such that } \frac{U_\pi(T)}{T} \leq \delta, \tag{5.5}$$

where $\delta \in [0, 1)$.

5.2 Bayesian Linear Regression

In the system model we have assumed the reward and safety function are linear and their weight vectors θ_* and μ_* are not known. We use Bayesian linear regression [21, Chapter 2] to estimate these weight vectors. Bayesian linear regression gives the posterior distribution over the weight vectors which captures the uncertainty about the weight vectors.
To understand the Bayesian linear regression, let us consider the standard linear model

$$f(x) = x^\top w, \qquad y = f(x) + \epsilon$$

where $x \in \mathbb{R}^d$ is the feature vector, $w \in \mathbb{R}^d$ is the unknown weight vector, f is the function value and y is noisy observation of the function value and noise ϵ is sampled from iid Gaussian distribution:

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

In Bayesian setting we require a prior information on the weight parameters which express our belief about the weight parameters before observing any data-points. We assume the prior distribution is multivariate normal distribution with zero mean and covariance matrix is Σ_p.

$$w \sim \mathcal{N}(0, \Sigma_p) \tag{5.6}$$

After n observations the posterior distribution on the w is given by the bayesian inference

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \qquad p(w|X, \mathbf{y}) = \frac{p(\mathbf{y}|X, w)p(w)}{p(\mathbf{y}|X)} \tag{5.7}$$

where $X \in \mathbb{R}^{n \times d}$ is the design matrix which contain the n input vectors and $y \in \mathbb{R}^{n \times 1}$ is the target vector.
The likelihood

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \prod_{i=1}^{n} p(y_i|x_i, w) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} \exp(-\frac{(y_i - x_i^\top w)^2}{2\sigma_n^2})$$

$$= \frac{1}{(2\pi\sigma_n^2)^{n/2}} \exp(-\frac{1}{2\sigma_n^2}|\mathbf{y} - X^\top w|^2) = \mathcal{N}(X^\top w, \sigma_n^2 I) \tag{5.8}$$

The posterior is proportional to the likelihood and the prior (see eqn. 5.7), by eqn (5.8 & 5.6)

$$p(w|X,\mathbf{y}) \propto \exp(\frac{-1}{2\sigma_n^2}(\mathbf{y}-X^\top w)^\top(\mathbf{y}-X^\top w))\exp(-\frac{1}{2}w^\top\Sigma_p^{-1}w)$$

$$\propto \exp(-\frac{1}{2}(w-\bar{w})^\top(\frac{1}{\sigma_n^2}XX^\top+\Sigma_p^{-1})(w-\bar{w}))$$

where $\bar{w} = \sigma_n^{-2}A^{-1}X\mathbf{y}$ and $A = \sigma_n^{-2}XX^\top + \Sigma_p^{-1}$, so the posterior is also a Gaussian distribution with mean $\bar{w}$ and covariance matrix A^{-1}.

$$p(w|X,\mathbf{y}) \sim \mathcal{N}(\bar{w}, A^{-1})$$

5.3 Bayesian Safe-LTS Algorithm

Bayesian Safe-LTS (BaySafeLTS) algorithm is modified version of the Safe-LTS algorithm proposed by the A Moradipari et al. [27]. They used regularized least square regression to estimate the parameters θ_* and μ_* instead of that we used Bayesian linear regression which gives posterior belief over the parameter. For exploration purpose at each time t they perturb the estimated $\hat{\theta}$ with a vector η_t which follow the special distribution (see the detail in the paper [27]) instead of that at each time we sample θ from the posterior distribution $P_{\theta,t}$. To estimate the safe set at each time they define confidence interval $\mathcal{C}_t$ on parameter μ, so the conservative approximation of true safe set is define by

$$S_t = \{x \in \mathcal{X} : x^\top v \leq C, \forall v \in \mathcal{C}_t\}$$

Instead of that we define the safe set by

Algorithm 5.1: BaySafeLTS

Input: $T, x_0, P_{\theta,0}, P_{\mu,0}, C$
for $t \leftarrow 0$ **to** T **do**
  $r_t = x_t^\top\theta_* + \epsilon_t$
  $w_t = x_t^\top\mu_* + \zeta_t$
  $P_{\theta,t+1}, P_{\mu,t+1} \leftarrow$ update the posterior for θ&μ respectively ▷ see section 5.2
  $\tilde{\mu} \sim P_{\mu,t+1}$
  $S_t = \{x \in \mathcal{X} : x^\top\tilde{\mu} \leq C - \alpha_t\}$
  $\tilde{\theta} \sim P_{\theta,t+1}$
  $x_* = \text{argmax}_{x\in S_t}\, x^\top\tilde{\theta}$
  $x_{t+1} = x_*$
end

