SAMPLE-EFFICIENT AUTOMATED MACHINE LEARNING WITH BAYESIAN OPTIMIZATION

DAI ZHONGXIANG

(B.Eng. (Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

SCHOOL OF COMPUTING

NATIONAL UNIVERSITY OF SINGAPORE

2021

Supervisors:

Associate Professor Kian Hsiang Low, Main Supervisor

Professor Patrick Jaillet, Co-Supervisor

Examiners:

Assistant Professor Jonathan Mark Scarlett

Assistant Professor Bryan Hooi Kuen-Yew

Assistant Professor Martinez-Cantin Ruben, University Of Zaragoza

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis. This thesis has also not been submitted for any degree in any university previously.

Dai Zhongxiang

July 8, 2021

Acknowledgements

I would like to express my sincere gratitude to my advisers, Prof Bryan Kian Hsiang Low and Prof Patrick Jaillet, for your constant support, patience and trust. Looking back, reaching out to Prof Bryan for an opportunity to work with both of you under the SMART program is the best decision I have made in my life so far. My experiences of working with you in the past few years have cemented my passion in machine learning and in research, and will undoubtedly help my career in the years to come.

I would also like to thank my parents, who may never read this since they don't speak English, for their unconditional love and support, for which I am forever grateful. I also appreciate the help and company from all the friends I have met along the way, who are too many to name. Special thanks go to those previously or currently in Prof Bryan's group, for making this the best research group imaginable.

Contents

Abstract

Automated hyperparameter optimization of machine learning (ML) models, referred to as AutoML, has been a challenging problem for practitioners, mainly due to the high computational cost of training modern ML models and the lack of gradient information with respect to the model hyperparameters. To this end, the black-box optimization method of *Bayesian optimization* (BO) has become a prominent method for optimizing the hyperparameters of ML models, which can be attributed to its impressive sample efficiency and theoretical convergence guarantee. Despite recent advances, there are still important scenarios where we can further *improve the sample efficiency of BO for AutoML* by exploiting naturally available auxiliary information, or *extend the applicability of BO to other ML tasks*. This thesis identifies five such important scenarios and, for each of them, proposes a novel BO algorithm that is both theoretically grounded and practically effective.

Firstly, many ML models require an iterative training process, which requires every hyperparameter evaluation during BO to run for a certain number of training epochs. As a result, the auxiliary observations from intermediate training epochs can be exploited to early-stop the evaluations of those unpromising hyperparameter configurations to save resource. We propose the *BO with Bayesian optimal stopping* (BO-BOS) algorithm, which incorporates BOS into BO in order to *improve the epoch efficiency of BO* using a principled optimal stopping mechanism. BO-BOS preserves the asymptotic no-regret property of

BO with our specified setting of BOS parameters which is amenable to an elegant interpretation in terms of the exploration-exploitation trade-off, and performs competitively in a number of AutoML experiments.

Secondly, the widely celebrated federated learning (FL) setting requires first-order optimization techniques, and is hence unable to handle zeroth-order optimization tasks such as hyperparameter optimization. We extend BO into the FL setting (FBO) and derive the *federated Thompson sampling* (FTS) algorithm, to *improve the efficiency of BO* in the FL setting by employing auxiliary information from other agents. FTS tackles a number of major challenges faced by FBO in a principled way: FTS uses random Fourier features approximation to derive the parameters to be communicated in order to avoid sharing the raw data, adopts the Thompson sampling algorithm which reduces the number of parameters to be exchanged, and is robust against heterogeneous agents due to a robust theoretical convergence guarantee.

Thirdly, the above-mentioned FTS algorithm, unfortunately, is not equipped with a rigorous privacy guarantee, which is an important consideration in FL. To this end, we integrate *differential privacy* (DP) into FTS through a general framework for adding DP to iterative algorithms. Moreover, we leverage the ability of this general DP framework to handle different parameter vectors, as well as the technique of local modeling for BO, to further improve the utility of our algorithm through distributed exploration (DE). The resulting DP-FTS-DE algorithm is able to *improve an agent's sample efficiency* by exploiting auxiliary information from other agents, while *rigorously hiding its participation* in the algorithm. DP-FTS-DE is amenable to a number of interesting theoretical insights regarding the privacy-utility trade-off, and achieves competitive utilities with strong privacy guarantees in real-world experiments.

Fourthly, when BO is used for hyperparameter optimization using a dataset, we often have access to previous completed hyperparameter optimization tasks using

other potentially related datasets. This prompts the question as to whether we can leverage these previous completed tasks to *improve the efficiency of the current BO task* through *meta-learning*, while ensuring its robustness against dissimilar tasks. We introduce a scalable, principled and robust meta-BO algorithm called *robust meta-Gaussian process-upper confidence bound* (RM-GP-UCB). We show that RM-GP-UCB is asymptotically no-regret even when all previous tasks are dissimilar to the current task, and is amenable to a principled method to learn the weights assigned to the individual previous tasks through regret minimization via online learning. RM-GP-UCB achieves effective performances in a wide range of real-world experiments.

Lastly, many ML tasks such as adversarial ML can be modeled as repeated games between boundedly rational, self-interested agents with unknown, complex, and costly-to-evaluate payoff functions. We introduce a recursive reasoning formalism of BO, called *Recursive Reasoning-Based BO* (R2-B2), which *extends the applicability of BO* to provide efficient strategies for players in this type of game. Under certain conditions, using R2-B2 to reason at one level higher than the other agents achieves faster asymptotic convergence to no regret than without using recursive reasoning. R2-B2 performs effectively in practice in adversarial ML and multi-agent reinforcement learning experiments.

List of Publications

Below is a list of publications whose contributions are included in this thesis:

- **Dai, Z.**, Yu, H., Low, B. K. H., & Jaillet, P. (2019). Bayesian optimization meets Bayesian optimal stopping. In *ICML-19*.
- **Dai, Z.**, Low, B. K. H., & Jaillet, P. (2020). Federated Bayesian optimization via Thompson sampling. In *NeurIPS-20*.
- **Dai, Z.**, Chen, Y., Low, B. K. H., Jaillet, P., & Ho, T. H. (2020). R2-B2: Recursive reasoning-based Bayesian optimization for no-regret learning in games. In *ICML-20*.

Listed below are some other publications:

- Kharkovskii, D., **Dai, Z.**, & Low, B. K. H. (2020). Private outsourced Bayesian optimization. *ICML-20*.
- Zhang, Y., **Dai, Z.**, & Low, B. K. H. (2020). Bayesian optimization with binary auxiliary information. *UAI-19*.
- Yu, H., Chen, Y., **Dai, Z.**, Low, B. K. H., & Jaillet, P. (2019). Implicit posterior variational inference for deep Gaussian processes. In *NeurIPS-19*.

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

In recent years, *machine learning* (ML), especially *deep learning* (DL), has demonstrated unprecedented levels of performances in various applications such as image recognition, natural language processing, complex board games, among other (LeCun et al., 2015; Silver et al., 2016). However, a considerable challenge repeatedly faced by ML practitioners is the choice of model hyperparamters, which have been found to have significant impacts on the performances of ML models (Shahriari et al., 2016). A major difficulty in optimizing the hyperparameters of ML models results from the massive computational cost of training modern ML models such as deep neural networks (DNN), which makes manual hyperparameter tuning infeasible. This therefore calls for automated hyperparameter optimization and has given rise to the widely celebrated field of *automated machine learning* (AutoML) (Hutter et al., 2019). Moreover, the lack of gradient information with respect to the model hyperparameters also exacerbates the difficulty of automated hyperparameter optimization since it precludes the use of gradient-based optimization methods.

To this end, *Bayesian optimization* (BO) has recently become a prominent

method for automated hyperparameter optimization mainly because (a) it has repeatedly demonstrated impressive sample efficiency, and (b) it is a zeroth-order black-box optimization method which does not require access to gradient information (Shahriari et al., 2016; Snoek et al., 2012). BO uses a *Gaussian process* (GP) as a surrogate to model the objective function, and sequentially selects input locations to query by trading-off *exploration* of the input domain and *exploitation* of the information collected so far. From the theoretical perspective, some classes of BO algorithms, such as *GP-upper confidence bound* (GP-UCB) (Srinivas et al., 2010) and *Thompson sampling* (TS) (Chowdhury and Gopalan, 2017), have been shown to enjoy strong theoretical performance guarantees, which facilitates the development of AutoML methods that are both theoretically principled and practically effective. Of note, the generality of BO as a sample-efficient black-box optimization method has allowed its application to extend beyond AutoML and into many other interesting areas. For example, BO has also been successfully applied in the automated design of black-box adversarial attacks for *adversarial ML* (Ru et al., 2020), and in many real-world experimental design problems such as material design (Frazier and Wang, 2016), molecule design (Korovina et al., 2020), design of chemical experiments (Burger et al., 2020), etc.

Despite significant progresses, there are still important scenarios where we can further *improve the sample efficiency of BO for AutoML* by exploiting naturally available auxiliary information, or *extend the applicability of BO to other ML tasks*. We identify here five such important scenarios:

(a) The evaluation of every hyperparameter configuration usually requires an iterative training process (e.g., stochastic gradient descent); therefore, when BO is used for hyperparameter optimization, we may exploit the auxiliary information during training to early-stop some unpromising hyperparameter evaluations to improve the epoch efficiency of BO.

(b) Current federated learning (FL) algorithms rely on first-order optimization techniques and are hence unable to handle zeroth-order optimization tasks such as hyperparameter tuning; therefore, BO might be extended to the FL setting (i.e., the FBO setting) to achieve faster hyperparameter optimization by collaborating with other agents, i.e., by exploiting auxiliary observations from other agents.

(c) In some applications of FBO, the auxiliary observations from other agents contain highly sensitive information; as a result, when exploiting these auxiliary observations to achieve a higher sample efficiency for BO, a rigorous guarantee on the privacy of the individual agents is required.

(d) When BO is used for hyperparameter optimization using a dataset, we often have access to previous completed optimization tasks using potentially related datasets; this begs the question as to whether we can leverage these previous optimization tasks to improve the efficiency of the current BO task through *meta-learning*.

(e) Some ML tasks can be modeled as repeated games with unknown, complex and costly-to-evaluate payoff functions, for example, adversarial ML can be modeled as a repeated game between an attacker and a defender; in these games, it is unclear whether BO, as a sample-efficient black-box optimization method, can be applied to derive efficient strategies for the players.

Scenarios (a-d) represent opportunities to further *improve the sample efficiency of BO for AutoML* by making use of naturally available side information (auxiliary observations obtainable at reasonable costs from (a) intermediate training epochs, (b-c) other agents and (d) previous tasks, respectively), whereas scenario (e) offers the potential for *new applications of BO* in other important ML tasks. This brings up a natural and important research question:

Can we design practical BO algorithms with theoretical performance guarantee for these important scenarios, to further improve the sample efficiency of BO for AutoML and extend the applicability of BO to other ML tasks?

We answer this question affirmatively in this thesis by introducing novel BO algorithms that are both theoretically principled and practically effective for each of these five scenarios.

Interestingly, our novel BO algorithms for these five scenarios span both the single-agent and multi-agent settings (Fig. 1.1). Specifically, as illustrated in Fig. 1.1, scenarios (a) and (d) correspond to the single-agent setting where an agent attempts to use its own auxiliary information (from intermediate training epochs and its previous tasks, respectively) to accelerate BO. The multi-agent setting encompasses scenarios (b), (c) and (e): In scenarios (b) and (c), multiple agents *collaborate* to improve their sample efficiency while considering important issues such as privacy and communication; in scenario (e), every agent is *self-interested* and attempts to achieve a high payoff in a game. Of note, beyond AutoML and ML tasks, our proposed algorithms for both single-agent and multi-agent settings can also find other interesting applications. For example, in the single-agent setting, our algorithms for scenarios (a) and (d) can also be applied to *precision agriculture* where BO can be used to optimize the growing conditions for crops, to early-stop those evaluated conditions that are unlikely to yield satisfactory outcomes (scenario (a)), or to accelerate the current BO task by leveraging previous tasks using other types of crops (scenario (d)). In the multi-agent setting with collaborative agents, when multiple hospitals use BO to select the patients to perform a medical test, our algorithms for scenarios (b) and (c) can be adopted to improve the efficiency of patient selection by allowing every hospital to exploit the information from the other hospitals without compromising the privacy of patients. On the other hand, self-interested agents in multi-agent systems (e.g., autonomous vehicles or robots) can employ our algorithm for scenario (e) to

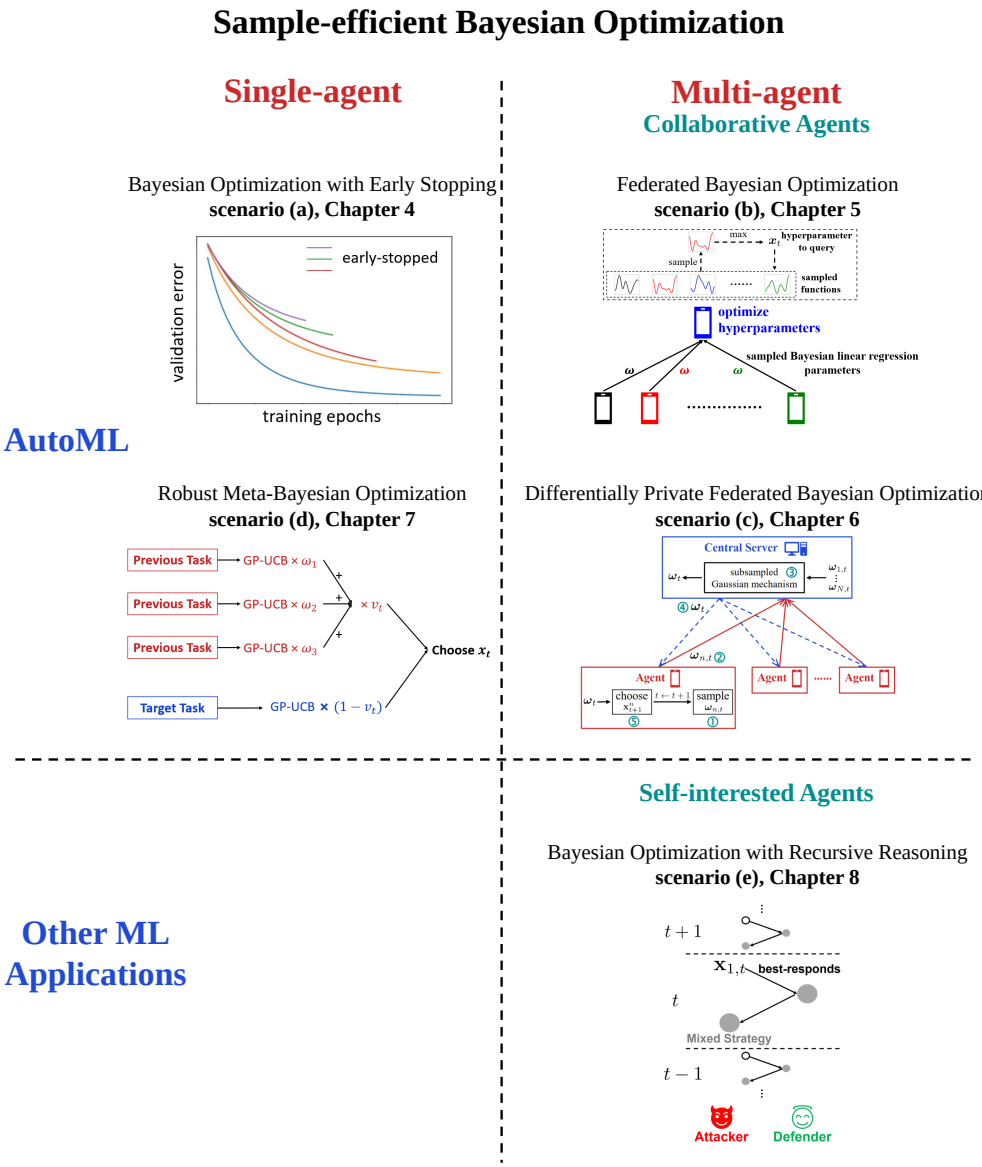


Figure 1.1: Overview of the five works in this thesis.

derive efficient strategies in the system.

Fig. 1.1 gives an overview of the five works in this thesis, categorized by whether they focus on AutoML or other ML applications, and by whether they are developed for the single-agent or multi-agent setting. In the next section, we outline our contributions in this thesis by providing a brief summary of each of these five works.

1.2 Contributions

Bayesian Optimization with Early Stopping (Dai et al., 2019). Many ML models require running an iterative training procedure for a certain number of epochs (e.g., stochastic gradient descent for DNN), which makes certain information during the training process available (e.g., the validation accuracy after each epoch). This motivates the question as to whether we can exploit these information to improve the epoch efficiency of BO algorithms by early-stopping model training under those hyperparameter settings that will end up underperforming. This work (Chapter 4) proposes to unify *BO* (specifically, GP-UCB) *with Bayesian optimal stopping* (BO-BOS) (Powell and Ryzhov, 2012) to boost the epoch efficiency of BO. To achieve this, while GP-UCB is sample-efficient in the number of function evaluations, BOS complements it with epoch efficiency for each function evaluation by providing a principled optimal stopping mechanism for early stopping. BO-BOS preserves the asymptotic no-regret performance of GP-UCB using our specified choice of BOS parameters that is amenable to an elegant interpretation in terms of the exploration-exploitation trade-off. We demonstrate the competitive performance of BO-BOS in hyperparameter tuning of ML models, and showcase its generality by applying it to two interesting applications of policy search for reinforcement learning (RL) and feature selection.

Federated Bayesian Optimization (Dai et al., 2020b). The massive

computational capability of edge devices such as mobile phones, coupled with privacy concerns, has led to immense recent interest in *federated learning* (FL) (McMahan et al., 2017), which focuses on collaborative training of DNN via *first-order optimization* techniques. However, some common ML tasks such as hyperparameter tuning of DNN lack access to gradients and thus require *zeroth-order optimization* (black-box optimization). This hints at the considerable potential of extending BO to the FL setting (which we refer to as the FBO setting), to allow agents to collaborate in these black-box optimization tasks. In this work (Chapter 5), we introduce the *federated Thompson sampling* (FTS) algorithm, which overcomes a number of key challenges of FBO and FL in a principled way: We (a) use *random Fourier features* (Rahimi and Recht, 2007) to approximate the Gaussian process surrogate model used in BO which naturally produces the parameters to be exchanged between agents and hence avoids the sharing of raw data, (b) design FTS based on *Thompson sampling* (Thompson, 1933) which significantly reduces the number of parameters to be exchanged, and (c) provide a theoretical convergence guarantee that is robust against heterogeneous agents which is a major challenge in FL and FBO. We empirically demonstrate the effectiveness of FTS in terms of communication efficiency, computational efficiency and practical performance.

Differentially Private Federated Bayesian Optimization. Despite being able to tackle a number of major challenges in the FBO setting, the above-mentioned FTS algorithm (Dai et al., 2020b) is not equipped with a rigorous privacy guarantee which is an important consideration in FL. Recent works (Abadi et al., 2016; McMahan et al., 2018b) have incorporated *differential privacy* (DP) into the training of DNN through a general framework for adding DP to iterative algorithms (McMahan et al., 2018a). In this work (Chapter 6), following this general DP framework, we integrate DP into FTS to preserve the *user-level privacy*, i.e., to rigorously hide the participation of any individual user. Moreover,

we leverage the ability of this general DP framework to handle different parameter vectors, as well as the technique of local modeling for BO, to further improve the utility of our algorithm through *distributed exploration* (DE). The resulting *differentially private FTS with DE* (DP-FTS-DE) algorithm is endowed with theoretical guarantees for both the privacy and utility and is amenable to interesting theoretical insights about the *privacy-utility trade-off*. We also use real-world experiments to show that DP-FTS-DE achieves a high utility (i.e., competitive performance) with a strong privacy guarantee (i.e., small privacy loss) and induces a practical trade-off between privacy and utility.

Robust Meta-Bayesian Optimization. When BO is used to optimize a target function, we often have access to some previous evaluations of potentially related functions. This begs the question as to whether we can leverage these previous experiences to accelerate the current BO task through *meta-learning* (meta-BO), while ensuring *robustness* against potentially harmful dissimilar tasks that could sabotage the convergence of BO. In this work (Chapter 7), we introduce a scalable, principled and robust meta-BO algorithm called *robust meta-Gaussian process-upper confidence bound* (RM-GP-UCB). RM-GP-UCB utilizes a weighted combination of separate acquisition functions from individual tasks for query selection, hence achieving scalability in the number of previous tasks and observations for each previous task. We derive a robust theoretical convergence guarantee for RM-GP-UCB and show that it is asymptotically no-regret even when some or all previous tasks are dissimilar to the current task. Moreover, the theoretical guarantee allows RM-GP-UCB to optimize the weights assigned to the individual previous tasks, hence diminishing the impact of dissimilar tasks, in a principled way through regret minimization via online learning. Empirical evaluation shows that RM-GP-UCB performs effectively and consistently across various applications.

Bayesian Optimization with Recursive Reasoning for Games (Dai et al.,

2020a). Some ML tasks can be modeled as repeated games between boundedly rational, self-interested agents with unknown, complex, and costly-to-evaluate payoff functions. For example, adversarial ML can be modeled as a repeated game between a defender and an attacker, whose payoff functions are the performance of the target ML model and its negation respectively. This work (Chapter 8) presents a recursive reasoning formalism of BO to model the reasoning process in the interactions among the players in this type of game, which we call *Recursive Reasoning-Based BO* (R2-B2). Our R2-B2 algorithm is general in that it does not constrain the relationship among the payoff functions of different agents and can thus be applied to various types of games such as constant-sum, general-sum, and common-payoff games. We prove that by reasoning at level 2 or more and at one level higher than the other agents, our R2-B2 agent can achieve faster asymptotic convergence to no regret than that without utilizing recursive reasoning. We also propose a computationally cheaper variant of R2-B2 called R2-B2-Lite at the expense of a weaker convergence guarantee. The performance and generality of our R2-B2 algorithm are empirically demonstrated using synthetic games, adversarial machine learning, and multi-agent reinforcement learning.

1.3 Organization

In the remainder of this thesis, we firstly introduce the necessary background and notations in Chapter 2, followed by the related works in Chapter 3. Next, the following five chapters present each of the five works in detail: BO with early stopping (Chapter 4), federated BO (Chapter 5), differentiallly private federated BO (Chapter 6), robust meta-BO (Chapter 7), and BO with recursive reasoning for games (Chapter 8). Lastly, we summary the thesis and provide a future outlook in Chapter 9.

Chapter 2

Background and Notations

In this chapter, we introduce the necessary background and notations which will be useful throughout this thesis, focusing on *Gaussian processes* (GP) and *Bayesian optimization* (BO). We use lower-case bold-faced symbols to denote (column) vectors (e.g., $\mathbf{x}$), and upper-case bold-faced symbols to represent matrices (e.g., $\mathbf{X}$). Scalars are not highlighted in bold (e.g., x).

2.1 Gaussian Processes

A GP is a stochastic process in which any finite subset of random variables follows a multivariate Gaussian distribution (Rasmussen and Williams, 2006). A GP defines a distribution over functions $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^d$ is the domain. We focus on the noisy setting in which every observation of the function f at an input $\mathbf{x}$ is corrupted by a zero-mean additive Gaussian noise: $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and σ^2 is the noise variance. A GP, represented as $\mathcal{GP}(\mu(\cdot), k(\cdot, \cdot))$, is fully characterized by its mean function $\mu(\mathbf{x})$ and covariance (kernel) function $k(\mathbf{x}, \mathbf{x}'), \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$. The kernel function k represents a similarity measure of a pair of inputs, and encodes our assumption on the smoothness of the function f. We focus on some commonly used stationary

positive semi-definite kernels, such as the *squared exponential* (SE) kernel and Matérn kernel. The SE kernel can be expressed as $k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp(-\frac{\left\|\mathbf{x}-\mathbf{x}'\right\|_2^2}{2l^2})$ where l and σ_0^2 are the kernel hyperparameters (referred to as the *length scale* and signal variance, respectively). The Matérn kernel can be written as: $k(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\left\|\mathbf{x}-\mathbf{x}'\right\|_2}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\left\|\mathbf{x}-\mathbf{x}'\right\|_2}{l} \right)$, where Γ is the gamma function, K_ν is a modified Bessel function, and ν and l are kernel hyperparameters. Throughout this thesis, we assume w.l.o.g. that $\mu(\mathbf{x}) = 0$ and $k(\mathbf{x}, \mathbf{x}') \leq 1, \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$. As a result of the definition of GP, conditioned on a set of T observed input-output pairs $D_T = \{(\mathbf{x}_1, y(\mathbf{x}_1)), \ldots, (\mathbf{x}_T, y(\mathbf{x}_T))\}$, the posterior belief at any input $\mathbf{x}$ is Gaussian-distributed, whose posterior mean and covariance can be expressed as:

$$\begin{aligned} \mu_T(\mathbf{x}) &= \mathbf{k}_T(\mathbf{x})^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_T, \\ \sigma_T^2(\mathbf{x}, \mathbf{x}') &= k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_T(\mathbf{x})^\top (\mathbf{K}_T + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_T(\mathbf{x}'), \end{aligned} \tag{2.1}$$

in which $\mathbf{K}_T = [k(\mathbf{x}_t, \mathbf{x}_{t'})]_{t,t'=1,\ldots,T}$ is the $T \times T$ gram matrix, $\mathbf{k}_T(\mathbf{x}) = [k(\mathbf{x}, \mathbf{x}_t)]_{t=1,\ldots,T}^\top$ and $\mathbf{y}_T = [y(\mathbf{x}_1), \ldots, y(\mathbf{x}_T)]^\top$ are both $T \times 1$ column vectors. We denote the posterior variance at $\mathbf{x}$ as $\sigma_T^2(\mathbf{x}) = \sigma_T^2(\mathbf{x}, \mathbf{x})$.

2.2 Bayesian Optimization

In a BO problem, we attempt to find a global maximum of an objective function f within a domain $\mathcal{X} \subset \mathbb{R}^d$, i.e., find $\mathbf{x}^* \in \arg\max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$, through sequentially querying the function f. Throughout this thesis, we assume the domain $\mathcal{X}$ to be discrete for simplicity, however, all our theoretical analysis can be straightforwardly extended to compact domain through a suitable discretization by following similar steps of analysis to the work of (Srinivas et al., 2010). In iteration t, BO queries an input $\mathbf{x}_t$ to observe a noisy output $y_t = y(\mathbf{x}_t) = f(\mathbf{x}_t) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ as defined in Section 2.1. The performance of BO is usually measured in terms of *regret*, which represents how much BO suffers from not

knowing the location of a global maximum in advance. The *instantaneous regret* in iteration t is defined as $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$, which gives rise to the definitions of the *cumulative regret*: $R_T = \sum_{t=1}^{T} r_t$ and *simple regret*: $S_T = \min_{t=1,\dots,T} r_t$. It is particularly desirable for a BO algorithm to be asymptotically *no-regret*, i.e., for R_T to grow sub-linearly. This implies that $S_T \leq \frac{R_T}{T}$ goes to 0 asymptotically, which guarantees that we are able to find a global maximum asymptotically.

In order to choose $\mathbf{x}_t$ intelligently to minimize regret, we use a GP as a surrogate to model the objective function f and hence choose $\mathbf{x}_t$ by maximizing an *acquisition function* α_t. That is, in iteration t of BO, we firstly update the GP posterior belief (2.1), and then use the updated posterior to calculate the acquisition function, whose maximizer is selected as $\mathbf{x}_t$ to query: $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha_t(\mathbf{x})$. The acquisition function needs to be able to balance *exploration* and *exploitation*, that is, balance (a) exploring the places in the input domain with large uncertainty in order to improve the GP posterior belief, and (b) exploiting the current GP posterior belief to prefer the input locations with large posterior mean. A number of acquisition functions have been proposed over the years, such as *Gaussian process-upper confidence bound* (GP-UCB) (Srinivas et al., 2010), *Thompson sampling* (TS) (Chowdhury and Gopalan, 2017), expected improvement (EI) (Jones et al., 1998), entropy search (Hennig and Schuler, 2012), predictive entropy search (Hernández-Lobato et al., 2014), knowledge gradient (Frazier, 2018), max-value entropy search (Wang and Jegelka, 2017), etc. In this thesis, we particularly focus on GP-UCB and TS, mainly due to their strong theoretical properties, which allow us to derive BO algorithms that are both theoretically grounded and practically competitive.

The GP-UCB acquisition function takes the simple form of a weighted combination of the posterior mean and standard deviation: $\alpha_t(\mathbf{x}) = \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x})$, which encourages exploitation and exploration respectively. TS requires sampling a function f_t from the GP posterior in iteration t, and then

choosing the maximum of the sampled function to query: $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$. Both GP-UCB and TS have been shown to be asymptotically no-regret (Srinivas et al., 2010; Chowdhury and Gopalan, 2017). Of note, for both GP-UCB and TS, the upper bound on the cumulative regret depends on the maximum *information gain* γ_T, which represents the maximum amount of information about the function f that can be obtained from querying any set of T input locations. The term γ_T is kernel-dependent, and its asymptotic growth has been characterized for some commonly used kernels (Srinivas et al., 2010). For example, for the SE kernel, $\gamma_T = O\left((\log T)^{d+1}\right)$; for the Matérn kernel with $\nu > 1$, $\gamma_T = O\left(T^{d(d+1)/(2\nu+d(d+1))}\log T\right)$.

Note that in future chapters, some notations may be adjusted to be more suitable for the particular setting under consideration. For example, in Chapter 4 where we give special treatment to the number of training epochs, instead of $\mathbf{x}$, we denote the input to the objective function by $\mathbf{z} = [\mathbf{x}, n]$ where $\mathbf{x}$ represents a hyperparameter configuration and n denotes the number of training epochs; in Chapter 8 where we model games involving two (or more) players, the input is denoted as $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$, where $\mathbf{x}_1$ and $\mathbf{x}_2$ represent the inputs from the two players.

2.3 Federated Learning

An important topic related to the works in this thesis (Chapters 5 and 6) is *federated learning* (FL) (McMahan et al., 2017). FL has been recently introduced to tackle some major challenges in collaborative training of modern deep neural networks. In every round of the *federated averaging* algorithm (i.e., one of the most representative FL algorithms), every user trains its local neural network via first-order optimization (e.g., stochastic gradient descent) and passes its parameters (or gradients) to a central server; the central server

then aggregates the received parameters/gradients from all users, and broadcasts the aggregated parameters/gradients back to all users for updating their local neural networks (McMahan et al., 2017). Some important challenges in FL include avoiding the sharing of the raw data of users, the requirement to satisfy the communication bandwidth, the potential high degree of heterogeity among different users, among others (Kairouz et al., 2019). An important consideration when applying FL to privacy-sensitive use cases is a rigorous privacy guarantee (Kairouz et al., 2019). Therefore, the privacy-preserving mechanism of *differential privacy* has been incorporated into FL (McMahan et al., 2018b), which modifies the aggregation process by the central server to incorporate privacy-preserving transformations.

Chapter 3

Related Works

In this chapter, we give a review of related works for each of the five works included in this thesis, to elucidate the position of our contributions within the literature.

3.1 Bayesian Optimization with Early Stopping

Some recent works have been proposed to incorporate early stopping into BO, in order to make BO more epoch-efficient. Freeze-thaw BO (Swersky et al., 2014) explores a diverse collection of hyperparameter settings in the initial stage by training their ML models with a small number of epochs, and then gradually focuses on (exploits) a small number of promising settings. Despite its promising epoch efficiency, its performance is not theoretically guaranteed and its computational cost can be excessive. Multi-fidelity BO (Kandasamy et al., 2016, 2017; Wu and Frazier, 2018; Zhang et al., 2017) reduces the resource consumption of BO by utilizing low-fidelity functions which can be obtained by training the ML model for a small number of epochs. However, in each BO iteration, since the fidelity (e.g., number of epochs) is determined before function evaluation, it is not influenced by information that is typically available during

the training process (e.g., validation accuracy after each epoch). In addition to BO, attempts have also been made to improve the epoch efficiency of other hyperparameter optimization algorithms. Some heuristic methods (Baker et al., 2017; Domhan et al., 2015; Klein et al., 2017) predict the final training outcome based on partially trained learning curves in order to identify hyperparameter settings that are predicted to under-perform and early-stop their model training. Hyperband (Li et al., 2017), which dynamically allocates the computational resource (e.g., training epochs) through random sampling and eliminates under-performing hyperparameter settings by successive halving, has been proposed and shown to perform well in practice. Both the learning curve prediction methods and Hyperband can be combined with BO to further improve their epoch efficiency (Domhan et al., 2015; Falkner et al., 2018; Klein et al., 2017), but their resulting performances are not theoretically guaranteed. Despite these recent advances, an epoch-efficient BO algorithm with theoretically guaranteed performance is still lacking, which is what we present in Chapter 4.

3.2 Federated Bayesian Optimization

Since its recent introduction in McMahan et al. (2017), *federated learning* (FL) has gained tremendous attention mainly due to its prominent practical relevance in the collaborative training of ML models such as DNN (McMahan et al., 2017) or decision tree-based models (Li et al., 2020a,b). Meanwhile, efforts have also been made to derive theoretical convergence guarantees for FL algorithms (Li et al., 2018, 2020d). Refer to recent surveys (Kairouz et al., 2019; Li et al., 2019a,b) for more comprehensive reviews of FL. Although our algorithm (Chapter 5) is the first BO algorithm in the FL setting, it is also related to some previous works on BO. Thompson sampling (TS) (Thompson, 1933) has been known as a highly effective practical technique for multi-armed bandit problems (Chapelle and Li,

2011; Russo et al., 2017). The Bayesian regret (Russo and Van Roy, 2014) and frequentist regret (Chowdhury and Gopalan, 2017) of TS in BO have both been analyzed, and TS has been shown to perform effectively in BO problems such as high-dimensional BO (Mutny and Krause, 2018). Our federated BO algorithm (Chapter 5) is also related to multi-fidelity BO (Kandasamy et al., 2016; Poloczek et al., 2017; Wu et al., 2020; Zhang et al., 2020, 2017) which has the option to query low-fidelity functions. This is analogous to our federated BO algorithm (Chapter 5) allowing the target agent to use the information from the other agents for query selection, and the similarity between an agent and the target agent can be interpreted as a measure of fidelity. Moreover, our algorithm also bears similarity to parallel/distributed BO algorithms (Contal et al., 2013; Daxberger and Low, 2017; Desautels et al., 2014), especially those based on TS (Hernández-Lobato et al., 2017; Kandasamy et al., 2018). However, there are fundamental differences: For example, they usually optimize a single objective function whereas we need to consider possibly heterogeneous objective functions from different agents. The work of Garcia-Barcos and Martinez-Cantin (2019) has proposed a fully distributed BO algorithm which has also adopted a sampling-based strategy for query selection. However, similar to other distributed BO methods, Garcia-Barcos and Martinez-Cantin (2019) have also focused on optimizing a single objective function, and have not considered the important issue of avoiding transmitting the raw data in the federated setting. Furthermore, some previous works on meta-learning for BO (Feurer et al., 2018; Wistuba et al., 2018), which study how to use the information from other related BO tasks to accelerate the current BO task, can be adapted for the federated BO setting. However, as we will discuss in Chapter 5, these works are not equipped with theoretical convergence guarantee and do not tackle some important issues in FL and FBO such as avoiding the transmission of raw data and achieving efficient communication.

3.3 Differentially Private Federated Bayesian Optimization

Recent works have incorporated privacy preservation into BO by applying DP to the output of BO (Kusner et al., 2015), using a different notion of privacy (other than DP) (Nguyen et al., 2018), adding DP to BO in the outsourced setting (Kharkovskii et al., 2020), or adding local DP into BO (Zhou and Tan, 2020). However, none of these works can tackle the FBO setting considered in this work. Similar to FTS (Section 3.2), our DP-FTS-DE algorithm in this chapter (Chapter 6) also shares similarity with previous works on parallel BO algorithms (Contal et al., 2013; Desautels et al., 2014; Hernández-Lobato et al., 2017; Kandasamy et al., 2018). However, parallel BO optimizes a single objective function while we allow agents to have different objective functions. Our DE technique bears similarity to that of Eriksson et al. (2019) which has also used separate GP surrogates to model different local sub-regions (hyper-rectangles) and shown that this significantly improves the performance. Privacy preservation using DP has been an important topic for FL, including both central DP (i.e., with a trusted central server) (McMahan et al., 2018b) and local (Kasiviswanathan et al., 2011; Warner, 1965) or distributed DP (Bittau et al., 2017; Cheu et al., 2019; Dwork et al., 2006a; Shi et al., 2011) (i.e., without a trusted central server).

3.4 Robust Meta-Bayesian Optimization

Some previous works on meta-BO have built a joint GP surrogate using all previous and current observations, and represented task similarity through meta-features (Bardenet et al., 2013; Schilling et al., 2016; Yogatama and Mann, 2014). However, these algorithms suffer from the requirement of handcrafted meta-features, which is avoided in other works that learn task similarity from

the observations (Swersky et al., 2013; Shilton et al., 2017). For example, multitask BO (Swersky et al., 2013) uses a multitask GP as a surrogate and models each task as an output of the GP. These works have included all previous and current observations in a single GP surrogate and are thus limited by GP's scalability. There have also been other empirical works which replace GP with Bayesian linear regression for scalability (Perrone et al., 2018), tackle sequentially arriving tasks (Golovin et al., 2017; Poloczek et al., 2016), learn a set of good initializations (Feurer et al., 2015; Wistuba et al., 2015b), learn a reduced search space for BO from previous tasks (Perrone et al., 2019), handle the issue of different function scales using Gaussian Copulas (Salinas et al., 2020), or use the meta-observations to learn the entire acquisition function through RL (Volpp et al., 2020). The work of Wang et al. (2018) has learned the GP prior from previous tasks and also given theoretical guarantee. However, they have shown in both theory and practice that a large training set of meta-observations (≥ 5000) is required for their method to work well, while we focus on the more practical setting of meta-BO where the number of available meta-observations may be small. We have also verified that our algorithm outperforms the method from Wang et al. (2018) in the experiment that is most favorable for their method among all our experiments (Chapter 7). Multi-fidelity BO methods (Kandasamy et al., 2016, 2017; Wu and Frazier, 2018; Zhang et al., 2017) can also be applied to solve meta-BO problems by treating the previous tasks as observations from lower-fidelity functions and only allowing queries of the target function (not lower-fidelity functions).

Some recent works have aimed to improve the scalability of GP-based meta-BO algorithms by building a separate GP surrogate for each task (Feurer et al., 2018; Wistuba et al., 2016, 2018). The work of Wistuba et al. (2016) has used a weighted combination of the posterior mean of each individual GP surrogate as the joint posterior mean while the posterior variance is derived using only

the target observations. The *Ranking-weighted Gaussian Process Ensemble* (RGPE) algorithm (Feurer et al., 2018) has extended the work of Wistuba et al. (2016) by estimating the joint objective function as a weighted combination of individual objective functions, such that the resulting joint surrogate remains a GP (unlike Wistuba et al. (2016)) and can thus be plugged into standard BO algorithms. The work of Wistuba et al. (2018) has proposed the *transfer acquisition function* (TAF) algorithm, which has also used a weighted combination of the acquisition functions (i.e., expected improvement) from the individual tasks for query selection. In these works, the weight of a previous task is heuristically chosen to be proportional to the accuracy of the *pairwise ranking of the target observations* produced by either (a) the posterior mean of the GP surrogate of the previous task (TAF) (Wistuba et al., 2018) or (b) functions sampled from the posterior GP surrogate (RGPE) (Feurer et al., 2018). As we will show in our experiments (Chapter 7), our proposed RM-GP-UCB algorithm outperforms both RGPE and TAF in a number of real-world experiments.

3.5 Bayesian Optimization with Recursive Reasoning for Games

The connection between BO and game theory has recently begun to be explored. The recent work of (Sessa et al., 2019) has combined online learning and GP-UCB to derive a no-regret learning algorithm called *GP-multiplicative weight* (GP-MW) for repeated games. As we will explain in Chapter 8, GP-MW can be encompassed by our R2-B2 algorithm as a special case in which no recursive reasoning is performed. Moreover, BO has also been recently applied in game theory to find the Nash equilibria (Picheny et al., 2019).

Humans possess the ability to reason about the mental states of others (Goldman, 2012). In particular, a person tends to reason recursively by analyzing

the others' thinking about himself, which gives rise to recursive reasoning (Pynadath and Marsella, 2005). The recursive reasoning model of humans has inspired the development of the cognitive hierarchy model in behavioral game theory, which uses recursive reasoning to explain the behavior of players in games (Camerer et al., 2004). Moreover, the improved decision-making capability offered by recursive reasoning has motivated its application in ML and sequential decision-making problems such as interactive partially observable Markov decision processes (Gmytrasiewicz and Doshi, 2005; Hoang and Low, 2013), multi-agent reinforcement learning (Wen et al., 2019), among others. In this work (Chapter 8), we incorporate recursive reasoning into BO to derive efficient strategies for players in repeated games.

Deep neural networks (DNN) have recently been found to be vulnerable to carefully crafted adversarial examples (Szegedy et al., 2014). Since then, a variety of adversarial attack methods have been developed to exploit this vulnerability of DNN (Goodfellow et al., 2015). However, most of the existing attack methods are *white-box* attacks since they require access to the gradient of the ML model. In contrast, the more realistic *black-box attacks* (Tu et al., 2019; Moon et al., 2019), which we have adopted in our experiments in Chapter 8, only require query access to the target ML model and have been attracting significant attention recently. Of note, BO has recently been used for black-box adversarial attacks (without considering defenses) and demonstrated promising query efficiency (Ru et al., 2020). On the other hand, many attempts have been made to design adversarial defense methods (Madry et al., 2017; Tramèr et al., 2018) to make ML models robust against adversarial attacks. In our experiments, we have adopted the input reconstruction/transformation technique (Meng and Chen, 2017; Samangouei et al., 2018) as the defense mechanism, in which the defender attempts to transform the perturbed input to ensure the correct prediction by the ML model. Refer to the detailed survey of adversarial ML in (Yuan et al.,

2019). Our algorithm, which leverages the combination of BO and recursive reasoning (Chapter 8), can be naturally applied to adversarial ML by modeling the interactions between the attacker and the defender as a repeated game, and hence deriving efficient strategies for both players.

Chapter 4

Bayesian Optimization with Early Stopping

This chapter is based on the following paper published at ICML 2019:

Dai, Z., Yu, H., Low, B. K. H., & Jaillet, P. (2019). Bayesian optimization meets Bayesian optimal stopping. In *Proc. ICML* (pp. 1496-1506).

4.1 Introduction

Many ML models require running an iterative training procedure for some number of epochs such as stochastic gradient descent for neural networks (LeCun et al., 2015) and boosting procedure for gradient boosting machines (Friedman, 2001). When using BO for hyperparameter tuning, any query of a hyperparameter setting usually involves training the ML model for a fixed number of epochs. The information that is typically available during the training process (e.g., validation accuracy after each epoch) might be exploited for improving the epoch efficiency of BO algorithms, specifically, by early-stopping model training under hyperparameter settings that will end up under-performing, hence eliminating unnecessary training epochs. Note that this objective is different from that of

standard early stopping during the training of neural networks, which is used to prevent overfitting. To address this challenging issue, some recent works have been proposed, such as free-thaw BO (Swersky et al., 2014), multi-fidelity BO (Kandasamy et al., 2016, 2017), learning curve prediction (Baker et al., 2017; Domhan et al., 2015; Klein et al., 2017), Hyperband (Li et al., 2017; Falkner et al., 2018), among others (refer to Section 3.1 for more detail). However, we still lack an epoch-efficient algorithm that can incorporate early stopping into BO (i.e., by exploiting information available during the training process) and yet offer a theoretical performance guarantee, the design of which is likely to require a principled decision-making mechanism for determining the optimal stopping time.

Optimal stopping is a classic research topic in statistics and operations research regarding sequential decision-making problems whose objective is to make the optimal stopping decision with a small number of observations (Ferguson, 2006). In *Bayesian optimal stopping* (BOS) or Bayesian sequential design, the decision between stopping vs. continuing is made to maximize the expected utility or, equivalently, minimize the expected loss (Powell and Ryzhov, 2012). BOS has found success in application domains such as finance (Longstaff and Schwartz, 2001), clinical design (Brockwell and Kadane, 2003; Müller et al., 2007; Wathen and Thall, 2008), and economics (Davis and Cairns, 2012). The capability of BOS in providing a principled optimal stopping mechanism makes it a prime candidate for introducing early stopping into BO in a theoretically sound and rigorous way.

This work proposes to unify *Bayesian optimization* (specifically, GP-UCB) *with Bayesian optimal stopping* (BO-BOS) to boost the epoch efficiency of BO (Section 4.3). Intuitively, GP-UCB is acclaimed for being sample-efficient in the number of function evaluations while BOS can reduce the required number of epochs for each function evaluation. BO-BOS unifies the best of both worlds to

yield an epoch-efficient hyperparameter optimization algorithm. Interestingly, in spite of the seemingly disparate optimization objectives of GP-UCB vs. BOS (respectively, objective function v.s. expected loss), BO-BOS can preserve the trademark (asymptotic) no-regret performance of GP-UCB with our specified choice of BOS parameters that is amenable to an elegant interpretation in terms of the exploration-exploitation trade-off (Section 4.4). Though the focus of this work here is on epoch-efficient BO for hyperparameter tuning, we additionally evaluate the performance of BO-BOS empirically in two other interesting applications to demonstrate its generality: policy search for reinforcement learning, and joint hyperparameter tuning and feature selection (Section 4.5).