$$S_t = \{x \in \mathcal{X} : x^\top\tilde{\mu} \leq C - \alpha_t\}$$

where $\tilde{\mu}$ is sampled from posterior distribution $P_{\mu,t}$ and α_t is a scheduling on C at time t. We define three type of scheduling on C, The **Type**-1 is max-variance scheduling which is given by

$$\alpha_t = \beta \times d \times (\max_{k=1,..d} \sigma^t_{\mu^k})$$

The **Type**-2 is max-variance square-root scheduling which is given by

$$\alpha_t = \frac{\beta \times d \times (\max_{k=1,..d} \sigma^t_{\mu^k})}{\sqrt{t+1}}$$

and the **Type**-3 is max-variance square-root log scheduling which is given by

$$\alpha_t = \frac{\beta \times d \times (\max_{k=1,..d} \sigma^t_{\mu^k})}{\sqrt{\log \frac{(t+1)}{\gamma}}}$$

where β and γ are positive parameter and $\sigma^t_{\mu^k}$ is the standard deviation of the k^{th} element of the parameter μ at time t.

5.4 Baseline Algorithm

In this section we discuss "Genie" algorithm which has access to *extra* or *side* information. Genie policies are not practically implementable since they assume the availability of such information, but are used to obtain insights into the best possible performance of policies.

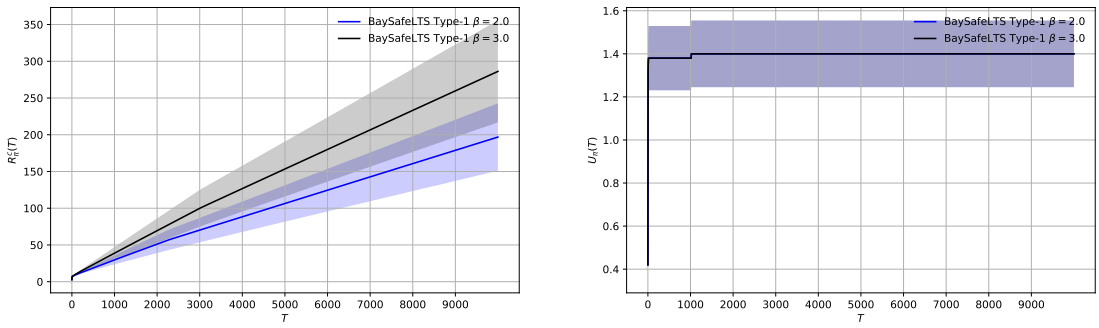
Genie-True-SS: This is a policy that has knowledge of the parameter μ_*. We note that since parameter μ_* is known so agent know the actual true safe set S_{true}, Genie-True-SS should achieve the minimum possible value of regret with no unsafe evaluations and provides an useful baseline for comparing with BaySafeLTS.

5.5 Simulation and Performance Analysis

For comparing the performance of the algorithms proposed above, we consider the reward function and safety constrain input domain are in $\mathbb{R}^4$, means $x \in \mathbb{R}^4$. Time horizon T is assumed to be $10,000$. The parameters θ_* and μ_* are sampled from $\mathcal{N}(0, I_4)$, C is sampled uniformly from $[0, 1]$, the measurement noise ϵ_t and ζ_t are sampled from $\mathcal{N}(0, 10^{-4})$.

In the experiment shown below, we draw 50 samples of θ_*, μ_* and C and the performance metrics ($R^c_\pi(T)$ and $U_\pi(T)$) are averaged over these samples.

In Figure 5.1 we illustrate $R^c_\pi(T)$ (5.1a) and $U_\pi(T)$ (5.1b) for BaySafeLTS **Type**-1 scheduling algorithm as a function of T. We plot the average of $R^c_\pi(T)$ and $U_\pi(T)$ over the 50 samples of $\theta_*, \mu_* \& C$ with the standard deviation around the mean for different β values. We observe that for **Type**-1 scheduling the cumulative regret $R^c_\pi(T)$ is proportional to the T and the slope for $\beta = 2$ is less compare to the slope for $\beta = 3$. And the averaged cumulative unsafe evaluation $U_\pi(T)$ is same for both β values.