4.2 Background and Problem Formulation

In this chapter, we denote the input to the objective function f as $\mathbf{z} = [\mathbf{x}, n] \in \mathcal{Z}$, where $\mathbf{x}$ represents a hyperparameter setting and $n \in [1, N]$ is an integer denoting the number of epochs trained for $\mathbf{x}$ where N is the maximum number of epochs. Therefore, in iteration t, the BO-BOS algorithm needs to select a hyperparameter configuration ($\mathbf{x}_t$) and how many epochs to run (n_t), which will combine to form the queried input $\mathbf{z}_t = [\mathbf{x}_t, n_t]$. BOS provides a principled mechanism for making the Bayes-optimal stopping decision with a small number of observations. As shall be seen in Algorithm 4.1, in each iteration t of BO-BOS, BOS is used to early-stop model training under the selected input hyperparameters $\mathbf{x}_t$ that will end up under-performing, hence reducing the required number of training epochs. In a BOS problem, the goal is to decide whether to (a) stop and conclude either hypothesis/event $\theta_t = \theta_{t,1}$ or $\theta_t = \theta_{t,2}$ corresponding to terminal decision d_1 or d_2, or to (b) gather one more observation via the continuation decision d_0. Let $y_{t,n'}$ be the noisy output (validation accuracy) observed in epoch n' and $\mathbf{y}_{t,n} \triangleq [y_{t,n'}]^\top_{n'=1,\ldots,n}$ be a vector of noisy outputs observed up till epoch n in

iteration t. Recall that in iteration t, the ML model is trained using the selected input hyperparameter setting and number of epochs $[\mathbf{x}_t, n_t]$ to yield the noisy observed output (validation accuracy) y_t. So, $y_t = y_{t,n_t}$ for $t = 1, \ldots, T$. After each epoch n, the posterior belief of event θ_t is updated to $\mathbb{P}(\theta_t|\mathbf{y}_{t,n})$ which will be used to compute the expected losses of terminal decisions d_1 and d_2. Such a loss function l has to encode the cost of making a wrong decision. Define $\rho_{t,n}(\mathbf{y}_{t,n})$ as the minimum expected loss among all decisions in epoch n:

$$\begin{aligned}\rho_{t,n}(\mathbf{y}_{t,n}) \triangleq \min\{\ &\mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_1,\theta_t)],\ \mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_2,\theta_t)],\\ &c_{d_0} + \mathbb{E}_{y_{t,n+1}|\mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n+1})]\ \}\end{aligned} \tag{4.1}$$

for $n = N_0 + 1, \ldots, N-1$ where the first two terms are the expected losses of terminal decisions d_1 and d_2, the last term sums the immediate cost c_{d_0} and expected future loss of making the continuation decision d_0 to continue model training in the next epoch $n+1$ to yield the noisy observed output (validation accuracy) $y_{t,n+1}$, and $\rho_{t,N}(\mathbf{y}_{t,N}) \triangleq \min\{\ \mathbb{E}_{\theta_t|\mathbf{y}_{t,N}}[l(d_1,\theta_t)],\ \mathbb{E}_{\theta_t|\mathbf{y}_{t,N}}[l(d_2,\theta_t)]\ \}$. Since $\rho_{t,n}$ depends on $\rho_{t,n+1}$, it naturally prompts the use of backward induction to solve the BOS problem (4.1) exactly, which is unfortunately intractable due to an uncountable set of possible observed outputs $y_{t,n+1}$. This computational difficulty can be overcome using approximate backward induction techniques (Brockwell and Kadane, 2003; Müller et al., 2007) whose main ideas include using summary statistics to represent the posterior beliefs, discretizing the space of summary statistics, and approximating the expectation terms via sampling. Appendix A.1 describes a commonly-used approximate backward induction algorithm (Müller et al., 2007).

Solving the BOS problem (4.1) yields a Bayes-optimal decision rule in each epoch n: Take the Bayes-optimal stopping decision if the expected loss of either

terminal decision d_1 or d_2 is at most that of the continuation decision d_0, that is,

$$\min\{\ \mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_1,\theta_t)],\ \mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_2,\theta_t)]\ \} \leq c_{d_0} + \mathbb{E}_{y_{t,n+1}|\mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n+1})].$$

Otherwise, continue model training to yield the noisy observed output (validation accuracy) $y_{t,n+1}$ and repeat this rule in the next epoch $n+1$.

4.3 BO-BOS Algorithm

In this section, we will describe our proposed BO-BOS algorithm (Section 4.3.1) and define the loss function l in BOS (4.1) such that it can serve as an effective early-stopping mechanism in BO (Section 4.3.2). We focus on problem settings where the objective function f is bounded and monotonically increasing in n:

Assumption 4.1. *(a)* $f(\mathbf{z}) \in [0,1]$ *for all* $\mathbf{z} \in \mathcal{Z}$ *and (b)* $f([\mathbf{x},n]) \leq f([\mathbf{x},n+1])$ *for all* $\mathbf{x}$ *and* $n = 1,\ldots,N-1$.

Assumption 4.1a is not restrictive since it applies to any bounded f with a proper transformation. Assumption 4.1b holds reasonably well in a number of important ML problems: (a) f represents the validation accuracy of an ML model and n denotes the number of training epochs or the number of selected features during feature selection, and (b) f represents the (discounted) cumulative rewards (assuming non-negative rewards) in reinforcement learning (RL) and n denotes the number of steps taken by the agent in the environment. Our experiments in Section 4.5 will demonstrate that BO-BOS outperforms the state-of-the-art hyperparameter optimization algorithms in these ML problems.

4.3.1 Algorithm Description

In each iteration t of BO-BOS (Algorithm 4.1), the input hyperparameters $\mathbf{x}_t$ are selected to maximize the GP-UCB acquisition function with the input dimension

of training epochs fixed at N (line 2). The ML model is trained using $\mathbf{x}_t$ for N_0 initial training epochs to yield the noisy observed outputs (validation accuracies) $\mathbf{y}_{t,N_0}$ (line 3). After that, the BOS problem is solved (line 5) to obtain Bayes-optimal decision rules (see Sections 4.2 and 4.3.2). Then, in each epoch $n > N_0$, model training continues under $\mathbf{x}_t$ to yield the noisy observed output (validation accuracy) $y_{t,n}$ (line 8). If both of the following conditions are satisfied (line 9):

C1. when the BOS decision rule in epoch n outputs the stopping decision;

C2. when $\sigma_{t-1}([\mathbf{x}_t, N]) \leq \kappa\, \sigma_{t-1}([\mathbf{x}_t, n])$,

then model training is early-stopped in epoch $n_t = n$. Otherwise, the above procedure is repeated in epoch $n + 1$. If none of the training epochs $n = N_0 + 1, \ldots, N - 1$ satisfy both C1 and C2, then $n_t = N$ (i.e., no early stopping). Finally, the GP posterior belief is updated with the selected input hyperparameter setting $\mathbf{z}_t = [\mathbf{x}_t, n_t]$ and the corresponding noisy observed output (validation accuracy) $y_t = y_{t,n_t}$ (line 11). BO-BOS then proceeds to the next iteration $t + 1$.

Algorithm 4.1 BO-BOS

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad \mathbf{x}_t \leftarrow \arg\max_{\mathbf{x}} \mu_{t-1}([\mathbf{x}, N]) + \sqrt{\beta_t}\sigma_{t-1}([\mathbf{x}, N])$
3: $\quad$ Train model using $\mathbf{x}_t$ for N_0 epochs to yield $\mathbf{y}_{t,N_0}$
4: $\quad n \leftarrow N_0$
5: $\quad$ Solve BOS problem (4.1) to obtain decision rules
6: $\quad$ **repeat**
7: $\quad\quad n \leftarrow n + 1$
8: $\quad\quad$ Continue model training using $\mathbf{x}_t$ to yield $y_{t,n}$
9: $\quad$ **until** $(n = N) \vee (\text{C1} \wedge \text{C2})$
10: $\quad n_t = n$
11: $\quad$ Update GP posterior belief with $\mathbf{z}_t = [\mathbf{x}_t, n_t]$ and $y_t = y_{t,n_t}$

To understand the rationale of our choices of C1 and C2, the BOS decision rule in C1 recommends the stopping decision to early-stop model training in epoch n if it concludes that model training under $\mathbf{x}_t$ for N epochs will produce a validation accuracy not exceeding the currently found maximum in iterations $1, \ldots, t - 1$; this will be formally described in Section 4.3.2. On the other hand,

C2 prefers to evaluate the validation accuracy f of the ML model with the input query $[\mathbf{x}_t, n]$ of fewer training epochs $n < N$ than $[\mathbf{x}_t, N]$ if the uncertainty of the validation accuracy $f([\mathbf{x}_t, N])$ achieved by model training under $\mathbf{x}_t$ for N epochs is not more than a factor of $\kappa \geq 1$ of that of $f([\mathbf{x}_t, n])$ for n epochs; the degree of preference is controlled by parameter κ. Thus, by satisfying both C1 and C2, C2 lends confidence to the resulting performance of model training under $\mathbf{x}_t$ for N epochs that is concluded by C1 to be underwhelming. So, model training can be early-stopped in epoch $n_t = n$. More importantly, both C1 and C2 are necessary for theoretically guaranteeing the no-regret performance of BO-BOS (Section 4.4).

4.3.2 BOS for Early Stopping in BO

Let the currently found maximum in iterations $1, \ldots, t-1$ be denoted as $y^*_{t-1} \triangleq \max_{t' \in \{1,\ldots,t-1\}} y_{t'}$. In the context of early stopping in BO, BOS has to decide in each epoch n of iteration t whether model training under $\mathbf{x}_t$ for N epochs will produce a validation accuracy not more than the currently found maximum (offset by a noise correction term ξ_t), i.e., $f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t$ where $y^*_0 \triangleq 0$ and ξ_t is defined later in Theorem 4.1. To achieve this, the terminal decisions d_1 and d_2 and the continuation decision d_0 in BOS are defined as follows: d_1 stops and concludes that $f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t$, d_2 stops and concludes that $f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t$, and d_0 continues model training for one more epoch. Then, the event θ (Section 4.2) becomes

$$\theta_t = \begin{cases} \theta_{t,1} & \text{if } f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t \ , \\ \theta_{t,2} & \text{otherwise} \ . \end{cases}$$

We define $l(d_1, \theta_t)$ and $l(d_2, \theta_t)$ of the respective terminal decisions d_1 and d_2 as 0-K loss functions which are commonly used in clinical designs (Jiang et al.,

2013; Lewis and Berry, 1994) due to their simplicity and interpretablility:

$$l(d_1, \theta_t) \triangleq K_1 \mathbb{1}_{\theta_t = \theta_{t,2}} \text{ and } l(d_2, \theta_t) \triangleq K_2 \mathbb{1}_{\theta_t = \theta_{t,1}} \tag{4.2}$$

where the parameters $K_1 > 0$ and $K_2 > 0$ represent the costs of making the wrong terminal decisions d_1 and d_2, respectively. Since $f([\mathbf{x}_t, N])$ is not known in epoch $n < N$, the expected losses of terminal decisions d_1 and d_2 have to be evaluated instead:

$$\begin{aligned} \mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_1, \theta_t)] &= K_1 \mathbb{P}(\theta_t = \theta_{t,2}|\mathbf{y}_{t,n}) \, , \\ \mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_2, \theta_t)] &= K_2 \mathbb{P}(\theta_t = \theta_{t,1}|\mathbf{y}_{t,n}) \, . \end{aligned} \tag{4.3}$$

According to (4.3), if K_1 (K_2) is set to $+\infty$, then $\mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_1, \theta_t)] = +\infty$ ($\mathbb{E}_{\theta_t|\mathbf{y}_{t,n}}[l(d_2, \theta_t)] = +\infty$). Consequently, terminal decision d_1 (d_2) is never recommended. The above definitions are plugged into (4.1) to derive the minimum expected loss $\rho_{t,n}(\mathbf{y}_{t,n})$ in epoch $n = N_0 + 1, \ldots, N$.

Our formulation of the BOS problem (4.1) for early stopping in BO can be solved using an adapted approximate backward induction algorithm: To account for Assumption 4.1b, a kernel with a prior bias towards exponentially decaying learning curves (Swersky et al., 2014) is used to fit a GP model to the validation errors $\mathbf{1} - \mathbf{y}_{t,N_0}$ of the ML model trained for N_0 initial epochs. Samples are then drawn from the resulting GP posterior belief for forward simulation of sample paths from epochs $N_0 + 1$ to N, which are used to estimate the $\mathbb{P}(\theta_t|\mathbf{y}_{t,n})$ and $\mathbb{P}(y_{t,n+1}|\mathbf{y}_{t,n})$ terms necessary for approximate backward induction. Following some applications of BOS (Jiang et al., 2013; Müller et al., 2007), the average validation error is used as the summary statistic. Our adapted approximate backward induction algorithm is explained in detail in Appendix A.2. Note that the use of the kernel favoring exponentially decaying learning curves in generating the forward simulation samples is critical for incorporating our prior knowledge

about the behavior of learning curves, which gives BO-BOS an advantage over multi-fidelity BO algorithms which do not exploit this prior knowledge, thus contributing to the favorable performance of BO-BOS.

After our BOS problem is solved, Bayes-optimal decision rules are obtained and used by C1 in BO-BOS (Algorithm 4.1): Specifically, after model training to yield validation accuracy $y_{t,n}$ (line 8), the summary statistic is first updated to $\sum_{n'=1}^{n} y_{t,n'}/n$ and the BOS decision rule in epoch n recommends a corresponding optimal decision. If the recommended decision is d_1 (i.e., stopping and concluding $f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t$), then model training under $\mathbf{x}_t$ is early-stopped in epoch n (assuming that C2 is satisfied). Otherwise, model training continues under $\mathbf{x}_t$ for one more epoch and the above procedure is repeated in epoch $n+1$ until the last epoch $n = N$ is reached. Note that terminal decision d_2 (i.e., stopping and concluding $f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t$) does not align with the BO objective of sequentially maximizing f. So, when the recommended decision is d_2, there is no early stopping and model training continues under $\mathbf{x}_t$ for one more epoch.

4.4 Theoretical Analysis

The goal of the theoretical analysis is to characterize the growth of the *simple regret* S_T (Section 2.2) of the proposed BO-BOS algorithm and thus show how the algorithm should be designed in order for S_T to asymptotically go to 0, i.e., for the algorithm to be *no-regret*. To account for the additional uncertainty introduced by BOS, we analyze the expected regret, in which the expectation is taken with respect to the posterior probabilities from the BOS algorithms: $\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t | \mathbf{y}_{t,n_t})$.

We make the following assumption on the smoothness of the objective function f:

Assumption 4.2. *Assume for the kernel k, for some a and b,*

$$\mathbb{P}(\sup_{\mathbf{z}\in\mathcal{Z}} |\partial f/\partial z_j| > L) \leq a\exp(-(L/b)^2)$$

for $j = 1, \ldots, d$ where z_j is the j-th component of input $\mathbf{z}$.

Assumption 4.2 is satisfied by some commonly used kernels such as the SE kernel and Matérn kernel with $\nu > 2$ (Srinivas et al., 2010) (Section 2.1). For simplicity, we assume that the underlying domain $\mathcal{Z}$ is discrete, i.e. $|\mathcal{Z}| < \infty$. However, it is straightforward to extend the analysis to general compact domain by following similar analysis strategies as those in Appendix A.2. of (Srinivas et al., 2010). Theorem 4.1 below shows an upper bound on the expected simple regret of the BO-BOS algorithm.

Theorem 4.1. *Suppose that Assumptions 4.1 and 4.2 hold. Let $\delta, \delta', \delta'' \in (0, 1)$, $\beta_t \triangleq 2\log(|\mathcal{Z}|t^2\pi^2/(6\delta))$, and $\tau_T \triangleq \sum_{t=1}^{T} \mathbb{1}_{n_t<N}$ be the number of BO iterations in which early stopping happens from iterations 1 to T. Let κ be the parameter used in C2. At iteration t, the BOS algorithm is run with the corresponding fixed cost parameters K_2 and c_{d_0}, as well as iteration-dependent cost parameter $K_{1,t}$, $\xi_1 \triangleq 0$, and $\xi_t \triangleq \sqrt{2\sigma^2\log(\pi^2t^2(t-1)/(6\delta''))}$ for $t > 1$. Then $\forall T \geq 1$, with probability of at least $1 - \delta - \delta' - \delta''$,*

$$\mathbb{E}[S_T] \leq \frac{\kappa\sqrt{TC_1\beta_T\gamma_T}}{T} + \frac{\sum_{t=1}^{T}\eta_t}{T} + \frac{1}{T}Nb\sqrt{\log\frac{da}{\delta'}}\tau_T$$

in which $\eta_t \triangleq \frac{K_2+c_{d_0}}{K_{1,t}}$, $C_1 = 8/log(1+\sigma^{-2})$, γ_T is the maximum information gain about the function f from any set of observations of size T, and the expectation is w.r.t. $\prod_{t\in\{t'|t'=1,\ldots,T,n_{t'}<N\}} \mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t})$ used in the BOS algorithm.

Theorem 4.2 below states how the BOS parameters should be chosen to make BO-BOS asymptotically no-regret.

Theorem 4.2. *In Theorem 4.1, if $K_{1,t}$ is an increasing sequence such that $K_{1,1} \geq K_2 + c_{d_0}$ and that $K_{1,t}$ goes to $+\infty$ in finite number of BO iterations, then, with probability of at least $1 - \delta - \delta' - \delta''$, $\mathbb{E}[S_T]$ goes to 0 asymptotically.*

The proof of both theorems is presented in Appendix A.3. The first term in the upper bound of $\mathbb{E}[S_T]$ in Theorem 4.1 matches that of the simple regret of the GP-UCB algorithm (up to the constant κ). Note that the theoretical results rely on the exact solution of the BOS problems; however, in practice, a trade-off exists between the quality of the approximate backward induction and the computational efficiency. In particular, increasing the number of forward simulation samples and making the grid of summary statistics more fine-grained both lead to better approximation quality, while increasing the computational cost. Recommended approximation parameters that work well in all our experiments and thus strike a reasonable balance between these two aspects are given in Section 4.5.

Interestingly, the choice of an increasing $K_{1,t}$ sequence as required by Theorem 4.2 is well justified in terms of the exploration-exploitation trade-off. As introduced in section 4.3.2, K_1 represents how much we would like to penalize the BOS algorithm for falsely early-stopping (taking decision d_1). Therefore, increasing values of K_1 implies that, as the BO-BOS algorithm progresses, we become more and more cautious at early-stopping. In other words, the preference of BOS for early stopping diminishes over BO iterations. Interestingly, this corresponds to sequentially shifting our preference from exploration (using small number of epochs) to exploitation (using large number of epochs) throughout all runs of the BOS algorithms, which is an important decision-making principle followed by many sequential decision-making algorithms such as BO, multi-armed bandit, reinforcement learning, among others.

Another intriguing interpretation of the theoretical results is that the growth rate of the $K_{1,t}$ sequence implicitly determines the trade-off between faster convergence of the BO algorithm (smaller number of BO iterations) and more

computational saving in each BO iteration (smaller number of training epochs on average). In particular, if $K_{1,t}$ grows quickly, the second and third terms in the upper bound in Theorem 4.1 both decay fast, since η_t is inversely related to $K_{1,t}$ and large penalty for early stopping results in small τ_T; as a result, a large number of hyperparameters are run with N epochs and the resulted BO-BOS algorithm behaves similarly to GP-UCB, which is corroborated by the upper bound on $\mathbb{E}[S_T]$ in Theorem 4.1 since the first term dominates. On the other hand, if the $K_{1,t}$ sequence grows slowly, then the second and third terms in Theorem 4.1 decay slowly; consequently, these two terms dominate the regret and the resulted algorithm early-stops very often, thus leading to smaller number of epochs on average, at the potential expense of requiring more BO iterations. Furthermore, the constant κ used in C2 also implicitly encodes our relative preference for early-stopping. Specifically, large values of κ favor early stopping by relaxing C2: $\sigma_{t-1}([\mathbf{x}_t, n]) \geq \sigma_{t-1}([\mathbf{x}_t, N])/\kappa$; however, more early-stopped function evaluations might incur larger number of required BO iterations as can be verified by the fact that larger κ increases the first regret term in Theorem 4.1 (which matches the regret of GP-UCB). In practice, as a result of the above-mentioned trade-offs, the best choices of the BOS parameters and κ are application-dependent. In addition, to ensure desirable behaviors of BOS, the BOS parameters should be chosen with additional care. In particular, c_{d_0} should be small, whereas $K_{1,t}$ should be of similar order with K_2 initially. In Section 4.5, we recommend some parameters that work well in all our experiments and thus are believed to perform robustly in practice.

4.5 Experiments and Discussion

The performance of BO-BOS is empirically compared with four other hyperparameter optimization algorithms: GP-UCB (Srinivas et al., 2010), Hyperband

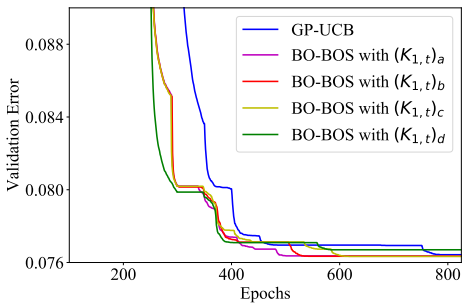


Figure 4.1: Best-found validation error of logistic regression v.s. the total number of epochs (averaged over 10 random initializations). $K_2 = 99$ and $c_{d_0} = 1$ are fixed; $K_{1,1} = 100$ for all four BO-BOS algorithms; for $t > 1$, the different $K_{1,t}$ sequences are: $(K_{1,t})_a = \frac{K_{1,t-1}}{0.89}$; $(K_{1,t})_b = \frac{K_{1,t-1}}{0.95}$; $(K_{1,t})_c = \frac{K_{1,t-1}}{0.99}$; $(K_{1,t})_d = \frac{K_{1,t-1}}{1.0} = K_{1,1}$.

(Li et al., 2017), multi-fidelity BO algorithm called *BO with continuous approximations* (BOCA) (Kandasamy et al., 2017), and GP-UCB with learning curve prediction using an ensemble of Bayesian parametric regression models (LC Prediction) (Domhan et al., 2015). Freeze-thaw BO is not included in the comparison since its implementation details are complicated and not fully available. We empirically evaluate the performance of BO-BOS in hyperparameter optimization of *logistic regression* (LR) and *convolutional neural networks* (CNN), respectively, in Sections 4.5.1 and 4.5.2, and demonstrate its generality in two other interesting applications in Section 4.5.3. Due to lack of space, additional experimental details are deferred to Appendix A.4.

4.5.1 Hyperparameter Optimization of Logistic Regression

We first tune three hyperparameters of LR trained on the MNIST image dataset. Although both the $K_{1,t}$ sequence and κ determine the trade-off between the number of BO iterations and the number of epochs on average as mentioned in section 4.4, for simplicity, we fix $\kappa = 2$ and investigate the impact of different sequences of $K_{1,t}$ values.

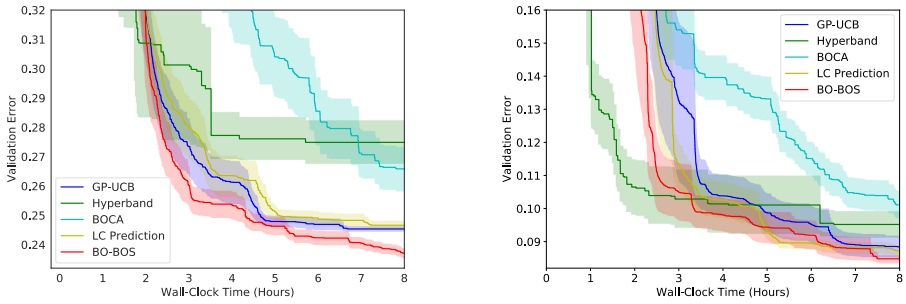
As shown in Fig. 4.1, the sequences $(K_{1,t})_a$, $(K_{1,t})_b$ and $(K_{1,t})_c$ lead to

similar performances, all of which outperform GP-UCB. On the other hand, the algorithm with fixed K_1 values ($(K_{1,t})_d$), despite having fast performance improvement initially, eventually finds a worse hyperparameter setting than all other algorithms. This undesirable performance results from the fact that fixed K_1 values give constant penalty to falsely early-stopping throughout all runs of the BOS algorithms, and as the incumbent validation error decreases, the preference of the algorithm for early stopping will increase, thus preventing the resulting algorithm from beginning to exploit (running promising hyperparameter settings with N epochs). This observation demonstrates the necessity of having an increasing sequence of $K_{1,t}$ values, thus substantiating the practical relevance of our theoretical analysis (Theorem 4.2). The sequence $(K_{1,t})_b$, as well as the values of $K_2 = 99$ and $c_{d_0} = 1$, will be used in the following experiments if not further specified.