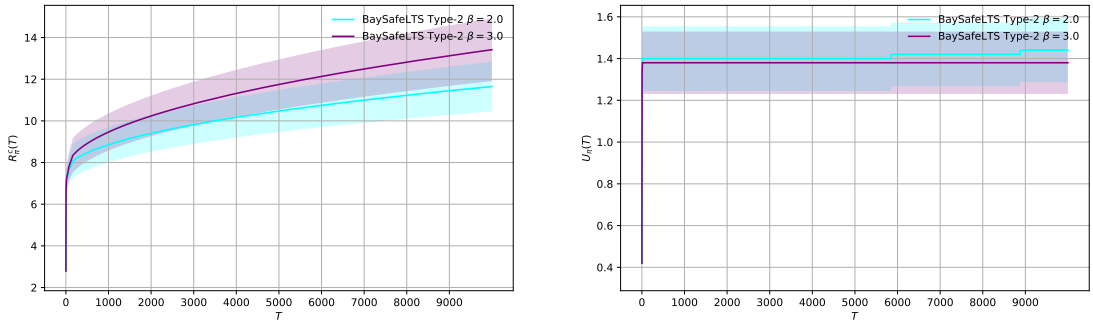


(a) **(b)**

Figure 5.1: Comparison of $R^c_\pi(T)$ (5.1a) and $U_\pi(T)$ (5.1b) as a function of T for max-variance scheduling with different β values. Observations are noisy with noise variance of 10^{-4}.

In Figure 5.2 we illustrate $R^c_\pi(T)$ (5.2a) and $U_\pi(T)$ (5.2b) for BaySafeLTS **Type**-2 scheduling algorithm as a function of T. We plot the average of $R^c_\pi(T)$ and $U_\pi(T)$ over the 50 samples of $\theta_*, \mu_* \& C$ with the standard deviation around the mean for different β values. We observe that for **Type**-2 scheduling the cumulative regret $R^c_\pi(T)$ is sub-linear and its value for proportional coefficient $\beta = 2$ is less compare to $\beta = 3$. And the averaged cumulative unsafe evaluation $U_\pi(T)$ is same for both β values.

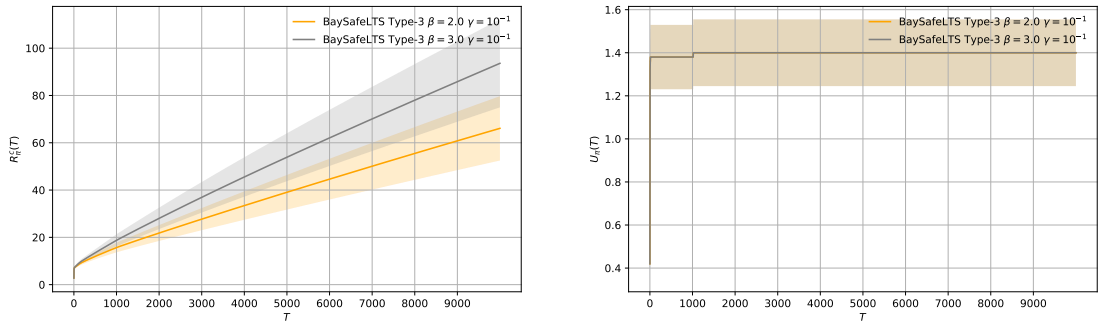
In Figure 5.3 we illustrate $R^c_\pi(T)$ (5.3a) and $U_\pi(T)$ (5.3b) for BaySafeLTS **Type**-3 scheduling algorithm as a function of T. We plot the average of $R^c_\pi(T)$ and $U_\pi(T)$ over the 50 samples of $\theta_*, \mu_* \& C$ with the standard deviation around the mean for different β values. We observe that for **Type**-3 scheduling the cumulative regret $R^c_\pi(T)$ is proportional to the T and the slope for $\beta = 2$ is less compare to the slope for $\beta = 3$. And the averaged



(a) **(b)**

Figure 5.2: Comparison of $R_\pi^c(T)$ (5.2a) and $U_\pi(T)$ (5.2b) as a function of T for max-variance square-root scheduling with different β values. Observations are noisy with noise variance of 10^{-4}.