4.5.2 Hyperparameter Optimization of Convolutional Neural Networks

In this section, we tune six hyperparameters of CNN using two image datasets: CIFAR-10 (Krizhevsky, 2009) and Street View House Numbers (SVHN) (Netzer et al., 2011). Note that the goal of the experiments is not to compete with the state-of-the-art models, but to compare the efficiency of different hyperparameter tuning algorithms, so no data augmentation or advanced network architectures are used.

As shown in Figures 7.2a and b, BO-BOS outperforms all other algorithms under comparison in terms of the run-time efficiency. Note that the horizontal axis, which represents the wall-clock time, includes all the time spent during the algorithms, including the running time of machine learning models, the approximate backward induction used to solve BOS, etc. Although Hyperband



(a) (b)

Figure 4.2: Best-found validation error of CNN v.s. run-time (averaged over 30 random initializations).

is able to quickly reduce the validation error in the initial stage, it eventually converges to sub-optimal hyperparameter settings compared with both GP-UCB and BO-BOS. Similar findings have been reported in previous works (Klein et al., 2017; Falkner et al., 2018) and they might be attributed to the pure-exploration nature of the algorithm. Our BO-BOS algorithm, which trades off exploration and exploitation, is able to quickly surpass the performance of Hyperband and eventually converges to significantly better hyperparameter settings. In addition, we also ran the BOHB algorithm (Falkner et al., 2018) which combines Hyperband and BO; however, BOHB did not manage to reach comparable performance with the other algorithms in these two tasks. We observed that the sub-optimal behavior of Hyperband and BOHB observed here can be alleviated if the search space of hyperparameters is chosen to be smaller, in which case the advantage of pure exploration can be better manifested. The unsatisfactory performance of BOCA might be explained by the fact that it does not make use of the intermediate validation errors when selecting the number of epochs. In contrast, BO-BOS takes into account the observations after each training epoch when choosing the optimal stopping time, and thus is able to make better-informed decisions. Moreover, since BOCA is designed for general scenarios, the useful assumption of monotonic learning curve utilized by BO-BOS is not exploited; therefore, BOCA is expected to perform better if the fidelity levels result from

data sub-sampling. LC Prediction performs similarly to GP-UCB, which might be because the predicted final values of the learning curves are used as real observations in the GP surrogate function (Domhan et al., 2015), thus invalidating the theoretical guarantee and deteriorating the convergence of GP-UCB, which offsets the computational saving provided by early stopping. BO-BOS, on the other hand, offers theoretically guaranteed convergence, thus allowing explicit control over the trade-off between the speed of convergence and the reduction in the average number of epochs as discussed in Section 4.4.

4.5.3 Novel Applications of the BO-BOS Algorithm

4.5.3.1 Policy Search for RL

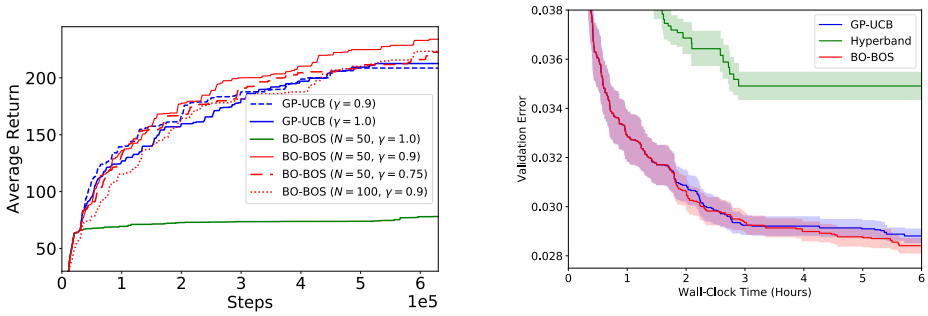
Thanks to its superb sample efficiency, BO has been found effective for policy search in RL (Martinez-Cantin et al., 2007; Wilson et al., 2014), especially when policy evaluation is costly such as gait optimization for robots (Lizotte et al., 2007) and vehicle navigation (Brochu et al., 2010). In policy search, the return of a policy is usually estimated by running the agent in the environment sequentially for a fixed number of steps and calculating the cumulative rewards (Wilson et al., 2014). Thus, the sequential nature of policy evaluation makes BO-BOS an excellent fit to improve the efficiency of BO for policy search in RL.

We apply our algorithm to the Swimmer-v2 task from OpenAI Gym, MuJoCo (Brockman et al., 2016; Todorov et al., 2012), and use a linear policy consisting of 16 parameters. Each episode consists of 1000 steps, and we treat every m consecutive steps as one single epoch such that $N = 1000/m$. Direct application of BO-BOS in this task is inappropriate since the growth pattern of cumulative rewards differs significantly from the evolution of the learning curves of ML models (Appendix A.4.3). Therefore, the rewards are discounted (by γ) when calculating the objective function, because the pattern of discounted return

(cumulative rewards) bears close resemblance to that of learning curves. Note that although the value of the objective function is the discounted return, we also record and report the corresponding un-discounted return, which is the ultimate objective to be maximized. As a result, N and γ should be chosen such that the value of discounted return faithfully aligns with its un-discounted counterpart. Fig. 4.3a plots the best (un-discounted) return in an episode against the total number of steps, in which BO-BOS (with $N = 50$ and $\gamma = 0.9$) outperforms GP-UCB (for both $\gamma = 0.9$ and $\gamma = 1.0$). The observation that the solid red line shows better returns than the two dotted red lines might be because overly small γ (0.75) and overly large N (100) both enlarge the disparity between discounted and un-discounted returns since they both downplay the importance of long-term discounted rewards. Moreover, not discounting the rewards ($\gamma = 1$) leads to poor performance, corroborating the earlier analysis motivating the use of discounted rewards. The results demonstrate that even though BO-BOS is not immediately applicable in the original problem, it can still work effectively if the problem is properly transformed, which substantiates the general applicability of BO-BOS. Moreover, we also applied Hyperband in this task, but it failed to converge to a comparable policy to the other methods (achieving an average return of around 2.5), which further supports our earlier claim stating that Hyperband under-performs when the search space is large because there are significantly more parameters (16) than the previous tasks.

Interestingly, both BO-BOS and GP-UCB use significantly less steps to substantially outperform the benchmarks[1] achieved by some recently developed deep RL algorithms, in which the best-performing algorithm (Deep Deterministic Policy Gradients) achieves an average return of around 120. Although the simplicity of the task might have contributed to their overwhelming performances, the results highlight the considerable potential of BO-based policy search algorithms

[1]https://spinningup.openai.com/en/latest/spinningup/bench.html



(a) (b)

Figure 4.3: (a) Best-found return (averaged over 5 episodes) v.s. the total number of steps of the robot in the environment (averaged over 30 random initializations) using the Swimmer-v2 task. The BO-BOS algorithm is run with different values of N (the maximum number of epochs) and γ (the discount factor). (b) Best-found validation error of XGBoost v.s. run-time (averaged over 30 random initializations), obtained using joint hyperparameter tuning and feature selection.

for RL. Note that following the common practice in RL, Fig. 4.3a is presented in terms of the total number of steps instead of run-time; we believe the additional computation required by BOS can be easily overshadowed in large-scale RL tasks, as demonstrated in the previous experiments. Most modern RL algorithms rely on enormous number of samples, making their applications problematic when sample efficiency is of crucial importance (Arulkumaran et al., 2017). As shown above, BO-BOS achieves significantly better results than popular deep RL algorithms while at the same time being far more sample-efficient, thus potentially offering practitioners a practically feasible option in solving large-scale RL problems.

4.5.3.2 Joint Hyperparameter Tuning and Feature Selection

Beside hyperparameter tuning, feature selection is another important pre-processing step in ML, to lower computational cost and boost model performance (Hall, 2000). There are two types of feature selection techniques: filter and wrapper methods; the wrapper methods, such as forward selection (which starts with an empty feature set and in each iteration greedily adds the feature with

largest performance improvement), have been shown to perform well, although computationally costly (Hall and Smith, 1999). Interestingly, as a result of the sequential nature of wrapper methods, they can be naturally solved by BO-BOS by simply replacing the sequential training of ML models with forward selection.

In this task, we tune four hyperparameters of the gradient boosting model (XGBoost (Chen and Guestrin, 2016)) trained on an email spam dataset. We additionally compare with Hyperband since it was previously applied to random feature approximation in kernel methods (Li et al., 2017). As shown in Fig. 4.3b, BO-BOS again delivers the best performance in this application. Consistent with Fig. 7.2, the wall-clock time includes all the time incurred during the algorithms. The $K_{1,t}$ sequence is made smaller than before: $K_{1,t} = K_{1,t-1}/0.99$, because in this setting, more aggressive early stopping is needed for BO-BOS to show its advantage. Although Hyperband works well for random feature approximation (Li et al., 2017), it does not perform favourably when applied to more structured feature selection techniques. Both hyperparameter optimization and feature selection have been shown to be effective for enhancing the performance of ML models. However, performing either of them in isolation may lead to sub-optimal performance since their interaction is un-exploited. Our results suggest that BO-BOS can effectively improve the efficiency of joint hyperparameter tuning and feature selection, making the combined usage of these two pre-processing techniques a more practical choice.

4.6 Conclusion

In this work, we present a unifying framework, BO-BOS, that integrates BOS into BO in a natural way, to derive a principled mechanism for optimally stopping hyperparameter evaluations during BO. We analyze the regret of the algorithm, and derive the BOS parameters that make the resulting BO-BOS algorithm

no-regret. Applications of BO-BOS to hyperparameter tuning of ML models, as well as two other novel applications, demonstrate the practical effectiveness of the algorithm.

Chapter 5

Federated Bayesian Optimization

This chapter is based on the following paper published at NeurIPS 2020:

Dai, Z., Low, B. K. H., & Jaillet, P. (2020). Federated Bayesian optimization via Thompson sampling. In *Proc. NeurIPS*.

5.1 Introduction

Bayesian optimization (BO) has recently become a prominent method for optimizing computationally costly black-box functions with no access to gradients, such as hyperparameter tuning of *deep neural networks* (DNN) (Shahriari et al., 2016). The rapidly growing computational capability of edge devices such as mobile phones, as well as increasing concerns over data privacy, has given rise to the widely celebrated paradigm of *federated learning* (FL) (McMahan et al., 2017), also known as *federated optimization* (Li et al., 2020d). In FL, individual agents, without transmitting their raw data, attempt to leverage the contributions from the other agents to more effectively optimize the parameters of their machine learning (ML) model (e.g., DNN) through *first-order optimization* techniques (e.g., stochastic gradient descent) (Kairouz et al., 2019; Li et al., 2019b). However, some common ML tasks such as hyperparameter tuning of

DNN lack access to gradients and thus require *zeroth-order optimization* (i.e., black-box optimization), and a recent survey (Kairouz et al., 2019) pointed out that hyperparameter optimization of DNN models in the FL setting is one of the promising research directions for FL. This opportunity, combined with the proven capability of BO to efficiently optimize computationally intensive black-box functions (Shahriari et al., 2016), naturally suggests the potential of extending BO to the FL setting, which we refer to as *federated BO* or FBO. Of note, to the best of our knowledge, our work in this chapter represents the first attempt to utilize zeroth-order information in the federated setting.

The setting of our FBO is similar to that of FL, except that FBO uses zeroth-order optimization, in contrast to first-order optimization adopted in FL. In FBO, every agent uses BO to optimize a black-box function (e.g., hyperparameter optimization of a DNN model) and attempts to improve the efficiency of its BO task by incorporating the information from other agents. The information exchange between agents has to take place without directly transmitting the raw data of their BO tasks (i.e., history of input-output pairs). A motivating example is when a number of mobile phone users collaborate in optimizing the hyperparameters of their separate DNN models used for next-word prediction in a smart keyboard application, without sharing the raw data of their own hyperparameter optimization tasks. This application cannot be handled by FL due to the lack of gradient information, and thus calls for FBO. Note that the generality of BO as a black-box optimization algorithm makes the applicability of FBO extend beyond hyperparameter tuning of DNN on edge devices. For example, hospitals can be agents in an FL system (Kairouz et al., 2019); when a hospital uses BO to select the patients to perform a medical test (Yu et al., 2015), FBO might be employed to help the hospital accelerate its BO task using the information from other hospitals without requiring their raw data. An important difference between FBO and FL is that FBO is more suitable for optimizing

expensive-to-evaluate functions (e.g., hyperparameter optimization) for which only a relatively small number of iterations is feasible, whereas FL usually involves a large number of iterations. Despite its promising applications, FBO faces a number of major challenges, some of which are only present in FBO, while others plague the FL setting in general.

The first challenge, which arises only in FBO yet not FL, results from the requirement for retaining (hence not transmitting) the raw data. In standard FL, the transmitted information consists of the parameters of DNN (McMahan et al., 2017), which reduces the risk of privacy violation compared to passing the raw data. In BO, the information about a BO task is contained in the *surrogate model*, which is used to model the objective function and hence guide the query selection (Section 5.2). However, unlike DNN, the *Gaussian process* (GP) model (Rasmussen and Williams, 2006), which is the most commonly used surrogate model in BO, is *nonparametric*. Therefore, a BO task has no parameters (except for the raw data of BO) that can represent the GP surrogate and thus be exchanged between agents, while the raw data of BO should be retained and never transmitted (Kusner et al., 2015). To overcome this challenge, we exploit *random Fourier features* (RFF) (Rahimi and Recht, 2007) to approximate a GP using a Bayesian linear regression model. This allows us to naturally derive parameters that contain the information about the approximate GP surrogate and thus can be communicated between agents without exchanging the raw data (Section 5.2). In fact, with RFF approximation, the parameters to be exchanged in FBO are equivalent to those of a linear model in standard FL (Section 5.3.2).

FBO also needs to handle some common challenges faced by FL in general: the communication efficiency and heterogeneity of agents. Firstly, communication efficiency is an important factor in the FL setting since a large number of communicated parameters places a demanding requirement on the communication bandwidth (Kairouz et al., 2019) and is also more vulnerable to potential malicious

privacy attacks (Chang et al., 2019). To this end, we use *Thompson sampling* (TS) (Thompson, 1933), which has been recognized as a highly effective practical method (Chapelle and Li, 2011), to develop our FBO algorithm. The use of TS reduces the required number of parameters to be communicated while maintaining competitive performances (Section 5.3.2). Secondly, the heterogeneity of agents is an important practical consideration in FL since different agents might have highly disparate properties (Li et al., 2019b). In FBO, heterogeneous agents represent those agents whose objective functions are significantly different from that of the *target agent* (i.e., the agent performing BO). For example, the optimal hyperparameters of the next-word prediction DNN model may vary significantly across agents as a result of the distinct typing habits of different mobile phone users. To address this challenge, we derive a theoretical convergence guarantee for our algorithm which is *robust against heterogeneous agents*. In particular, our algorithm is asymptotically *no-regret* even when some or all other agents have highly different objective functions from the target agent.

In this work, we introduce the first algorithm for the FBO setting, *federated Thompson sampling* (FTS), which is both theoretically principled and practically effective. We provide a theoretical convergence guarantee for FTS that is robust against heterogeneous agents (Section 5.4). We demonstrate the empirical effectiveness of FTS in terms of communication efficiency, computational efficiency and practical performance, using a landmine detection experiment and two activity recognition experiments using Google glasses and mobile phone sensors (Section 5.5).

5.2 Background and Problem Formulation

5.2.1 Gaussian Processes with Random Fourier Features Approximation.

GPs are known to suffer from poor scalability ($\mathcal{O}(t^3)$) and thus calls for approximation techniques. Bochner's theorem states that any continuous stationary kernel k (e.g., the SE kernel) can be expressed as the Fourier integral of a spectral density $p(\mathbf{s})$ (Rasmussen and Williams, 2006). As a result, random samples can be drawn from $p(\mathbf{s})$ to construct the M-dimensional ($M \geq 1$) *random features* $\boldsymbol{\phi}(\mathbf{x}), \forall x \in \mathcal{X}$ (Appendix B.1), whose inner product can be used to approximate the kernel values: $k(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}'), \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$ (Rahimi and Recht, 2007). The approximation quality of this technique, referred to as *random Fourier features* (RFF) approximation, is theoretically guaranteed with high probability: $\sup_{\mathbf{x},\mathbf{x}' \in \mathcal{X}} |k(\mathbf{x}, \mathbf{x}') - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')| \leq \epsilon$, in which $\epsilon = \mathcal{O}(M^{-1/2})$ (Rahimi and Recht, 2007). Therefore, more random features (larger M) results in a better approximation (smaller ϵ). In this work, we focus on the widely used *Squared Exponential* (SE) kernel.

A GP with RFF approximation can be interpreted as a Bayesian linear regression model with $\boldsymbol{\phi}(\mathbf{x})$ as the features: $\hat{f}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}$. With the prior of $\mathbb{P}(\boldsymbol{\omega}) = \mathcal{N}(0, \mathbf{I})$ and given the set of observations D_t, the posterior distribution of $\boldsymbol{\omega}$ can be derived as:

$$\mathbb{P}(\boldsymbol{\omega} | \boldsymbol{\Phi}(\mathbf{X}_t), \mathbf{y}_t) = \mathcal{N}(\boldsymbol{\nu}_t, \sigma^2 \boldsymbol{\Sigma}_t^{-1}), \tag{5.1}$$

in which $\boldsymbol{\Phi}(\mathbf{X}_t) = [\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_t)]^\top$ is a $t \times M$-dimensional matrix, $\mathbf{y}_t = [y(\mathbf{x}_1), \ldots, y(\mathbf{x}_t)]^\top$ is a $t \times 1$ column vector, and

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Phi}(\mathbf{X}_t)^\top \boldsymbol{\Phi}(\mathbf{X}_t) + \sigma^2 \mathbf{I}, \qquad \boldsymbol{\nu}_t = \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\Phi}(\mathbf{X}_t)^\top \mathbf{y}_t, \tag{5.2}$$

which contain M^2 and M parameters respectively. As a result, we can sample a function $\tilde{f}$ from the approximate GP with RFF approximation by firstly sampling $\tilde{\boldsymbol{\omega}}$ from the posterior (5.1), and then setting $\tilde{f}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \tilde{\boldsymbol{\omega}}, \forall \mathbf{x} \in \mathcal{X}$. Moreover, $\boldsymbol{\Sigma}_t$ and $\boldsymbol{\nu}_t$ (5.2) fully define the posterior predictive distribution of the approximate GP at any input $\mathbf{x}$, which is Gaussian-distributed with the mean and variance: $\hat{\mu}_t(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\nu}_t, \hat{\sigma}_t^2(\mathbf{x}) = \sigma^2 \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\phi}(\mathbf{x})$ (Appendix B.2).

5.2.2 Problem Setting of Federated Bayesian Optimization.

Assume that there are $N+1$ agents in the system: $\mathcal{A}$ and $\mathcal{A}_1, \ldots, \mathcal{A}_N$. For ease of exposition, we focus on the perspective of $\mathcal{A}$ as the *target agent*, i.e., $\mathcal{A}$ attempts to use the information from agents $\mathcal{A}_1, \ldots, \mathcal{A}_N$ to accelerate its BO task. We denote $\mathcal{A}$'s objective function as f and a sampled function from $\mathcal{A}$'s GP posterior at iteration t as f_t. We represent $\mathcal{A}_n$'s objective function as g_n, and a sampled function from $\mathcal{A}_n$'s GP posterior with RFF approximation as $\hat{g}_n$. We assume that all agents share the same set of random features $\boldsymbol{\phi}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$, which is easily achievable since it is equivalent to sharing the first layer of a neural network in FL (Appendix B.1). For theoretical analysis, we assume that all objective functions are defined on the same domain $\mathcal{X} \subset \mathbb{R}^d$, which is assumed to be discrete for simplicity but our analysis can be easily extended to compact domain through discretization (Chowdhury and Gopalan, 2017). A smoothness assumption on these functions is required for theoretical analysis, so we assume that they have bounded norm induced by the *reproducing kernel Hilbert space* (RKHS) associated with the kernel k: $\|f\|_k \leq B$ and $\|g_n\|_k \leq B, \forall n = 1, \ldots, N$. This further suggests that the absolute function values are upper-bounded: $|f(\mathbf{x})| \leq B$ and $|g_n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}$. We denote the maximum difference between f and g_n as d_n: $d_n = \max_{\mathbf{x} \in \mathcal{X}} |f(\mathbf{x}) - g_n(\mathbf{x})|$, which characterizes the similarity between f and g_n. A smaller d_n implies that f and g_n are more similar, and heterogeneous agents are those whose d_n's are large. We use t_n to denote number

of BO iterations $\mathcal{A}_n$ has completed (i.e., the number of observations of $\mathcal{A}_n$) when it passes information to $\mathcal{A}$; t_n's are constants unless otherwise specified.

5.3 Federated Bayesian Optimization (FBO)

5.3.1 Federated Thompson Sampling (FTS)

Before agent $\mathcal{A}$ starts to run a new BO task, it can request for information from the other agents $\mathcal{A}_1, \ldots, \mathcal{A}_N$. Next, every agent $\mathcal{A}_n, \forall n = 1, \ldots, N$ uses its own history of observations, as well as the shared random features (Section 5.2), to calculate the posterior distribution $\mathcal{N}(\boldsymbol{\nu}_n, \sigma^2 \boldsymbol{\Sigma}_n^{-1})$ (5.1), in which $\boldsymbol{\nu}_n$ and $\boldsymbol{\Sigma}_n$ represent $\mathcal{A}_n$'s parameters of RFF approximation (5.2); next, $\mathcal{A}_n$ draws a sample from the posterior distribution: $\boldsymbol{\omega}_n \sim \mathcal{N}(\boldsymbol{\nu}_n, \sigma^2 \boldsymbol{\Sigma}_n^{-1})$, and then passes $\boldsymbol{\omega}_n$ (an M-dimensional vector) to the target agent $\mathcal{A}$ (possibly via a central server). After receiving the messages from other agents, $\mathcal{A}$ uses them to start the FTS algorithm (Algorithm 5.1). To begin with, $\mathcal{A}$ needs to define (a) a monotonically increasing sequence $[p_t]_{t \geq 1}$: $p_t \in (0, 1], \forall t \geq 1$ and $p_t \to 1$ as $t \to +\infty$, and (b) a discrete distribution P_N over the agents $\mathcal{A}_1, \ldots, \mathcal{A}_N$: $P_N[n] \in [0, 1], \forall n = 1, \ldots, N$ and $\sum_{n=1}^{N} P_N[n] = 1$. In iteration $t \geq 1$ of FTS, with probability p_t (line 4 of Algorithm 5.1), $\mathcal{A}$ samples a function f_t using its current GP posterior and chooses $\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t(\mathbf{x})$; with probability $1 - p_t$ (line 6), $\mathcal{A}$ firstly samples an agent $\mathcal{A}_n$ from P_N, and then chooses $\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} \hat{g}_n(\mathbf{x})$ where $\hat{g}_n(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_n$ corresponds to a sampled function from the GP posterior of $\mathcal{A}_n$ with RFF approximation. Next, $\mathbf{x}_t$ is queried to produce $y(\mathbf{x}_t)$, and FTS proceeds to the next iteration $t + 1$.