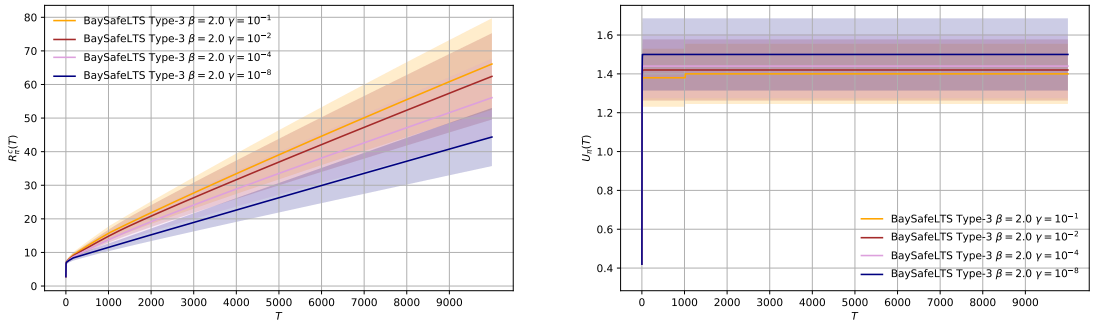
cumulative unsafe evaluation $U_\pi(T)$ is same for both β values.



(a) **(b)**

Figure 5.3: Comparison of $R_\pi^c(T)$ (5.3a) and $U_\pi(T)$ (5.3b) as a function of T for max-variance square-root log scheduling with different β values. Observations are noisy with noise variance of 10^{-4}.

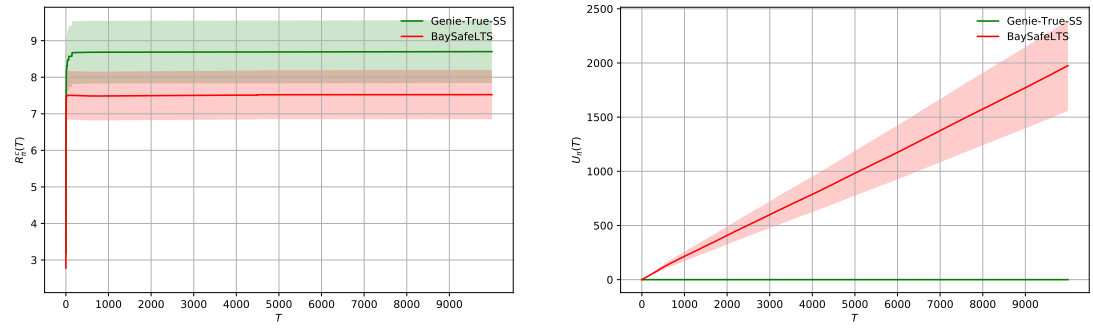
In Figure 5.4 we illustrate $R_\pi^c(T)$ (5.4a) and $U_\pi(T)$ (5.4b) for BaySafeLTS **Type**-3 scheduling algorithm as a function of T. We plot the average of $R_\pi^c(T)$ and $U_\pi(T)$ over the 50 samples of $\theta_*, \mu_* \& C$ with the standard deviation around the mean for different γ values. We observe that for **Type**-3 scheduling the cumulative regret $R_\pi^c(T)$ is proportional to the T and the slope is less for less value of γ. And the averaged cumulative unsafe evaluation $U_\pi(T)$ is increases with decrease in γ but the change is not significant.



(a) **(b)**

Figure 5.4: Comparison of $R_\pi^c(T)$ (5.4a) and $U_\pi(T)$ (5.4b) as a function of T for max-variance square-root log scheduling with different γ values. Observations are noisy with noise variance of 10^{-4}.

In Figure 5.5 we illustrate $R_\pi^c(T)$ (5.5a) and $U_\pi(T)$ (5.5b) for Genie-True-SS and BaySafeLTS with no scheduling algorithms as a function of T. We observe that for both algorithm the cumulative regret $R_\pi^c(T)$ coverage very quickly and $R_\pi^c(T)$ for BaySafeLTS with no scheduling is less compare to Genie-True-SS. And the cumulative unsafe evaluation $U_\pi(T)$ is proportional to T in case of BaySafeLTS with no scheduling and zero for Genie-True-SS.

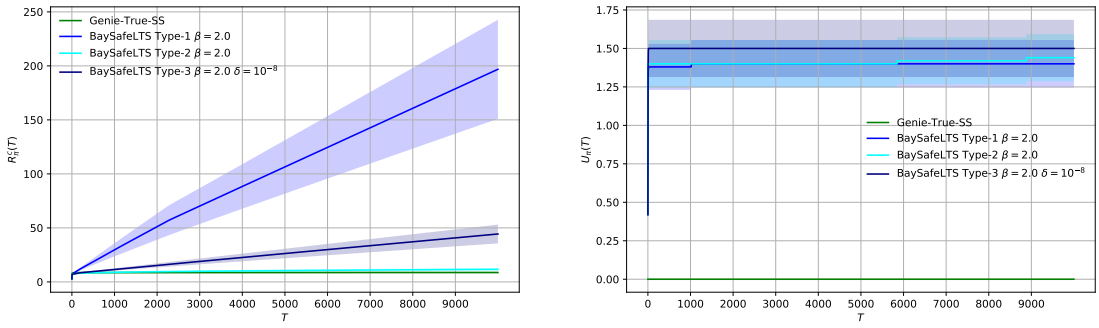


(a) **(b)**

Figure 5.5: Comparison of $R_\pi^c(T)$ (5.5a) and $U_\pi(T)$ (5.5b) as a function of T for all algorithm with no scheduling. Observations are noisy with noise variance of 10^{-4}.

In Figure 5.6 we illustrate (5.6a) and $U_\pi(T)$ (5.6b) for Genie-True-SS and BaySafeLTS with all type of scheduling as a function of T. We observe that for BaySafeLTS **Type**-1 and **Type**-3 scheduling the $R_\pi^c(T)$ is proportional to T but slope is less for **Type**-3

scheduling, for BaySafeLTS **Type**-2 scheduling the $R^c_\pi(T)$ is sub-linear and for Genie-True-SS $R(\pi)(T)$ converges very quickly. $U_\pi(T)$ is same for all the type of scheduling in BaySafeLTS and zero for Genie-True-SS.



(a) **(b)**

Figure 5.6: Comparison of $R^c_\pi(T)$ (5.6a) and $U_\pi(T)$ (5.6b) as a function of T for all algorithm with all type of scheduling. Observations are noisy with noise variance of 10^{-4}.

5.6 Conclusions

We formulated the problem of safe optimization in linear bandit setting. And we proposed a Thompson sampling based heuristic algorithm called BaySafeLTS and evaluated its performance via simulations. For the proposed algorithm there is only scheduling parameter α_t is to tune for the better performance expect the prior on unknown parameter θ_* and μ_*. We noted the for **Type**-2 scheduling the cumulative regret is sub-linear and for other type of scheduling it is linear and cumulative unsafe evaluation is very less and same for all type of scheduling.

Chapter 6
Conclusion

we formulated the problem of safe exploration and safe sequential optimization in switching environments. We identified a two way trade-off for exploration for safety and exploration for change points for safe exploration problem and for safe sequential optimization there is another trade-off for exploitation for the maximization. We proposed policies Adaptive Safe-Explore for safe exploration in switching environment and Adaptive-SafeOpt for safe sequential optimization in switching environment. We evaluated the performance of proposed algorithms via simulation and also compared their performance with baseline algorithms. We observed that a major challenge for extending safe exploration and optimization to switching environments is finding a safe point to continue exploration when there is a change in the unknown function. We find that on average, cumulative unsafe evaluation $U_\pi(T)$ increases for those policies for which the side information about the safe set is not available after the change-point. This is found to happen because the proposed algorithms get attracted to local maxima, which are unsafe. In case of safe optimization the variant Optimistic Adaptive-SafeOpt algorithm doesn't have this issue so it has less unsafe evaluation compare to variant Pessimistic Adaptive-SafeOpt. And these two perform same if we exclude the outlier cases. We note that the proposed Adaptive-SafeOpt converges but since the safe set that they explore is limited in size, the convergence is to a local maxima. We also note that Adaptive-SafeOpt with modified covariance and mean function perform better compare to Adaptive-SafeOpt without modified covariance and mean function.