Interestingly, FTS (Algorithm 5.1) can be interpreted as a variant of *TS with a mixture of GPs*. That is, in each iteration t, we firstly sample a GP: the GP of $\mathcal{A}$ is sampled with probability p_t, and the GP of $\mathcal{A}_n$ is sampled with

[1] β_t will be defined in Theorem 5.1 in Section 5.4.

Algorithm 5.1 Federated Thompson Sampling

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Sample r from the Uniform distribution in $[0, 1]$: $r \sim U(0, 1)$
3: **if** $r \leq p_t$ **then**
4: Sample $f_t \sim \mathcal{GP}(\mu_{t-1}(\cdot), \beta_t^2 \sigma_{t-1}^2(\cdot, \cdot))$,[1] and choose $\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t(\mathbf{x})$
5: **else**
6: Sample agent $\mathcal{A}_n$ from the distribution P_N, and choose $\mathbf{x}_t = \arg\max_{\mathbf{x} \in \mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_n$
7: Query $\mathbf{x}_t$ to observe $y(\mathbf{x}_t)$, and update GP posterior with $(\mathbf{x}_t, y(\mathbf{x}_t))$

probability $(1 - p_t) P_N[n]$ for all $n = 1, \ldots, N$. Next, we draw a function from the sampled GP, whose maximizer is selected to query. As a result, $\mathbf{x}_t$ follows the same distribution as the maximizer of the mixture of GPs, and the mixture model gradually converges to the GP of $\mathcal{A}$ as $p_t \to 1$. Furthermore, our FTS algorithm also shares similarities with the work of Hoffman et al. (2011) which has also proposed sampling an acquisition function from a mixture of acquisition functions. The sequence $[p_t]_{t \geq 1}$ controls the degree to which the information from the other agents is exploited, such that decreasing the value of this sequence encourages the utilization of these information. The distribution P_N decides the preferences for different agents. A natural choice for P_N is the uniform distribution $P_N[n] = 1/N, \forall n = 1, \ldots, N$ indicating equal preferences for all agents, which is a common choice when we have no knowledge regarding which agents are more similar to the target agent. In FTS, *stragglers*[2] can be naturally dealt with by simply assigning 0 to the corresponding agent $\mathcal{A}_n$ in the distribution P_N such that $\mathcal{A}_n$ is never sampled (line 6 of Algorithm 5.1). Therefore, FTS is robust against communication failure which is a common issue in FL (Li et al., 2020d).

Since only one message ($\boldsymbol{\omega}_n$) is received from each agent *before* the beginning of FTS, once an agent $\mathcal{A}_n$ is sampled and its message $\boldsymbol{\omega}_n$ is used (line 6 of Algorithm 5.1), we remove it from P_N by setting the corresponding element to 0,

[2]Stragglers refer to those agents whose information is not received by the target agent (Li et al., 2020d).

and then re-normalize P_N. However, FTS can be easily generalized to allow $\mathcal{A}$ to receive information from each agent after every iteration (or every few iterations), such that every agent can be sampled multiple times. This more general setting requires more rounds of communication. In practice, FTS is expected to perform similarly in both settings when (a) the number of agents N is large (i.e., a common assumption in FL) and (b) P_N gives similar or equal preferences to all agents, such that the probability of an agent being sampled more than once is small. Furthermore, this setting can be further generalized to encompass the scenario where multiple (even all) agents are concurrently performing optimization tasks using FTS. In this case, the information received from $\mathcal{A}_n$ can be updated as $\mathcal{A}_n$ collects more observations, i.e., t_n may increase as updated information is received from $\mathcal{A}_n$.

5.3.2 Comparison with Other BO Algorithms Modified for the FBO Setting

Although FTS is the first algorithm for the FBO setting, some algorithms for *meta-learning* in BO, such as *ranking-weighted Gaussian process ensemble* (RGPE) (Feurer et al., 2018) and *transfer acquisition function* (TAF) (Wistuba et al., 2018), can be adapted to the FBO setting through a heuristic combination with RFF approximation. Meta-learning aims to use the information from previous tasks to accelerate the current task. Specifically, both RGPE and TAF use a separate GP surrogate to model the objective function of every agent (previous task), and use these GP surrogates to accelerate the current BO task. To modify both algorithms to suit the setting of FBO, firstly, every agent $\mathcal{A}_n$ applies RFF approximation to its own GP surrogate, and then passes the resulting parameters $\boldsymbol{\nu}_n$ and $\boldsymbol{\Sigma}_n^{-1}$ (Section 5.2) to the target agent $\mathcal{A}$. Next, after receiving $\boldsymbol{\nu}_n$ and $\boldsymbol{\Sigma}_n^{-1}$ from the other agents, $\mathcal{A}$ can use them to calculate the approximate

GP surrogate of each agent (Section 5.2), which can then be plugged into the original RGPE/TAF algorithm.[3] However, unlike FTS, RGPE and TAF are not equipped with theoretical convergence guarantee, and thus lack an assurance to guarantee consistent performances in the presence of heterogeneous agents. Moreover, as we analyze below and will show in Section 5.5, FTS outperforms both RGPE and TAF in a number of major aspects in our experiments, including communication efficiency, computational efficiency and practical performance.

Firstly, regarding communication efficiency, both RGPE and TAF require $\boldsymbol{\nu}_n$ and $\boldsymbol{\Sigma}_n^{-1}$ ($M + M^2$ parameters) from each agent since both the posterior mean and variance of every agent are needed. Moreover, TAF additionally requires the incumbent (currently found maximum observation value) of every agent, which could further increase the risk of privacy leak. In a given experiment and for a fixed M, our FTS algorithm is superior in terms of communication efficiency since it only requires an M-dimensional vector ($\boldsymbol{\omega}_n$) from each agent, which is equivalent to standard FL using a linear model with M parameters. Secondly, FTS is also advantageous in terms of computational efficiency. When $\mathbf{x}_t$ is selected using $\boldsymbol{\omega}_n$ from an agent, FTS only needs to solve the optimization problem of $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_n$ (line 6 of Algorithm 5.1), which incurs minimal computational cost[4]; when $\mathbf{x}_t$ is selected by maximizing a sampled function from $\mathcal{A}$'s GP posterior (line 4 of Algorithm 5.1), this maximization step can also be approximated by RFF, which is also computationally cheap. In contrast, for both RGPE and TAF, every evaluation of the acquisition function (which is maximized to select $\mathbf{x}_t$) at an input $\mathbf{x} \in \mathcal{X}$ requires calculating the posterior mean and variance using the GP surrogate of *every* agent at $\mathbf{x}$. Therefore, their required computation in every iteration grows linearly in the number of agents (N), and thus can become prohibitive when N is large. We also empirically verify this in

[3]Refer to (Feurer et al., 2018) and (Wistuba et al., 2018) for more details about RGPE and TAF.

[4]We use the DIRECT method for this optimization problem, which takes on average 0.76 seconds per iteration (landmine detection experiment, Section 5.5.2).

our experiments (Fig. 5.3d in Section 5.5.2).

5.4 Theoretical Analysis

In our theoretical analysis, since we allow the presence of heterogeneous agents (i.e., the other agents may have significantly different objective functions from the target agent), we do not aim to show that FTS achieves a faster convergence than standard TS, and instead prove a convergence guarantee that is robust against heterogeneous agents. This is consistent with most works proving the convergence of FL algorithms (Li et al., 2018, 2020d), and makes the theoretical results more generally applicable since the presence of heterogeneous agents is a major and inevitable challenge of FL and FBO. Note that we analyze FTS in the more general setting, in which communication is allowed before every iteration instead of only before the first iteration. However, as discussed in Section 5.3.1, FTS behaves similarly in both settings in the common scenario when N is large and P_N assigns similar probabilities to all agents. Theorem 5.1 below is our main theoretical result (proof in Appendix B.3).

Theorem 5.1. *Denote by γ_t the maximum information gain about f from any set of t observations. Let $\delta \in (0,1)$, $\beta_t = B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)}$ and $c_t = \beta_t(1 + \sqrt{2\log(|\mathcal{X}|t^2)})$ for $t \geq 1$. Choose $[p_t]_{t\geq 1}$ as a monotonically increasing sequence satisfying: $p_t \in (0,1], \forall t \geq 1$, $p_t \to 1$ as $t \to +\infty$, and $(1-p_t)c_t \leq (1-p_1)c_1, \forall t \geq 2$. For FTS, we have with probability of $\geq 1-\delta$ that*[5]

$$R_T = \tilde{\mathcal{O}}\Big((B + 1/p_1)\gamma_T\sqrt{T} + \sum\nolimits_{t=1}^{T}\psi_t\Big),$$

where $\psi_t = 2(1-p_t)\sum_{n=1}^{N} P_N[n]\Delta_{n,t}$, and $\Delta_{n,t} = \tilde{\mathcal{O}}(M^{-1/2}Bt_n^2 + B + \sqrt{\gamma_{t_n}} + \sqrt{M} + d_n + \sqrt{\gamma_t})$.

[5] $\tilde{\mathcal{O}}$ ignores all logarithmic factors.

For the SE kernel which we focus on in this chapter, the first term in the upper bound is sublinear in T since $\gamma_T = \mathcal{O}((\log T)^{d+1})$ for the SE kernel (Srinivas et al., 2010). Moreover, since the sequence of $[p_t]_{t\geq 1}$ is chosen to be monotonically increasing and goes to 1 when $t \to \infty$, $1 - p_t$ goes to 0 asymptotically. Therefore, the second term in the upper bound also grows sublinearly.[6] For example, if $[p_t]_{t\geq 1}$ is chosen such that $1 - p_t = \mathcal{O}(1/\sqrt{t})$, $\sum_{t=1}^{T} \psi_t = \tilde{\mathcal{O}}(\sqrt{T})$. As a result, FTS achieves *no regret* asymptotically regardless of the difference between the target agent and the other agents, which is a highly desirable property for FBO in which the heterogeneity among agents is a prominent challenge. Such a robust regret upper bound is achieved because we upper-bound the worst-case error for *any* set of agents (i.e., any set of values of d_n and t_n for $n = 1, \ldots, N$) in the proof. The robust nature of the regret upper bound can be reflected in its dependence on the sequence $[p_t]_{t\geq 1}$, as well as d_n and t_n. When the value of the sequence $[p_t]_{t\geq 1}$ is small, i.e., when the information from the other agents is exploited more (Section 5.3.1), the worst-case error due to more utilization of these information is also increased. This is corroborated by Theorem 5.1 since smaller values of p_t increase the regret upper bound through the terms $1/p_1$ and $(1 - p_t)$ in ψ_t. Theorem 5.1 also shows that the regret bound becomes worse with larger values of d_n and t_n, because a larger d_n increases the difference between the objective functions of $\mathcal{A}_n$ and $\mathcal{A}$, and more observations from $\mathcal{A}_n$ (a larger t_n) also makes the upper bound looser since for a fixed d_n, a larger number of observations increases the worst-case error by accumulating the individual errors[7].

The dependence of the regret upper bound (through $\Delta_{n,t}$) on the number of random features, M, is particularly interesting due to the interaction between two

[6] Recall we have mentioned in Section 2.2 that for the SE kernel, $\sqrt{\gamma_t}$ in $\Delta_{n,t}$ is logarithmic in t: $\sqrt{\gamma_t} = \mathcal{O}((\log t)^{(D+1)/2})$.

[7] In the most general setting where $\mathcal{A}_n$ may collect more observations between different rounds of communication such that t_n may increase (Section 5.3.1), $1 - p_t$ can decay faster to preserve the no-regret convergence.

opposing factors. Firstly, the term $M^{-1/2}Bt_n^2$ arises since better approximation quality of the agent's GP surrogates (i.e., larger M) improves the performance. However, the term $\sqrt{M}$ suggests the presence of another factor with an opposite effect. This results from the need to upper-bound the distance between $\boldsymbol{\omega}_n$ (i.e., an M-dimensional Gaussian random variable) and its mean $\boldsymbol{\nu}_n$ (5.1), which grows at a rate of $\mathcal{O}(\sqrt{M})$ (Lemma B.3 in Appendix B.3). Taking the derivative of both terms w.r.t. M reveals that the regret bound is guaranteed to become tighter with increasing M (i.e., the effect of the term $M^{-1/2}Bt_n^2$ is more dominant) when t_n is sufficiently large, i.e., when $t_n = \Omega(\sqrt{M/B})$. An intuitive explanation of this finding, which is verified in our experiments (Section 5.5.1), is that the positive effect (i.e., tighter regret bound) of better RFF approximation (larger M) is amplified when more observations are available (i.e., t_n is large). In contrast, when t_n is small, minimal information is offered by agent $\mathcal{A}_n$ and thus increasing the quality of RFF approximation only leads to marginal or negligible improvement in the performance. The practical implication of this insight is that when the other agents only have a small number of observations, it is not recommended to use a large number of random features since it requires a larger communication bandwidth (Section 5.3.2) yet is unlikely to improve the performance.

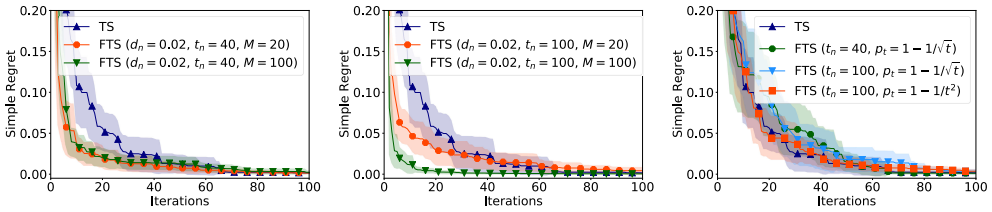
5.5 Experiments and Discussion

We firstly use synthetic functions to investigate the behavior of FTS. Next, using 3 real-world experiments, we demonstrate the effectiveness of FTS in terms of communication efficiency, computational efficiency and practical performance. Since it has been repeatedly observed that the theoretical choice of β_t that is used to establish the confidence interval is overly conservative (Bogunovic et al., 2018; Srinivas et al., 2010), we set it to a constant: $\beta_t = 1.0$. As a result, c_t (Theorem 5.1)

grows slowly (logarithmic) and thus we do not explicitly check the validity of the condition $(1-p_t)c_t \leq (1-p_1)c_1, \forall t \geq 2$. All error bars represent standard errors. For simplicity, here we focus on the simple setting where communication happens only before the beginning of FTS (Section 5.3.1). However, in Appendix B.4.2.1, we also evaluate our performance in the most general setting where the other agents are also performing optimization tasks such that they may collect more observations between different rounds of communication (i.e., t_n is increasing). The results (Fig. B.1 in Appendix B.4.2.1) show consistent performances of FTS in both settings. More experimental details and results are deferred to Appendix C.2 due to space constraint.

5.5.1 Optimization of Synthetic Functions

In synthetic experiments, the objective functions are sampled from a GP (defined on a 1-D discrete domain within $[0, 1]$) using the SE kernel and scaled into the range $[0, 1]$. We fix the total number of agents as $N = 50$, and vary d_n, t_n and M to explore their impacts on the performance. We use the same d_n and t_n for all agents for simplicity. We choose P_N to be uniform: $P_N[n] = 1/N, \forall n = 1, \ldots, N$, and choose the sequence $[p_t]_{t\geq 1}$ as: $p_t = 1 - 1/\sqrt{t}, \forall t \geq 2$ and $p_1 = p_2$. Figs. D.1a and b show that when $d_n = 0.02$ is small, FTS is able to perform better than TS. Intuitively, the performance advantage of FTS results from its ability to exploit the additional information from the other agents to reduce the need for exploration. These results also reveal that when t_n of every agent is small (Fig. D.1a), the impact of M is negligible; on the other hand, when t_n is large (Fig. D.1b), increasing M leads to evident improvement in the performance. This corroborates our theoretical analysis (Section 5.4) stating that when t_n is large, increasing the value of M is more likely to tighten the regret bound and hence improve the performance. Moreover, comparing the green curves in Figs. D.1a and b shows that when the other agents' objective functions are similar to the target



(a) (b) (c)

Figure 5.1: Simple regret in synthetic experiments, showing the impact of M when t_n is (a) small and (b) large, and (c) the performance when $d_n = 1.2$ is large. Each curve is averaged over 5 randomly sampled functions from a GP, and 5 random initializations of 1 input for each function.

agent's objective function (i.e., $d_n = 0.02$ is small) and the RFF approximation is accurate (i.e., $M = 100$ is large), increasing the number of observations from the other agents ($t_n = 100$ vs. $t_n = 40$) improves the performance. Lastly, Fig. D.1c verifies FTS's theoretically guaranteed robustness against heterogeneous agents (Section 5.4), since it shows that even when all other agents are heterogeneous (i.e., every $d_n = 1.2$ is large), the performances of FTS are still comparable to that of standard TS. Note that Fig. D.1c demonstrates a potential limitation of our method, i.e., in this scenario of heterogeneous agents, FTS may converge slightly slower than TS if p_t does not grow sufficiently fast. However, the figure also shows that making p_t grow faster (red curve), i.e., making the impact of the other agents decay faster, allows FTS to match the performance of TS in this adverse scenario.

5.5.2 Real-world Experiments

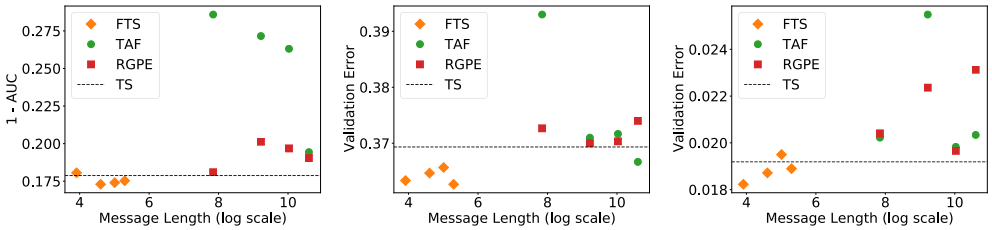
For real-world experiments, we use 3 datasets generated in the federated setting which naturally contain heterogeneous agents (Smith et al., 2017). Firstly, we use a landmine detection dataset in which the landmine fields are located at two different terrains (Xue et al., 2007). Next, we use two activity recognition datasets collected using Google glasses (Rahman et al., 2015) and mobile phone sensors (Anguita et al., 2013), both of which contain heterogeneous agents

since cross-subject heterogeneity has been a major challenge for human activity recognition (Rahman et al., 2015). We compare our FTS algorithm with standard TS (i.e., no communication with other agents), RGPE and TAF. Note that RGPE and TAF are meta-learning algorithms for BO, and are hence not designed for the FBO setting (Section 5.3.2).

Landmine Detection. This dataset includes 29 landmine fields. For each field, every entry in the dataset consists of 9 features and a binary label indicating whether the corresponding location contains landmines. The task of every field is to tune 2 hyperparameters of an SVM classifier (RBF kernel parameter in $[0.01, 10]$ and L2 regularization parameter in $[10^{-4}, 10]$) that is used to predict whether a location contains landmines. We fix one of the landmine fields as the target agent and the remaining $N = 28$ fields as the other agents, each of which has completed a BO task of $t_n = 50$ iterations.

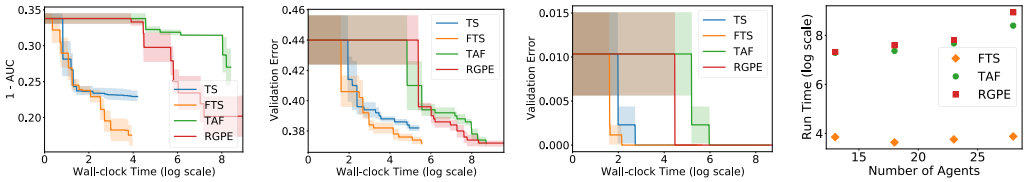
Activity Recognition Using Google Glasses. This dataset contains sensor measurements from Google glasses worn by 38 participants. Every agent attempts to use 57 features, which we extracted from the corresponding participant's measurements, to predict whether the participant is eating or performing other activities. Every agent uses logistic regression (LR) for activity prediction, and needs to tune 3 hyperparameters of LR: batch size ($[20, 60]$), L2 regularization parameter ($[10^{-6}, 1]$) and learning rate ($[0.01, 0.1]$). We fix one of the participants as the target agent and all other $N = 37$ participants as the other agents, each possessing $t_n = 50$ BO observations.

Activity Recognition Using Mobile Phone Sensors. This dataset consists of mobile phone sensor measurements from 30 subjects performing 6 activities. Each agent attempts to tune the hyperparameters of a subject's activity prediction model, whose input includes 561 features and output is one of the 6 activity classes. The activity prediction model and tuned hyperparameters, as well as their ranges, are the same as the Google glasses experiment. We again fix one of



(a) (b) (c)

Figure 5.2: Best performance after 50 iterations (vertical) vs. the length of the message (i.e., the number of parameters) communicated from each agent (horizontal) for the (a) landmine detection, (b) Google glasses and (c) mobile phone sensors experiments. The more to the *bottom left*, the better the performance and the less the required communication. The results for every method correspond to $M = 50, 100, 150, 200$ respectively. Every result is averaged over 6 different target agents, and each target agent is averaged over 5 different initializations of 3 randomly selected inputs.



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

Figure 5.3: Best performance observed vs. run time (seconds) for the (a) landmine detection, (b) Google glasses and (c) mobile phone sensors experiments, in which *FTS converges faster than other methods*. These results correspond to the first (of the 6) target agent used in each experiment in Fig. 5.2 with $M = 100$, averaged over 5 random initializations of 3 inputs.[8] Every method is run for 50 iterations. (d) Total runtime vs. the number of agents for the landmine detection experiment.

the subjects as the target agent, and all other $N = 29$ subjects as the other agents with $t_n = 50$ observations each.

For all experiments, we set P_N to be uniform: $P_N[n] = 1/N, \forall n = 1, \ldots, N$, and $p_t = 1 - 1/t^2$ for $t \geq 2$ and $p_1 = p_2$. We use validation error as the performance metric for the two activity recognition experiments, and use Area Under the Receiver Operating Characteristic Curve (AUC) to measure the performance of the landmine detection experiment since this dataset is extremely imbalanced (i.e., only 6.2% of all locations contain landmines). We repeat every experiment 6 times, every time treating one of the first 6 agents as the target agent.

Fig. 5.2 shows the (averaged) best performance after 50 iterations of different methods (vertical axis), as well as their required number of parameters to be passed from each agent (horizontal axis). FTS outperforms both RGPE and TAF in terms of both the *performance metric* and the *communication efficiency*. Note that this comparison is unfair for FTS, since FTS is much more computationally efficient than RGPE and TAF (Section 5.3.2) such that it completes 50 iterations in significantly shorter time (Fig. 5.3). Fig. 5.3 plots the best performance achieved vs. the run time of different algorithms with the first agent treated as the target agent (refer to Appendix B.4.2.3 for the results of the other agents[8]). Fig. 5.3 shows that FTS achieves the fastest convergence among all methods, and showcases the advantage of FTS over RGPE and TAF in terms of *computational efficiency* (Section 5.3.2). Overall, the consistent performance advantage of FTS across all real-world experiments is an indication of its practical robustness, which might be largely attributed to its robust theoretical convergence guarantee ensuring its consistent performance even in the presence of heterogeneous agents (Section 5.4). Furthermore, we also use the landmine detection experiment to illustrate the scalability of our method w.r.t. the number of agents N. The results (Fig. 5.3d) show that increasing N has minimal impact on the runtime of FTS yet leads to growing computational cost for RGPE and TAF. This verifies the relevant discussion at the end of Section 5.3.2.