We also formulated the problem of safe optimization in linear bandit setting. And we proposed a Thompson sampling based heuristic algorithm called BaySafeLTS and evaluated its performance via simulations. For the proposed algorithm there is only scheduling parameter α_t is to tune for the better performance expect the prior on unknown parameter θ_* and μ_*. We noted the for **Type**-2 scheduling the cumulative regret is sub-linear and for

other type of scheduling it is linear and cumulative unsafe evaluation is very less and same for all type of scheduling.

6.1 Future Work

Future work involves, to extend this to the MDP setting and also to obtain worst case as well as instance specific lower bounds to the performance of such safe optimization algorithms.

Bibliography

[1] R. S. Sutton and A. G. Barto, "Reinformcent learning: An introduction," 1998.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] A. Krause, M. Turchetta, and F. Berkenkamp, "Safe exploration in finite markov decision processes with gaussian processes," *Advances in Neural Information Processing Systems 29*, pp. 4312–4320, 2017.

[5] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with gaussian processes," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 491–496.

[6] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International Conference on Machine Learning*. PMLR, 2015, pp. 997–1005.

[7] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained mdps using gaussian processes." in *AAAI*, 2018, pp. 6548–6556.

[8] C. Paduraru, D. J. Mankowitz, G. Dulac-Arnold, J. Li, N. Levine, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning:definitions, benchmarks & analysis," *Machine Learning Journal*, 2021.

[9] A. Wachi and Y. Sui, "Safe reinforcement learning in constrained markov decision processes," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9797–9806.

[10] A. Wachi, H. Kajino, and A. Munawar, "Safe exploration in markov decision processes with time-variant safety using spatio-temporal gaussian process," *arXiv preprint arXiv:1809.04232*, 2018.

[11] J. Mellor and J. Shapiro, "Thompson sampling in switching environments with bayesian online change detection," in *Artificial Intelligence and Statistics*. PMLR, 2013, pp. 442–450.

[12] R. Alami, O. Maillard, and R. Féraud, "Memory bandits: a bayesian approach for the switching bandit problem," in *NIPS 2017-31st Conference on Neural Information Processing Systems*, 2017.

[13] G. Ghatak, "A change-detection based Thompson sampling framework for non-stationary bandits," *IEEE Transactions on Computers*, 2020.

[14] S. Padakandla, "A survey of reinforcement learning algorithms for dynamically varying environments," *arXiv preprint arXiv:2005.10619*, 2020.

[15] S. P. Coraluppi and S. I. Marcus, "Risk-sensitive and minimax control of discrete-time, finite-state markov decision processes," *Automatica*, vol. 35, no. 2, pp. 301–309, 1999.

[16] P. Geibel and F. Wysotzki, "Risk-sensitive reinforcement learning applied to control under constraints," *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.

[17] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[18] T. M. Moldovan and P. Abbeel, "Safe exploration in markov decision processes," *arXiv preprint arXiv:1205.4810*, 2012.

[19] E. Biyik, J. Margoliash, S. R. Alimo, and D. Sadigh, "Efficient and safe exploration in deterministic markov decision processes with unknown transition models," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1792–1799.

[20] M. Roderick, V. Nagarajan, and J. Z. Kolter, "Provably safe pac-mdp exploration using analogies," *arXiv preprint arXiv:2007.03574*, 2020.

[21] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[22] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*. KIT Scientific Publishing, 2010, vol. 9.

[23] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration in finite markov decision processes with gaussian processes," *arXiv preprint arXiv:1606.04753*, 2016.

[24] R. P. Adams and D. J. MacKay, "Bayesian online changepoint detection," *arXiv preprint arXiv:0710.3742*, 2007.

[25] A. Slivkins, "Book announcement: Introduction to multi-armed bandits," *SIGecom Exch.*, vol. 18, no. 1, p. 28–30, Dec. 2020. [Online]. Available: https://doi.org/10.1145/3440959.3440965

[26] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: no regret and experimental design," in *In International Conference on Machine Learning*, 2010.

[27] A. Moradipari, S. Amani, M. Alizadeh, and C. Thrampoulidis, "Safe linear thompson sampling with side information," *arXiv preprint arXiv:1911.02156*, 2019.