5.6 Conclusion

In this work, we have introduced the first algorithm for the FBO setting, FTS, which addresses some key challenges in FBO in a principled manner. We theoretically show its convergence guarantee which is robust against heterogeneous agents, and empirically demonstrate its communication efficiency, computational efficiency

[8]We can not average the results across different agents since the output scales of different agents vary significantly.

and practical effectiveness using three real-world experiments.

Chapter 6

Differentially Private Federated Bayesian Optimization

6.1 Introduction

In the previous chapter (Chapter 5), we have extended BO into the *federated learning* (FL) setting, to derive the *federated Thompson sampling* (FTS) algorithm for the *federated BO* (FBO) setting (Dai et al., 2020b). An important consideration in FL has been a rigorous protection of the privacy of the users/agents, i.e., how to guarantee that by participating in a FL system, an agent would not reveal sensitive information about itself (Kairouz et al., 2019). Furthermore, incorporating a rigorous privacy preservation into BO has recently attracted increasing attention due to its importance to real-world BO applications (Kharkovskii et al., 2020; Kusner et al., 2015; Nguyen et al., 2018; Zhou and Tan, 2020). However, our FTS algorithm (Chapter 5), which is the state-of-the-art algorithm for the FBO setting (Dai et al., 2020b), is not equipped with a privacy guarantee and hence lacks a rigorous protection of the sensitive information of the agents.

Differential privacy (DP) (Dwork et al., 2014) provides a rigorous privacy guarantee for data release and has become the state-of-the-art method for designing

privacy-preserving ML algorithms (Ji et al., 2014). Recently, DP has been applied to the iterative training of DNN using *stochastic gradient descent* (DP-SGD) (Abadi et al., 2016) and the FL algorithm of *federated averaging* (DP-FedAvg) (McMahan et al., 2018b), which have achieved competitive performances (utility) with a strong privacy guarantee. Notably, these methods have followed a general framework for adding DP to generic iterative algorithms (McMahan et al., 2018a) (referred to as *the general DP framework* hereafter), which applies a *subsampled Gaussian mechanism* (Section 6.2) in every iteration. For an iterative algorithm (e.g., FedAvg) applied to a database with multiple *records* (e.g., data from multiple users), the general DP framework (McMahan et al., 2018a) can hide the participation of any single record in the algorithm in a principled way. For example, DP-FedAvg (McMahan et al., 2018b) guarantees that an adversary, even with arbitrary side information, cannot infer whether the data from a particular user has been used by the algorithm, hence preserving *user-level privacy*. Unfortunately, our FTS algorithm (Chapter 5) is not amenable to a straightforward integration of the general DP framework (McMahan et al., 2018a) (Section 6.3.1). So, we modify FTS to be compatible with the general DP framework and hence introduce the DP-FTS algorithm to preserve user-level privacy in the FBO setting. In addition to the challenge of accounting for our modifications of FTS to integrate DP in our theoretical analysis, we have to ensure that DP-FTS preserves the practical performance advantage (utility) of FTS. To this end, we leverage the ability of the general DP framework to handle different parameter vectors (McMahan et al., 2018a), as well as the method of local modeling for BO, to further improve the practical performance (utility) of DP-FTS.

Note that FTS, as well as DP-FTS, is able to achieve better performance (utility) than standard TS by *accelerating exploration* using the information from the other agents (aggregated by the central server) (Chapter 5). That is, an agent using

FTS/DP-FTS benefits from needing to perform less exploration *in the early stages*. To improve the utility of DP-FTS even more, we further accelerate exploration in the early stages using our proposed *distributed exploration* technique which is an elegant combination of local modeling for BO and the ability of the general DP framework to handle different parameter vectors. Specifically, we divide the entire search space into smaller local *sub-regions* and let every agent explore only one local sub-region *at initialization*. As a result, compared with the entire search space, every agent can explore the local sub-region more effectively because its GP can model the objective function more accurately in a smaller local sub-region (Eriksson et al., 2019). Subsequently, in every BO iteration, the central server aggregates the information (vector) for every sub-region separately: For a sub-region, the aggregation (i.e., weighted average) gives more emphasis (i.e., weights) to the information (vectors) from those agents who are assigned to explore this particular sub-region. Interestingly, this technique can be seamlessly integrated into the general DP framework due to its ability to process different parameter vectors (i.e., one vector for every sub-region) while still preserving the interpretation as a single subsampled Gaussian mechanism (McMahan et al., 2018a) (Section 6.3.3).[1] As a result, the information aggregated by the central server can help the agents explore every sub-region (hence the entire search space) more effectively in the early stages and thus significantly improve the practical convergence (utility), as demonstrated in our experiments (Section 6.5). We refer to the resulting DP-FTS algorithm with *distributed exploration* (DE) as DP-FTS-DE. Note that DP-FTS is a special case of DP-FTS-DE with only one sub-region (i.e., entire search space). So, we will refer to DP-FTS-DE as our main algorithm in the rest of this chapter.

In this chapter, we introduce the *differentially private FTS with DE* (DP-FTS-DE) algorithm (Section 6.3) which is the first algorithm with a rigorous

[1]By analogy, the vectors for different sub-regions in our algorithm play a similar role to the parameters of different layers of a DNN in DP-FedAvg (McMahan et al., 2018b).

guarantee on the user-level privacy in the FBO setting. In particular, DP-FTS-DE guarantees that an adversary cannot infer whether an agent has participated in the algorithm, hence assuring every agent that its participation will not reveal its sensitive information.[2] We provide theoretical guarantees for both the privacy and utility of DP-FTS-DE, which combine to yield a number of elegant theoretical insights about the *privacy-utility trade-off* (Section 6.4). Next, we empirically demonstrate that DP-FTS-DE delivers an effective performance with a strong privacy guarantee and induces an interesting trade-off between privacy and utility in real-world applications (Section 6.5).

6.2 Background and Problem Formulation

6.2.1 Federated Bayesian Optimization

The setting of FBO we focus on in this chapter is the same as the one in Chapter 5 (Section 5.2.2), and we make some changes to the notations here for convenience. Specifically, FBO involves N agents $\mathcal{A}_1, \ldots, \mathcal{A}_N$. Every agent $\mathcal{A}_n$ attempts to maximize its objective function $f^n : \mathcal{X} \to \mathbb{R}$, i.e., to find $\mathbf{x}^{n,*} \in \arg\max_{\mathbf{x}\in\mathcal{X}} f^n(\mathbf{x})$, by querying input $\mathbf{x}_t^n$ and observing noisy output $y_t^n, \forall t = 1, \ldots, T$.[3] Without loss of generality, our theoretical analyses mainly focus on the perspective of agent $\mathcal{A}_1$. That is, we derive an upper bound on the cumulative regret of $\mathcal{A}_1$: $R_T^1 \triangleq \sum_{t=1}^{T}(f^1(\mathbf{x}^{1,*}) - f^1(\mathbf{x}_t^1))$ in Section 6.4. We characterize the similarity between $\mathcal{A}_1$ and $\mathcal{A}_n$ by $d_n \triangleq \max_{\mathbf{x}\in\mathcal{X}} |f^1(\mathbf{x}) - f^n(\mathbf{x})|$ such that $d_1 = 0$ and a smaller d_n indicates a larger degree of similarity between $\mathcal{A}_1$ and $\mathcal{A}_n$. Following Chapter 5, we assume that all participating agents share the same set of random features $\boldsymbol{\phi}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$. In our theoretical analysis,

[2]Following that of McMahan et al. (2018b), we assume the central server is trustworthy.

[3]Note that the notations in Chapter 5 are slightly different: In Chapter 5, there are in total $N + 1$ agents consisting of a target agent and N other agents, whose objective functions are denoted by f, and $g_n, \forall n = 1, \ldots, N$, respectively.

we assume that all objective functions have a bounded norm induced by the *reproducing kernel Hilbert space* (RKHS) associated with the kernel k, i.e., $\|f^n\|_k \leq B, \forall n = 1, \ldots, N$.

6.2.2 Differential Privacy

Differential Privacy (DP) provides a rigorous framework for privacy-preserving data release (Dwork et al., 2006b). Consistent with that of McMahan et al. (2018b), we define two datasets as *adjacent* if they differ by the data of a single user/agent, which leads to the definition of user-level DP:

Definition 6.1 (User-level DP)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ satisfies (ϵ, δ)-DP if for any two adjacent datasets D_1 and D_2 and any subset of outputs $\mathcal{S} \subset \mathcal{R}$,*

$$\mathbb{P}(\mathcal{M}(D_1) \in \mathcal{S}) \leq e^{\epsilon}\, \mathbb{P}(\mathcal{M}(D_2) \in \mathcal{S}) + \delta\ .$$

Here, ϵ and δ are DP parameters such that the smaller they are, the better the privacy guarantee. Intuitively, user-level DP ensures that adding or removing any single user/agent from the algorithm has an imperceptible impact on the output of the algorithm. The DP-FedAvg algorithm (McMahan et al., 2018b) has incorporated user-level DP into the FL setting by adopting a general DP framework (McMahan et al., 2018a). In DP-FedAvg, the central server applies a *subsampled Gaussian mechanism* to the vectors (gradients) received from multiple agents in every iteration t:

1. Select a subset of agents by choosing every agent with a fixed probability q,
2. Clip the vector $\boldsymbol{\omega}_{n,t}$ from every selected agent n such that its L_2 norm is upper-bounded by S, and
3. Add a Gaussian noise (with a variance proportional to S^2) to the weighted average of the clipped vectors.

The central server then broadcasts the vector produced by step 3 to all agents. As a result of the general DP framework, they are able to not only provide a rigorous privacy guarantee, but also use the *moments accountant* method (Abadi et al., 2016) to calculate the *privacy loss*.[4] More recently, the work of McMahan et al. (2018a) has formalized the methods used by Abadi et al. (2016) and McMahan et al. (2018b) as a general DP framework that is applicable to generic iterative algorithms. Notably, the general DP framework (McMahan et al., 2018a) can naturally process different parameter vectors (e.g., parameters of different layers of a DNN), which is an important property that allows us to integrate distributed exploration (Section 6.3.2) into our algorithm.

6.3 Differentially Private Federated Thompson Sampling with Distributed Exploration

In this section, we will first introduce how we modify the FTS algorithm to integrate DP to derive the DP-FTS algorithm (Section 6.3.1). Then, in Section 6.3.2, we will describe distributed exploration which can be seamlessly integrated into DP-FTS to further improve utility. Lastly, we will present the complete DP-FTS-DE algorithm (Section 6.3.3).

6.3.1 Differentially Private Federated Thompson Sampling (DP-FTS)

The original FTS algorithm (Dai et al., 2020b) (Algorithm 5.1 in Chapter 5) is not amenable to a straightforward integration of the general DP framework (Section 6.2.2). So, we modify FTS by (a) adding a central server for performing the privacy-preserving transformations, and (b) passing a single aggregated vector

[4]Given a fixed δ, the privacy loss is defined as an upper bound on the value of ϵ calculated by the moments accountant method.

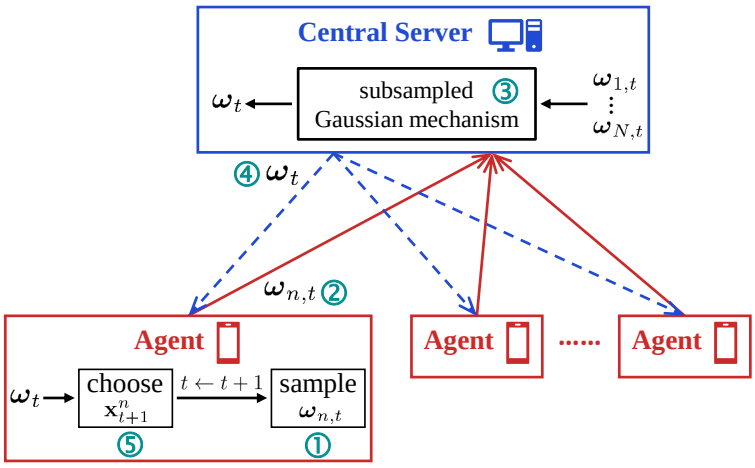


Figure 6.1: DP-FTS algorithm (without distributed exploration).

(instead of one vector from every agent) to the agents. Fig. 6.1 illustrates our DP-FTS algorithm which is obtained by integrating the general DP framework into modified FTS. Every iteration t of DP-FTS consists of the following steps:

① and ② (by Agents): Every agent $\mathcal{A}_n$ samples a vector $\boldsymbol{\omega}_{n,t}$ (5.1) using its own current history of t input-output pairs (step ①) and sends $\boldsymbol{\omega}_{n,t}$ to the central server (step ②).

③ and ④ (by Central Server): Next, the central server processes the received vectors $\boldsymbol{\omega}_{n,t}, \forall n = 1, \ldots, N$ using a subsampled Gaussian mechanism (step ③): It (a) selects a subset of agents $\mathcal{S}_t$ by choosing each agent with probability q, (b) clips the vector of every selected agent $\boldsymbol{\omega}_{n,t}, \forall n \in \mathcal{S}_t$ such that its L_2 norm is upper-bounded by S, and (c) calculates a *weighted average* of the clipped vectors using weights $\{\varphi_n, \forall n = 1, \ldots, N\}$, and adds to it a Gaussian noise. The final vector $\boldsymbol{\omega}_t$ is then broadcast to all agents (step ④).

⑤ (by Agents): After an agent $\mathcal{A}_n$ receives the vector $\boldsymbol{\omega}_t$ from the central server, it can choose the next query $\mathbf{x}^n_{t+1}$ (step ⑤): With probability $p_{t+1} \in (0, 1]$, $\mathcal{A}_n$ chooses $\mathbf{x}^n_{t+1}$ using standard TS by firstly sampling a function f^n_{t+1} from its own GP posterior and then choosing $\mathbf{x}^n_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} f^n_{t+1}(\mathbf{x})$; with probability $1 - p_{t+1}$, $\mathcal{A}_n$ chooses $\mathbf{x}^n_{t+1}$ using $\boldsymbol{\omega}_t$ received from the central server: $\mathbf{x}^n_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t$. Consistent with Chapter 5 (Section 5.3.1), $[p_t]_{t \in \mathbb{Z}^+}$

is chosen as a monotonically increasing sequence such that $p_t \in (0, 1], \forall t$ and $p_t \to 1$ as $t \to \infty$.

After choosing $\mathbf{x}_{t+1}^n$ and observing y_{t+1}^n, $\mathcal{A}_n$ adds $(\mathbf{x}_{t+1}^n, y_{t+1}^n)$ to its history and samples a new vector $\boldsymbol{\omega}_{n,t+1}$ (step ①). Next, $\mathcal{A}_n$ sends $\boldsymbol{\omega}_{n,t+1}$ to the central server (step ②), and the algorithm is repeated. The detailed algorithm will be presented in Section 6.3.3 since DP-FTS is equivalent to the DP-FTS-DE algorithm with $P = 1$ sub-region.

6.3.2 Distributed Exploration (DE)

To accelerate the practical convergence (utility) of DP-FTS (Section 6.3.1), we introduce the DE technique to further accelerate the exploration in the early stages (Section 6.1). At the beginning of a BO algorithm, a small number of initial points are usually selected from the entire domain using an exploration method (e.g., random search) to warm-start BO. We use DE to allow every agent to explore a smaller local sub-region *at initialization*, which is easier to model for the GP surrogate (Eriksson et al., 2019), and leverage the ability of the general DP framework to handle different parameter vectors (McMahan et al., 2018a) to integrate DE into DP-FTS in a seamless way.

Specifically, we partition the input domain $\mathcal{X}$ into $P \geq 1$ disjoint sub-regions: $\mathcal{X}_1, \ldots, \mathcal{X}_P$ such that $\cup_{i=1,\ldots,P} \mathcal{X}_i = \mathcal{X}$ and $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset, \forall i \neq j$. At initialization, we assign every agent $\mathcal{A}_n$ to explore (i.e., choose the initial points randomly from) a particular sub-region $\mathcal{X}_{i_n}$.[5] Note that if an agent $\mathcal{A}_n$ is assigned to explore a sub-region $\mathcal{X}_{i_n}$ (instead of exploring the entire domain), its vector $\boldsymbol{\omega}_{n,t}$ (5.1) sent to the central server is more informative about its objective function *in this sub-region* $\mathcal{X}_{i_n}$.[6] As a result, for a sub-region $\mathcal{X}_i$, the vectors from those

[5]For simplicity, we choose the sub-regions to be hyper-rectangles with equal volumes and assign an approximately equal number of agents ($\approx N/P$) to explore every sub-region.

[6]This is because its GP surrogate can model the objective function in this local sub-region more accurately (Eriksson et al., 2019).

agents exploring $\mathcal{X}_i$ contain information that can help better explore $\mathcal{X}_i$. So, we let the central server construct a separate vector $\boldsymbol{\omega}_t^{(i)}$ for every sub-region $\mathcal{X}_i$ and when constructing $\boldsymbol{\omega}_t^{(i)}$, give more weights to the vectors from those agents exploring $\mathcal{X}_i$ because they are more informative about $\mathcal{X}_i$. Consequently, the central server needs to construct P different vectors $\{\boldsymbol{\omega}_t^{(i)}, \forall i = 1, \ldots, P\}$ with each $\boldsymbol{\omega}_t^{(i)}$ using a separate set of weights $\{\varphi_n^{(i)}, \forall n = 1, \ldots, N\}$. Interestingly, from the perspective of the general DP framework (McMahan et al., 2018a), the different vectors $\{\boldsymbol{\omega}_t^{(i)}, \forall i = 1, \ldots, P\}$ can be interpreted as analogous to the parameters of different layers of a DNN and can thus be naturally handled by the general DP framework. After receiving the P vectors from the central server, as illustrated in Fig. 6.2b, every agent uses $\boldsymbol{\omega}_t^{(i)}$ to reconstruct the sampled function in the sub-region $\mathcal{X}_i$: $\widetilde{f}_t(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i)}, \forall \mathbf{x} \in \mathcal{X}_i$, and then (with probability $1 - p_t$) chooses the next query by maximizing the sampled functions from all sub-regions; see more details in Section 6.3.3.

After initialization, every agent is allowed to query any input in the entire domain $\mathcal{X}$ regardless of the sub-region it is assigned to. So, as t becomes larger, every agent is likely to have explored (and become informative about) more sub-regions in addition to the one it is assigned to. In this regard, for every sub-region $\mathcal{X}_i$, we make the set of weights $\{\varphi_n^{(i)}, \forall n = 1, \ldots, N\}$ *adaptive* such that they gradually become uniform among all agents as t becomes large. The dependence of the weights on t only requires minimal modifications to the algorithm and the theoretical results. So, we drop this dependence to simplify notations.

6.3.3 DP-FTS-DE Algorithm

Our complete DP-FTS-DE algorithm after integrating DE (Section 6.3.2) into DP-FTS (Section 6.3.1) is presented in Algorithm 6.1 (central server's role)

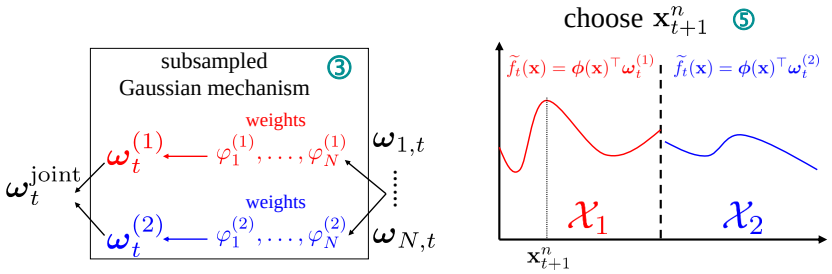
[7] $i^{[\mathbf{x}]}$ represents the sub-region $\mathbf{x}$ is assigned to.

Algorithm 6.1 DP-FTS-DE (central server)

$\boldsymbol{\omega}_{-1}^{\text{joint}} = \mathbf{0}$
for iterations $t = 0, 1, 2, \ldots, T$ **do**
  for agents $n = 1, 2, \ldots, N$ **in parallel do**
    $\boldsymbol{\omega}_{n,t} \leftarrow$ **BO-Agent-**$\boldsymbol{\mathcal{A}_n}(t, \boldsymbol{\omega}_{t-1}^{\text{joint}})$ ②
  $\boldsymbol{\omega}_t^{(i)} = \mathbf{0}, \forall i = 1, \ldots, P$
  Choose a random subset $\mathcal{S}_t \subset \{1, \ldots, N\}$ of agents
  for sub-regions $i = 1, 2, \ldots, P$ **do**
    for agents $n \in \mathcal{S}_t$ **do**
      $\widehat{\boldsymbol{\omega}}_{n,t} = \boldsymbol{\omega}_{n,t} \Big/ \max\left(1, \frac{\|\boldsymbol{\omega}_{n,t}\|_2}{S/\sqrt{P}}\right)$ ③
      $\boldsymbol{\omega}_t^{(i)} \mathrel{+}= (\varphi_n^{(i)}/q)\, \widehat{\boldsymbol{\omega}}_{n,t}$
    $\boldsymbol{\omega}_t^{(i)} \mathrel{+}= \mathcal{N}\left(\mathbf{0}, (z\varphi_{\max} S/q)^2 \mathbf{I}\right)$
  Broadcast $\boldsymbol{\omega}_t^{\text{joint}} = [\boldsymbol{\omega}_t^{(i)}]_{i=1,\ldots,P}$ to all agents ④

Algorithm 6.2 BO-Agent-$\boldsymbol{\mathcal{A}_n}(t, \boldsymbol{\omega}_{t-1}^{\text{joint}} = [\boldsymbol{\omega}_{t-1}^{(i)}]_{i=1,\ldots,P})$

if $t = 0$ **then**
  Randomly select and query N_{init} initial points from sub-region $\mathcal{X}_{i_n}$
else
  With probability p_t: ⑤
    $\mathbf{x}_t^n = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t^n(\mathbf{x})$,
  With probability $1 - p_t$:
    $\mathbf{x}_t^n = \arg\max_{\mathbf{x} \in \mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{t-1}^{(i^{[\mathbf{x}]})}$.[7]
Query $\mathbf{x}_t^n$ to observe y_t^n
Sample $\boldsymbol{\omega}_{n,t}$ and send it to central server ①, ②



(a) (b)

Figure 6.2: Replacing steps ③ and ⑤ of DP-FTS (Fig. 6.1) with that in (a) and (b) to derive the DP-FTS-DE algorithm ($P = 2$).

and Algorithm 6.2 (agent's role), with the steps in circle corresponding to those in Fig. 6.1. DP-FTS-DE differs from DP-FTS in two major aspects: Firstly, at initialization ($t = 0$), every agent only explores a local sub-region instead of the entire domain (line 2 of Algorithm 6.2). Secondly, instead of a single vector $\boldsymbol{\omega}_t$, the central server produces and broadcasts P vectors $\{\boldsymbol{\omega}_t^{(i)}, \forall i = 1, \ldots, P\}$ with each vector $\boldsymbol{\omega}_t^{(i)}$ corresponding to a different sub-region $\mathcal{X}_i$ and using a different set of weights $\{\varphi_n^{(i)}, \forall n = 1, \ldots, N\}$. Applying different transformations to different vectors (e.g., parameters of different DNN layers) can be naturally incorporated into the general DP framework (McMahan et al., 2018a). Different transformations performed by our central server to produce P vectors can be interpreted as *a single subsampled Gaussian mechanism* producing a single joint vector $\boldsymbol{\omega}_t^{\text{joint}} \triangleq [\boldsymbol{\omega}_t^{(i)}]_{i=1,\ldots,P}$, as illustrated in Fig. 6.2a. We will present the transformations performed by the central server in DP-FTS-DE (lines 5-12 of Algorithm 6.1) from this perspective.

Subsampling: To begin with, after receiving the vectors $\boldsymbol{\omega}_{n,t}$'s from the agents (lines 3-4 of Algorithm 6.1), the central server firstly chooses a random subset of agents $\mathcal{S}_t$ by selecting each agent with probability q (line 6). Next, for every selected agent $n \in \mathcal{S}_t$, the central server constructs a $P \times M$-dimensional joint vector: $\boldsymbol{\omega}_{n,t}^{\text{joint}} \triangleq [N\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}]_{i=1,\ldots,P}$.

Clipping: Subsequently, clip every selected vector $\boldsymbol{\omega}_{n,t}$ to obtain $\widehat{\boldsymbol{\omega}}_{n,t}$ whose L_2 norm is upper-bounded by $S/\sqrt{P}$ (line 9). This is equivalent to

clipping the joint vector to obtain: $\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}} \triangleq [N\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}]_{i=1,\dots,P}$, whose L_2 norm is bounded by $\left\|\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}\right\|_2 \leq (N^2\varphi_{\max}^2 \sum_{i=1}^{P} \left\|\widehat{\boldsymbol{\omega}}_{n,t}\right\|_2^2)^{1/2} \leq N\varphi_{\max} S$ where $\varphi_{\max} \triangleq \max_{i=1,\dots,P, n=1,\dots,N} \varphi_n^{(i)}$.

Weighted Average: Next, calculate a weighted average of the clipped joint vectors by giving equal weights[8] to all agents: $\boldsymbol{\omega}_t^{\text{joint}} = (qN)^{-1} \sum_{n\in\mathcal{S}_t} \widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}$.[9] Note that $\boldsymbol{\omega}_t^{\text{joint}}$ results from the concatenation of the vectors from all sub-regions: $\boldsymbol{\omega}_t^{\text{joint}} = [\boldsymbol{\omega}_t^{(i)}]_{i=1,\dots,P}$ where

$$\boldsymbol{\omega}_t^{(i)} = (qN)^{-1} \sum_{n\in\mathcal{S}_t} N\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t} = q^{-1} \sum_{n\in\mathcal{S}_t} \varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}, \tag{6.1}$$

which corresponds to line 10 of Algorithm 6.1.

Gaussian Noise: Finally, add to each element of $\boldsymbol{\omega}_t^{\text{joint}} = [\boldsymbol{\omega}_t^{(i)}]_{i=1,\dots,P}$ a zero-mean Gaussian noise with a standard deviation of $z(N\varphi_{\max}S)/(qN) = z\varphi_{\max}S/q$ (line 11).

Next, the output $\boldsymbol{\omega}_t^{\text{joint}} = [\boldsymbol{\omega}_t^{(i)}]_{i=1,\dots,P}$ of the single subsampled Gaussian mechanism is broadcast to all agents. After an agent $\mathcal{A}_n$ receives $\boldsymbol{\omega}_t^{\text{joint}}$, with probability p_{t+1}, $\mathcal{A}_n$ chooses the next query $\mathbf{x}_{t+1}^n$ by maximizing a function f_{t+1}^n sampled from its own GP posterior (line 5 of Algorithm 6.2); with probability $1-p_{t+1}$, $\mathcal{A}_n$ uses $\boldsymbol{\omega}_t^{\text{joint}} = [\boldsymbol{\omega}_t^{(i)}]_{i=1,\dots,P}$ received from the central server to choose $\mathbf{x}_{t+1}^n$ by maximizing the reconstructed functions for all sub-regions (line 7 of Algorithm 6.2), as illustrated in Fig. 6.2b. Finally, it queries $\mathbf{x}_{t+1}^n$ to observe y_{t+1}^n, samples a new vector $\boldsymbol{\omega}_{n,t+1}$ and sends it to the central server (line 9 of Algorithm 6.2), and the algorithm is repeated.

[8]Note that this weight is only used for the purpose of interpretation and is different from the weights $\varphi_n^{(i)}$'s used in our algorithm.

[9]The summation is divided by qN (i.e., expected number of agents selected) to make it unbiased (McMahan et al., 2018b).

6.4 Theoretical Analysis

In this section, we will present the theoretical guarantees on both the privacy (Section 6.4.1) and utility (Section 6.4.2) of our DP-FTS-DE algorithm, which combine to yield some interesting insights about the *privacy-utility trade-off* (Section 6.4.2).

6.4.1 Privacy Guarantee

Proposition 6.1 below formalizes our privacy guarantee:

Proposition 6.1. *There exist constants* c_1 *and* c_2 *such that for fixed* q *and* T *and any* $\epsilon < c_1 q^2 T, \delta > 0$, *DP-FTS-DE (Algorithm 6.1) is* (ϵ, δ)*-DP if* $z \geq c_2 q \sqrt{T \log(1/\delta)}/\epsilon$.

Its proof (Appendix C.1.1) follows directly from Theorem 1 of Abadi et al. (2016). Proposition 6.1 shows that a larger z (i.e., larger variance for Gaussian noise), a smaller q (i.e., smaller expected number of selected agents) and a smaller T (i.e., smaller number of iterations) all improve the privacy guarantee because for a fixed δ, they all allow ϵ to be smaller.

6.4.2 Privacy-Utility Trade-off

Theorem 6.1 below (proved in Appendix C.1.2) shows an upper bound on the cumulative regret of agent $\mathcal{A}_1$ running DP-FTS-DE:

Theorem 6.1. *Let* γ_t *be the maximum information gain on* f^1 *from any set of* t *observations,* ε *denote an upper bound on the approximation error of RFF (Section 5.2),* $\mathcal{C}_t \triangleq \{n = 1, \ldots, N \big| \left\|\boldsymbol{\omega}_{n,t}\right\|_2 > S/\sqrt{P}\}, \forall t \in \mathbb{Z}^+, \delta \in (0, 1)$. *Choose* $[p_t]_{t \in \mathbb{Z}^+}$ *as a monotonically increasing sequence satisfying* $1 - p_t =$

$\mathcal{O}(1/t^2)$. *Then, with probability of at least* $1-\delta$, [10]

$$R_T^1 = \tilde{\mathcal{O}}\left(\left(B+1/p_1\right)\gamma_T\sqrt{T} + \textstyle\sum_{t=1}^{T}\psi_t + B\sum_{t=1}^{T}\vartheta_t\right)$$

where $\psi_t \triangleq \tilde{\mathcal{O}}((1-p_t)P\varphi_{\max}q^{-1}(\Delta_t + zS\sqrt{M}))$, $\Delta_t \triangleq \sum_{n=1}^{N}\Delta_{n,t}$, $\Delta_{n,t} \triangleq \tilde{\mathcal{O}}(\varepsilon Bt^2 + B + \sqrt{M} + d_n + \sqrt{\gamma_t})$, *and* $\vartheta_t \triangleq (1-p_t)\sum_{i=1}^{P}\sum_{n\in\mathcal{C}_t}\varphi_n^{(i)}$.

Note that all three terms in the regret upper bound grow sub-linearly in t: The first term is sub-linear because $\gamma_T = \mathcal{O}((\log T)^{d+1})$ for the SE kernel, and the second and third terms are both sub-linear since we have chosen $1-p_t = \mathcal{O}(1/t^2)$. As a result, our DP-FTS-DE algorithm preserves the no-regret property of the original FTS algorithm (Dai et al., 2020b) (Chapter 5). That is, agent $\mathcal{A}_1$ achieves *no regret asymptotically* even if all other agents are heterogeneous, i.e., even if all other agents have significantly different objective functions from $\mathcal{A}_1$. This is a particularly desirable property because it ensures the robustness of DP-FTS-DE against the heterogeneity of agents, which is an important challenge in both FL and FBO (Dai et al., 2020b; Kairouz et al., 2019) and has also been an important consideration for other works proving the theoretical convergence of FL algorithms (Li et al., 2018, 2020d). Similar to Chapter 5, to prove this robust regret upper bound, we have upper-bounded the *worst-case error* introduced by the information from *any* set of agents, which explains the dependence of the regret upper bound on d_n and $[p_t]_{t\in\mathbb{Z}^+}$. Specifically, larger d_n's indicate larger differences between the objective functions of $\mathcal{A}_1$ and the other agents and hence lead to a worse regret upper bound (through the term ψ_t). Smaller values of the sequence $[p_t]_{t\in\mathbb{Z}^+}$ increase the utilization of the information from the other agents (line 6 of Algorith 6.2), hence inflating the worst-case error resulting from these information.

Theorem 6.1, when interpreted together with Proposition 6.1, also reveals

[10] $\tilde{\mathcal{O}}$ hides all logarithmic factors.

some interesting theoretical insights regarding the privacy-utility trade-off. Firstly, a larger z (i.e., larger variance for Gaussian noise) improves the privacy guarantee (Proposition 6.1) and yet results in a worse utility since it leads to a worse regret upper bound (through ψ_t). Secondly, a larger q (i.e., more selected agents in each iteration) improves the utility since it tightens the regret upper bound (by reducing the value of ψ_t) at the expense of a worse privacy guarantee (Proposition 6.1). A general guideline for choosing the values of z and q is to aim for a good utility while ensuring that the privacy loss is within the single-digit range (i.e., < 10) (Abadi et al., 2016). The value of the clipping threshold S exerts no impact on the privacy guarantee. However, S affects the regret upper bound (hence the utility) through two conflicting effects: Firstly, a smaller S reduces the value of ψ_t and hence the regret bound. However, a smaller S is likely to enlarge the cardinality of the set $\mathcal{C}_t$, hence increasing the value of ϑ_t. Intuitively, a smaller S impacts the performance positively by reducing the noise variance (line 11 of Algorithm 6.1) and yet negatively by causing more vectors to be clipped (line 9 of Algorithm 6.1). A general guide on the selection of S is to choose a small value while ensuring that a small number of vectors are clipped.

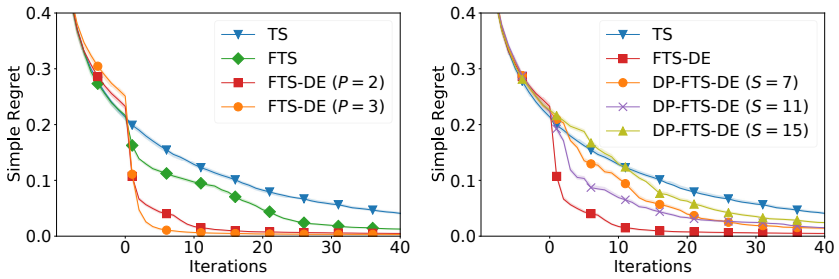
Regarding the dependence of the regret upper bound on the number M of random features, in addition to the dependence through $\Delta_{n,t}$ which has been analyzed in Chapter 5 (Section 5.4) (Dai et al., 2020b), the integration of DP introduces another dependence that implicitly affects the privacy-utility trade-off. Besides increasing the value of ψ_t, a larger M enlarges the value of ϑ_t as a larger dimension for the vectors $\boldsymbol{\omega}_{n,t}$'s is expected to increase their L_2 norms, hence increasing the cardinality of the set $\mathcal{C}_t$ and consequently the value of ϑ_t. So, the additional dependence, which arises due to the integration of DP, loosens the regret upper bound with an increasing M. As a result, if M is larger, then we can either *sacrifice privacy to preserve utility* by reducing z or increasing q (both of which can counter the increase of ψ_t), or *sacrifice utility to preserve privacy* by

keeping z and q unchanged.

The number P of sub-regions also induces a trade-off about the performance of our algorithm, which is partially reflected by Theorem 6.1. Specifically, the regret bound depends on P through three terms. Two of the terms (i.e., P in ψ_t and the summation of P terms in ϑ_t) arise due to the worst-case nature of our regret upper bound, as discussed earlier: They cause the regret upper bound to increase with P due to the accumulation of the worst-case errors resulting from the P vectors: $\{\boldsymbol{\omega}_t^{(i)}, \forall i = 1, \ldots, P\}$. Regarding the third term (i.e., in the definition of $\mathcal{C}_t$), a larger value of P is expected to increase the cardinality of the set $\mathcal{C}_t$ (similar to the effect of a larger M discussed above), consequently loosening the regret upper bound by inflating the value of ϑ_t. In this case, as described above, we can choose to sacrifice either privacy or utility. On the other hand, a larger value of P (i.e., larger number of sub-regions) can improve the practical performance (utility) because it allows every agent to explore only a smaller sub-region which can be better modeled by its GP surrogate (Section 6.3.2). As a result of the worst-case nature of the regret upper bound mentioned earlier, the latter positive effect leading to better practical utility is not reflected by Theorem 6.1. Therefore, we instead verify this trade-off induced by P about the practical performance in our experiments (Fig. C.3 in Appendix C.2.1).

6.5 Experiments

In all experiments, when calculating the privacy loss using moments accountant Abadi et al. (2016), we follow the practice of McMahan et al. (2018b) and set $\delta = 1/N^{1.1}$. The requirement on the sequence $[p_t]_{t\in\mathbb{Z}^+}$ by Theorem 6.1 (i.e., $1 - p_t = \mathcal{O}(1/t^2)$) is conservative in practice due to the worst-case nature of the regret upper bound (Section 6.4.2). So, we choose $1 - p_t$ to decay slower in our experiments. For every sub-region $\mathcal{X}_i$, we choose the corresponding set of



(a) (b)

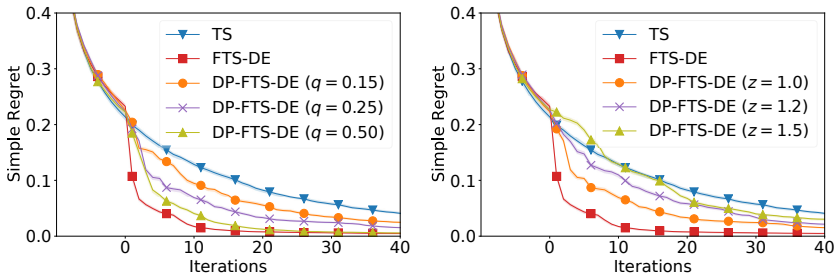
Figure 6.3: (a) Benefit of *distributed exploration* (DE). (b) Impact of clipping threshold S with $q = 0.25, z = 1.0$. Every curve results from averaging over $N = 200$ agents s.t. each agent's performance is further averaged over 5 runs with different random initializations of size $N_{\text{init}} = 10$. Error bars denote standard errors.

weights such that $\varphi_n^{(i)} \propto \exp(a)$ for the agents exploring $\mathcal{X}_i$ and $\varphi_n^{(i)} \propto \exp(b)$ for the other agents with $a \geq b > 0$. To make the weights adaptive (Section 6.3.2), we fix b and decay the value of a. Due to space constraints, some experimental details are deferred to Appendix C.2.

6.5.1 Synthetic Experiments

In our synthetic experiments, we first sample a function from a GP with the SE kernel, and then apply different small random perturbations to the values of the sampled function to obtain the objective functions of $N = 200$ different agents. We choose $M = 50$ and $1 - p_t = 1/\sqrt{t}, \forall t \in \mathbb{Z}^+$.

We firstly demonstrate the performance advantage of modified FTS and FTS-DE (without DP) over standard TS. As shown in Fig. 6.3a, FTS converges faster than standard TS and more importantly, FTS-DE (Section 6.3.2) further improves the performance of FTS considerably. Moreover, using a larger number P of sub-regions (i.e., smaller sub-regions) brings more performance benefit. This is consistent with our analysis of DE (Section 6.3.2) suggesting that smaller sub-regions are easier to model for the GP surrogate and hence can be better explored. Moreover, we have also verified that after the integration of DP, DP-FTS-DE ($P = 2$) can still achieve significantly better convergence (utility)



(a) (b)

Figure 6.4: Privacy-utility trade-off: Privacy loss after $T = 40$ iterations are $5.93, 9.91, 20.12$ for the respective $q = 0.15, 0.25, 0.5$ with (a) $z = 1.0, S = 11$, and $9.91, 7.39, 5.22$ for the respective $z = 1.0, 1.2, 1.5$ with (b) $q = 0.25, S = 11$.

than DP-FTS for the same level of privacy guarantee (Fig. C.1 in Appendix C.2.1). These results justify the practical significance of our DE technique (Section 6.3.2). We have also shown (Fig. C.2 in Appendix C.2.1) that both components in our DE technique (i.e., letting every agent explore only a local sub-region and giving more weights to those agents exploring the sub-region) are necessary for the performance of DE.

Fig. 6.3b explores the impact of the clipping threshold S. From Fig. 6.3b, an overly small S may hinder the performance since it causes too many vectors to be clipped, and an excessively large S may also be detrimental due to increasing the variance of the added Gaussian noise. This corroborates our analysis in Section 6.4.2. The value of $S = 11$, which delivers the best performance in Fig. 6.3b, has been chosen such that only a small percentage (0.8%) of the vectors are clipped. Fig. 6.4 shows the privacy-utility trade-off of our DP-FTS-DE algorithm induced by q and z. The results verify our theoretical insights regarding the impact of the parameters q and z on the privacy-utility trade-off (Section 6.4.2), i.e., a larger q (smaller z) leads to a better utility at the expense of a greater privacy loss.

6.5.2 Real-World Datasets

We perform experiments using three commonly used real-world datasets in FL and FBO (Dai et al., 2020b; Smith et al., 2017). The first two experiments use the same datasets and settings as those of the experiments on *landmine detection* and *human activity recognition* using mobile phone sensors in Chapter 5. The third experiment uses the images of handwritten characters by $N = 50$ persons (agents) from the EMNIST dataset (i.e., a commonly used benchmark in FL) (Cohen et al., 2017) and tunes 3 hyperparameters of a convolutional neural network used for image classification. In all three experiments, we choose $P = 4$ sub-regions, $S = 22.0$, $M = 100$, and $1 - p_t = 1/t, \forall t \in \mathbb{Z}^+$.

Figs. 6.5a,c,e plot the privacy (horizontal axis, more to the left is better) and utility (vertical axis, lower is better) after 60 iterations. The green dots correspond to $z = 1, 1.6, 2, 3, 4$ ($q = 0.35$) and the red dots represent $q = 0.1, 0.15, 0.2, 0.25, 0.35$ ($z = 1$). Results show that with small privacy loss (i.e., in the single digit range), DP-FTS-DE is able to achieve a competitive performance (utility) and significantly outperforms standard TS in most settings. The figures also reveal a clear trade-off between privacy and utility, i.e., a smaller privacy loss (more to the left) generally results in a worse utility (larger vertical value). In addition, these two observations can also be obtained from Figs. 6.5b,d,f which plot some convergence results:[11] DP-FTS-DE and FTS-DE converge faster than TS; a smaller privacy loss (i.e., larger z or smaller q) in general leads to a slower convergence. Figs. 6.5a,c,e also justify the importance of DE (Section 6.3.2) since FTS-DE (and some settings of DP-FTS-DE) significantly outperforms FTS without DE in all three experiments. We also verify the importance of DE when DP is integrated in Appendix C.2.2 (Fig. C.4). Lastly, our DP-FTS-DE can be easily adapted to use Rényi DP (Wang et al., 2019)[12], which, although requiring

[11] Here, we only show the average performances without error bars because the agents in these datasets are highly heterogeneous and thus have significantly different scales.

[12] We only need to modify step 6 of Algorithm 6.1 such that we randomly select a fixed number

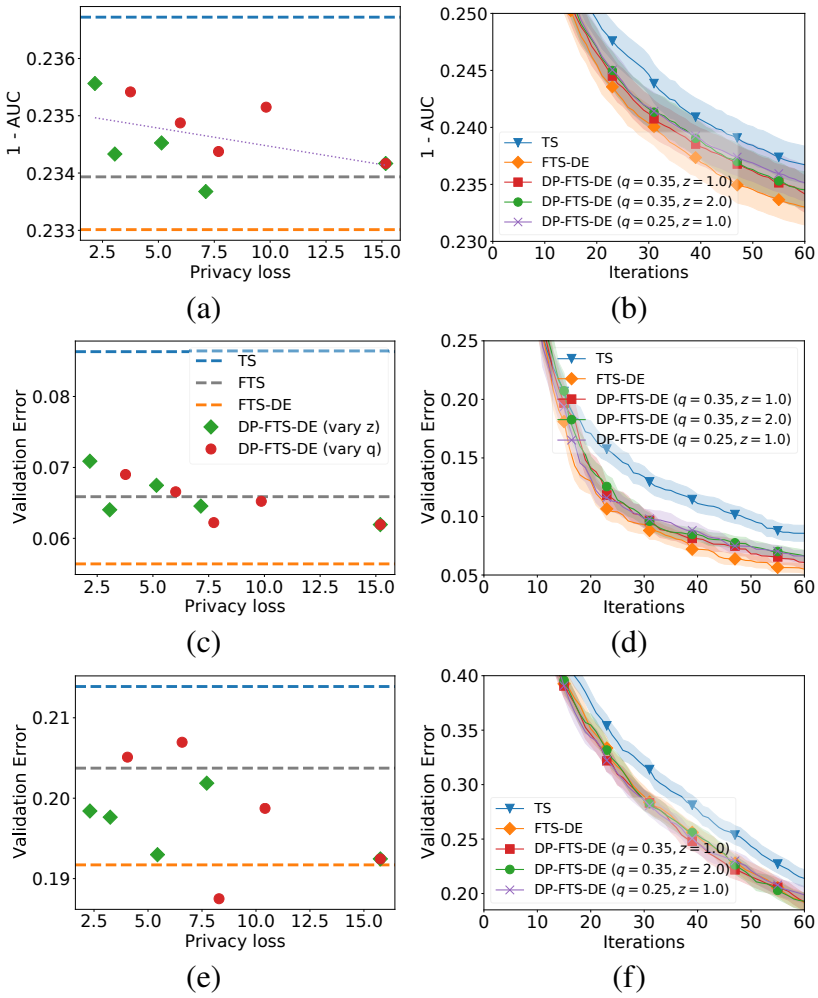


Figure 6.5: (a,c,e) Privacy loss vs. performance after 60 iterations for landmine detection, human activity recognition, and EMNIST. The more to the *left* (*bottom*), the better the privacy (utility). (b,d,f) Convergence results for some settings in each experiment. Results are averaged over N agents s.t. each agent's performance is further averaged over 10 diff. random initializations of size $N_{\text{init}} = 10$.

modification to our theoretical analysis, offers slightly better privacy loss with comparable performances (Fig. C.5 in Appendix C.2.2). Here, a better privacy loss indicates that for a fixed δ, Rényi DP achieves a tighter upper bound on the value of ϵ (note that the privacy loss is defined as an upper bound on ϵ as described in footnote 4).

of Nq agents in every iteration.

6.6 Conclusion

This chapter describes the first algorithm called DP-FTS-DE for the FBO setting with a rigorous guarantee for user-level privacy. DP-FTS-DE is equipped with theoretical guarantees for both the privacy and utility and amenable to theoretical insights about the privacy-utility trade-off. In practice, DP-FTS-DE achieves a competitive utility and demonstrates an interesting trade-off between privacy and utility.

Chapter 7

Robust Meta-Bayesian Optimization

7.1 Introduction

When using BO to optimize a *target function*, we sometimes have access to a set of evaluations of some potentially related functions. For example, when using BO for hyperparameter optimization of an ML model trained on a target dataset, we often have access to some previously completed BO tasks using other potentially related datasets (Golovin et al., 2017). These previous tasks, if similar to the current task, may be exploited to accelerate the current BO task. However, if some (or even all) previous tasks are in fact dissimilar to the current task, their use may turn out to incorporate harmful information and hence sabotage the convergence of BO (Feurer et al., 2018). This begs the question as to whether we can leverage these previous tasks to improve the efficiency of the current BO task, while ensuring robustness against these harmful dissimilar tasks such that (a) they would not affect the no-regret convergence of BO, and (b) we can identify them and thus diminish their impact in a principled manner.

Exploiting previous learning experiences to improve the efficiency of the

current task is the goal of *meta-learning* (Vanschoren, 2018). Meta-learning is a broad field with various applications in supervised learning (Finn et al., 2017), RL (Xu et al., 2018), active learning (Pang et al., 2018), among others. The major challenges in meta-learning include (a) the transfer of information from previous tasks to the current task, and (b) characterization of task similarity which is crucial for identifying harmful dissimilar tasks (Vanschoren, 2018). The application of meta-learning to BO (or *meta-BO*) has been explored by previous studies which differ in how these two challenges are addressed. Some works, such as multitask BO (Swersky et al., 2013), transfer the information from previous tasks by building a joint GP surrogate using the observations from all previous and current tasks, with the task similarity either represented by meta-features (Bardenet et al., 2013; Yogatama and Mann, 2014) or learned from observations (Swersky et al., 2013; Wang et al., 2018) (Section 3.4). Multi-fidelity BO methods (Kandasamy et al., 2016, 2017; Wu and Frazier, 2018; Zhang et al., 2017) can also be applied to meta-BO by considering the previous tasks as lower-fidelity observations and restricting online queries to only the target function. These works, however, are limited by the scalability of GP due to including all previous and current observations in a single GP (Feurer et al., 2018). To this end, other recent works transfer information from previous tasks using a more scalable approach: They build a separate GP surrogate for each individual task and use a weighted combination of either the individual surrogate functions or acquisition functions for query selection (Feurer et al., 2018; Wistuba et al., 2016, 2018). However, these works on scalable meta-BO do not provide a theoretical performance guarantee to ensure robust performances in the presence of harmful dissimilar tasks, and use heuristics to attempt to identify these dissimilar tasks. If not handled in a principled manner, these dissimilar tasks might mislead the current BO algorithm by providing poisonous information, consequently resulting in a suboptimal convergence.

This chapter presents a scalable, principled and robust meta-BO algorithm: *robust meta-GP-upper confidence bound* (RM-GP-UCB). Like the works of Feurer et al. (2018); Wistuba et al. (2016, 2018), we transfer information from previous tasks by combining individual acquisition functions. In particular, we compute the GP-UCB acquisition function for each individual task (including the previous and current tasks) and employ a weighted combination of these acquisition functions to select the next query. As a result, a separate GP surrogate is built for each previous task, making RM-GP-UCB scale well in the number of meta-tasks and observations in each meta-task. In stark contrast to the works of Feurer et al. (2018); Wistuba et al. (2016, 2018), RM-GP-UCB achieves *principled robustness* against harmful dissimilar tasks as a result of our two major contributions: RM-GP-UCB (a) is endowed with a theoretical convergence guarantee that is robust against dissimilar previous tasks, and (b) learns the task similarity to identify dissimilar tasks in a principled way through online learning. Firstly, we derive an upper bound on the regret of RM-GP-UCB and show that RM-GP-UCB is asymptotically no-regret for *any* given set of previous tasks, i.e., without requiring assumptions on the similarity between the previous and current tasks (Section 7.4). This allows us to guarantee robust performances of RM-GP-UCB in a wide range of applications, even when some or all previous tasks are significantly dissimilar to the target task. Moreover, we also use our theoretical analysis to show that when the meta-tasks are similar to the target task, they can help RM-GP-UCB converge faster than GP-UCB at the initial stage by accelerating the exploration process. Secondly, the theoretical guarantee allows us to learn the task similarity (and hence identify dissimilar tasks) in a principled way, by minimizing the upper bound on the regret of RM-GP-UCB via a computationally cheap online learning algorithm known as *Follow-The-Regularized-Leader* (Section 7.5). We demonstrate in Section 7.6 that, as a result of its robust performance guarantee and principled learning of task similarity, RM-GP-UCB performs effectively and

consistently across a number of interesting applications.

7.2 Formulation of Meta-Bayesian Optimization

We refer to the function f being maximized as the *target function* and the functions f_i for $i = 1, \ldots, M$ of the M previous tasks as *meta-functions*. We use *target task/observations* and *meta-tasks/observations* in a similar manner. All functions are defined on the same domain $\mathcal{X}$. We assume that f and all f_i's are sampled from the same GP with kernel k. This assumption is justified in the sense that the target function and meta-functions are assumed to be generated from *the same underlying phenomenon* (e.g., performance of ML models trained using the activity recognition data of different individuals), and thus have the same degree of smoothness characterized by the kernel k. We also assume that all meta- and target observations are corrupted by a Gaussian noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ with variance σ^2. These two assumptions are in line with the work of Wang et al. (2018) which has also performed theoretical analysis of a meta-learning algorithm for BO (Section 3.4). The number of observations from meta-task i is a constant denoted as N_i, and $N \triangleq \max_{i=1,\ldots,M} N_i$. $\mathbf{x}_{i,j}$ and $y_{i,j}$ represent the j-th input and noisy output of meta-task i respectively. We define the *function gap* $d_i \triangleq \max_{j=1,\ldots,N_i} \left|f(\mathbf{x}_{i,j}) - f_i(\mathbf{x}_{i,j})\right| < \infty$ which represents the maximum difference between the function values of f and f_i at any corresponding input $\mathbf{x}_{i,j}$ of meta-task i. Note that for a given set of meta-observations for meta-task i, the function gap d_i is an unknown constant characterizing the similarity between meta-task i and the target task: a smaller function gap implies a stronger similarity. Our algorithm is designed to be robust such that it performs effectively even when some or all function gaps are large, therefore, we do not place assumptions on the magnitude of the functions gaps except that they are finite.

7.3 Robust Meta-Gaussian Process-Upper Confidence Bound (RM-GP-UCB)

The acquisition function (7.1) adopted by RM-GP-UCB in iteration t is a weighted combination of $M+1$ individual GP-UCB acquisition functions (Srinivas et al., 2010) for the target task and the M meta-tasks, each of which is calculated using the observations from a particular task:

$$\overline{\zeta}_t(\mathbf{x}) \triangleq \nu_t\left[\sum\nolimits_{i=1}^{M}\omega_i\left[\overline{\mu}_i(\mathbf{x})+\sqrt{\tau}\overline{\sigma}_i(\mathbf{x})\right]\right]+(1-\nu_t)\left[\mu_{t-1}(\mathbf{x})+\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x})\right]. \tag{7.1}$$

In (7.1), $\mu_{t-1}(\mathbf{x})$ and $\sigma_{t-1}(\mathbf{x})$ represent, respectively, the GP posterior mean and standard deviation (2.1) at $\mathbf{x}$ calculated using the target observations from iterations 1 to $t-1$. $\overline{\mu}_i(\mathbf{x})$ and $\overline{\sigma}_i(\mathbf{x})$ are computed using all meta-observations from meta-task i. $\beta_t > 0$ and $\tau > 0$ will be defined in Section 7.4. $\nu_t \in [0,1]$ can be interpreted as the overall weight given to all meta-tasks in iteration t and should be chosen to be non-increasing in t, which enforces the impact of meta-tasks in (7.1) to be non-increasing. The *meta-weights* ω_i's form a probability simplex (all $\omega_i \geq 0$ and $\sum_{i=1}^{M}\omega_i = 1$) and can be understood as the weights assigned to individual meta-tasks. Note that since the dataset used to calculate $\overline{\mu}_i(\mathbf{x})$ and $\overline{\sigma}_i(\mathbf{x})$ is fixed with size N_i, the matrix inversion in (2.1) (i.e., computational bottleneck for GP) can be pre-computed. So, after T iterations, RM-GP-UCB incurs $\mathcal{O}(T^3)$ time for covariance matrix inversion (since only the target covariance matrix of size $T \times T$ needs to be inverted) and $\mathcal{O}(MN^2+T^2)$ time during predictive inference, which are less than the respective $\mathcal{O}((MN+T)^3)$ and $\mathcal{O}((MN+T)^2)$ time incurred when all observations are included in a single GP. Hence, RM-GP-UCB is scalable in the number of meta-tasks and observations in each meta-task. In iteration t (Algorithm 7.1), we first optimize the meta-weights and update ν_t (Section 7.5.2). Next, input $\mathbf{x}_t$ is selected by maximizing the acquisition

function (7.1), after which we query $\mathbf{x}_t$ and use the newly collected input-output pair $(\mathbf{x}_t, y_t)$ to update the GP posterior belief (2.1). Note that our acquisition function (7.1) may be modified in practice to use a weighted combination of other existing acquisition functions (e.g., expected improvement), however, our theoretical results (Sections 7.4 and 7.5) would not hold anymore.

Algorithm 7.1 RM-GP-UCB

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Update ω_i for $i = 1, \ldots, M$ via online meta-weight optimization and update ν_t (more details on these updates are given in Section 7.5.2)
3: $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x}\in\mathcal{D}} \overline{\zeta}_t(\mathbf{x})$ (7.1)
4: Query $\mathbf{x}_t$ to observe y_t, and update GP posterior belief (2.1) using $(\mathbf{x}_t, y_t)$

7.4 Theoretical Analysis

Theorem 7.1 presents the main theoretical result of this work and its proof is given in Appendix D.1:

Theorem 7.1. *Let* $\delta \in (0, 1)$. *Suppose that RM-GP-UCB is run with the following parameters:* $\nu_t \in [0, 1]$ *and* $\nu_{t+1} \leq \nu_t \; \forall t \geq 1$, $\tau \triangleq 2\log(3\,|\mathcal{X}|\,M/\delta)$, $\beta_t \triangleq 2\log(|\mathcal{X}|\,t^2\pi^2/2\delta) \; \forall t \geq 1$, $\omega_i \geq 0$ *for* $i = 1, \ldots, M$ *and* $\sum_{i=1,\ldots,M} \omega_i = 1$. *Then, with probability of at least* $1 - \delta$,

$$R_T \leq 2(\alpha + \sqrt{\tau})\sum_{t=1}^{T} \nu_t + \sqrt{C_1 T (1 - \nu_T)^2 \beta_T \gamma_T},$$

where $C_1 \triangleq \frac{8}{1+\sigma^{-2}}$, γ_T *is the maximum information gain about* f *from observing any set of* T *observations,* $\alpha \triangleq \sum_{i=1}^{M} \omega_i \alpha_i$, *and* $\alpha_i \triangleq \frac{N_i}{\sigma^2}(2\sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}} + d_i\sqrt{N_i})$.

If ν_t is designed according to Corollary 7.1 below, both terms in the regret upper bound grow sublinearly, which suggests that RM-GP-UCB is asymptotically no-regret for some commonly used kernels (Srinivas et al., 2010).

Corollary 7.1. *If $\nu_t \to 0$ as $t \to \infty$, then RM-GP-UCB achieves no regret asymptotically for the Squared Exponential (SE) and Mátern kernels.* [1]

Our theoretical results hold for a given set of meta-tasks with fixed yet unknown d_i's. Note that we do not impose assumptions on the values of d_i's, i.e., the similarities between the meta- and target tasks. Therefore, Theorem 7.1 gives a robust regret upper bound which holds for *any* given set of meta-tasks. In other words, even in adverse scenarios where some or all meta-tasks are extremely dissimilar to the target task (i.e., when some or all d_i's are very large), RM-GP-UCB is still asymptotically no-regret, which indicates the robustness and generality of our algorithm. In our proof, the key step (Lemma D.1 in Appendix D.1) is to upper bound the overall error induced by the use of any given set of meta-observations, instead of the target observations at the same corresponding input locations, when calculating the acquisition function (7.1). This provides an assurance about the *worst-case behavior* in *any* given scenario.[2] Concretely, α_i in Theorem 7.1 can be interpreted as an upper bound on the cumulative error incurred by using the meta-observations of f_i in calculating the acquisition function (7.1). Similarly, α can be viewed an upper bound on the overall error induced by the use of all meta-observations. These interpretations also explain the dependence of α, hence the regret upper bound, on d_i and N_i: Larger function gaps increase the error resulting from the use of the meta-observations, and a larger number of meta-observations also inflates the worst-case upper bound by accumulating the individual errors[3]. On the other hand, to achieve robustness against dissimilar tasks, we may sacrifice our ability to fully utilize

[1] Recall we have mentioned in Section 2.2 that for the SE kernel, $\gamma_T = \mathcal{O}((\log T)^{d+1})$; for the Matérn kernel with parameter $\nu > 1$, $\gamma_T = \mathcal{O}(T^{d(d+1)/(2\nu+d(d+1))} \log T)$. Therefore, for both kernels, the second term in Theorem 7.1 grows sublinearly.

[2] This notion of robustness is in line with that of *robust optimization* (RO) Beyer and Sendhoff (2007) which also attempts to optimize the performance in the worst-case scenario. The difference is that RO optimizes an explicit objective, while we aim at preserving the no-regret property in the worst case.

[3] Note that a larger N_i is also likely to increase the value of d_i, as a result of the definition of d_i (Section 7.2).

the similar meta-tasks to improve the convergence, which may be a limitation of our RM-GP-UCB algorithm.

Meta-tasks Can Improve the Convergence by Accelerating Exploration. In addition to characterizing the worst-case behavior, we also exploit our theoretical analysis to illustrate how meta-tasks can help RM-GP-UCB converge faster than standard GP-UCB. As we have proved in Appendix D.1.3, at the early stage of the algorithm, the meta-tasks (if similar to the target task) can help RM-GP-UCB obtain a smaller regret upper bound than GP-UCB by *reducing the uncertainty at the selected input*. Equivalently, the additional information from the meta-tasks allows RM-GP-UCB to *reduce the degree of exploration at the early stage*. Since the initial exploration phase of BO usually incurs large regrets, less exploration results in smaller regrets. At later stages when ν_t becomes close to 0, RM-GP-UCB converges to no regret at a similar rate to GP-UCB.

Besides the above insights, Theorem 7.1 provides two natural hints to the practical design of RM-GP-UCB. Firstly, note that Theorem 7.1 holds for all choices of meta-weights ω_i's as long as they form a probability simplex. So, we have the flexibility to choose the optimal ω_i's (i.e., learn the task similarity) by minimizing the regret upper bound (Theorem 7.1). Secondly, the interpretations of α and ν_t suggest that we can make the influence of the meta-tasks (quantified by ν_t) decay faster if the error produced by using the meta-tasks (measured by its upper bound α) is larger. Both design choices can strengthen the robustness of RM-GP-UCB against dissimilar meta-tasks by lessening their impact. Unfortunately, they both require the value of α, which we lack access to since the function gaps d_i's are unavailable. To this end, we devise a principled technique to estimate upper bounds on the function gaps, which is presented in Section 7.5.1.

7.5 Online Meta-Weight Optimization

In this section, we first introduce a principled technique for estimating high-probability upper bounds on the function gaps (Section 7.5.1) that, when combined with Theorem 7.1, naturally yields a principled method for optimizing the meta-weights through regret minimization via online learning.

7.5.1 Online Estimation of Function Gaps

Inspired by the confidence region constructed by GP-UCB (Srinivas et al., 2010) that contains the target function with high probability, after $t \geq 1$ target observations have been collected, define

$$\begin{aligned} U_{t,i,j} &\triangleq \mu_t(\mathbf{x}_{i,j}) + \sqrt{\beta_{t+1}}\sigma_t(\mathbf{x}_{i,j}) \ , \\ L_{t,i,j} &\triangleq \mu_t(\mathbf{x}_{i,j}) - \sqrt{\beta_{t+1}}\sigma_t(\mathbf{x}_{i,j}), \end{aligned} \tag{7.2}$$

where $\mathbf{x}_{i,j}$ is the j-th input of meta-task i, β_{t+1} is previously defined in Theorem 7.1, and $U_{t,i,j}$ and $L_{t,i,j}$ can be interpreted, respectively, as the upper and lower confidence bounds of f at $\mathbf{x}_{i,j}$ after t iterations. Lemma D.2 (Appendix D.1) implies that with probability of at least $1-\delta$ (δ is previously defined in Theorem 7.1): $L_{t,i,j} \leq f(\mathbf{x}_{i,j}) \leq U_{t,i,j}, \forall t, i, j$. Consequently, the following result gives high-probability upper bounds on the function gaps (proof in Appendix D.1.4):

Lemma 7.1. *Let $\delta, \delta' \in (0,1)$. Then, with probability of at least $1-\delta-\delta'$,*

$$\begin{aligned} d_i \leq \sqrt{2\sigma^2 \log\left[\left(2\sum\nolimits_{i=1}^{M} N_i\right)/\delta'\right]} + \\ \max_{j=1,\ldots,N_i} \left[\max\{|y_{i,j} - U_{t,i,j}|, |y_{i,j} - L_{t,i,j}|\}\right] \triangleq \overline{d}_{i,t}, \end{aligned}$$

for $t = 1, \ldots, T$ and $i = 1, \ldots, M$.

Unlike d_i, $\overline{d}_{i,t}$ can be efficiently calculated as its incurred time is linear in both M and N. Note that although the theoretical upper bound on d_i's from

Lemma 7.1 may be loose, it provides some useful practical insights on online meta-weight optimization, which are detailed in the next section.

7.5.2 Online Meta-Weight Optimization through Regret Minimization

Lemma 7.1 and Theorem 7.1 allow us to derive the following result (proof in Appendix D.1.5):

Proposition 7.1. *Let* $\delta, \delta' \in (0,1)$. *Then, with probability of at least* $1-\delta-\delta'$,

$$R_T \leq \frac{2}{\sigma^2}\left[\sum_{t=1}^{T}\boldsymbol{\omega}^\top \boldsymbol{l}_t\right]\left[\sum_{t=1}^{T}\nu_t\right] + 2\sqrt{\tau}\sum_{t=1}^{T}\nu_t + \sqrt{C_1 T\left(1-\nu_T\right)^2 \beta_T \gamma_T},$$

where $\boldsymbol{\omega} \triangleq [\omega_i]_{i=1,\ldots,M}$, $\boldsymbol{l}_t \triangleq [l_{i,t}]_{i=1,\ldots,M}$, *and* $l_{i,t} \triangleq 2N_i^{3/2}\sqrt{2\sigma^2\log(6N_i/\delta)} + \overline{d}_{i,t}N_i^{3/2}$.

Note that $\boldsymbol{l}_t$ can be efficiently computed after the t-th observation is collected. The regret upper bound in Proposition 7.1 depends on ω_i's only through the term $\sum_{t=1}^{T}\boldsymbol{\omega}^\top\boldsymbol{l}_t$ which can be minimized to optimize the meta-weights. This constitutes an *online learning* problem with linear loss function and its solution $\boldsymbol{\omega}$ constrained to a probability simplex. An additional entropic regularization term is preferred so as to encourage a solution with a large entropy to stabilize it (Bubeck, 2011). This corresponds to encouraging the meta-weights to spread across a large number of meta-tasks, in order to discover as many similar meta-tasks as possible. As a result, by using $1/\eta$ ($\eta > 0$) as the regularization parameter, the optimized $\boldsymbol{\omega}$ in iteration $t > 1$ is obtained by solving the following optimization problem:

$$\boldsymbol{\omega} \triangleq \arg\min_{\boldsymbol{\omega}'} \sum_{s=1}^{t-1}\boldsymbol{\omega}'^\top\boldsymbol{l}_s + \eta^{-1}\sum_{i=1}^{M}\omega_i'\log\omega_i', \tag{7.3}$$

subject to the constraints: $\omega_i' \geq 0$ for all i and $\sum_{i=1}^{M}\omega_i' = 1$. When $t =$

1, the optimized $\boldsymbol{\omega}$ follows from optimizing only the entropic regularization term, thus naturally entailing the uniform distribution $\omega_i = 1/M$ for all i. Consequently, (7.3) corresponds exactly to the online learning algorithm called *Follow-The-Regularized-Leader* with an entropic regularizer (Bubeck, 2011) where η represents the learning rate. Its optimal solution in iteration t can be derived via Lagrange multiplier (Appendix D.1.6) as

$$\omega_i = \frac{e^{-\eta \sum_{s=1}^{t-1} l_{i,s}}}{\sum_{j=1}^{M} e^{-\eta \sum_{s=1}^{t-1} l_{j,s}}} \overset{(a)}{\approx} \frac{e^{-\eta N^{\frac{3}{2}} \sum_{s=1}^{t-1} \overline{d}_{i,s}}}{\sum_{j=1}^{M} e^{-\eta N^{\frac{3}{2}} \sum_{s=1}^{t-1} \overline{d}_{j,s}}}, \tag{7.4}$$

for $i = 1, \ldots, M$ where (a) follows from assuming that all N_i's are close to N for simplicity. With this simplification, the first (noise-correction) term in the expression of $\overline{d}_{i,t}$ from Lemma 7.1 also cancels out, thus leading to a neat and elegant update rule for ω_i which we use in all our experiments. As is evident from (7.4), the update of ω_i's in each iteration only involves computing $\overline{d}_{i,t}$'s (i.e., incurring $\mathcal{O}(MN)$ time), adding one term to the summation on the exponent ($\mathcal{O}(M)$ time), and a normalization step ($\mathcal{O}(M)$ time), all of which are computationally cheap; this is another factor contributing to the scalability of RM-GP-UCB. Intuitively, (7.4) assigns small weights to meta-tasks with a large cumulative estimated function gap which implies a less similar meta-task. So, RM-GP-UCB handles dissimilar meta-tasks in a principled way by reducing their impact.

In addition, $\overline{d}_{i,t}$ from Lemma 7.1 also allows for the estimation of an upper bound on α (Theorem 7.1) in each iteration (i.e., by simply replacing d_i with $\overline{d}_{i,t}$) and thus facilitates an adaptive selection of ν_t, as mentioned in Section 7.4. Specifically, we set $\nu_1 = 1$ and $\nu_t = \nu_{t-1} \times \min(r, (\sum_{i=1}^{M} \omega_i \overline{d}_{i,t})^{-\epsilon})$ for $t > 1$, in which we have dropped the constants independent of $\overline{d}_{i,t}$. $r \in (0, 1)$ represents the minimum decaying rate to ensure the monotonic decay of ν_t required by Corollary 7.1, and $\epsilon > 0$ controls the aggressiveness of the adaptive decay such

that a larger ϵ results in a faster decay. With this adaptive tuning scheme, when the overall estimated function gaps are larger (the meta-tasks are dissimilar), ν_t decays faster and thus the impact of the meta-tasks vanishes more quickly.

7.6 Experiments and Discussion

We use extensive real-world experiments to compare our RM-GP-UCB algorithm with *(1)* GP-UCB, two other recently introduced scalable meta-BO algorithms: *(2) ranking-weighted Gaussian process ensemble* (RGPE) (Feurer et al., 2018) and *(3) transfer acquisition function* (TAF) (Wistuba et al., 2018) (Section 3.4), *(4)* multitask BO (MTBO) (Swersky et al., 2013), and *(5)* the method from Wang et al. (2018) named *point estimate meta-BO* (PEM-BO). Since MTBO is relatively not scalable (Sections 7.1 and 3.4), we only apply it to those experiments with relatively small number of meta-tasks and observations for which MTBO is still computationally feasible. However, MTBO is found to be significantly more time-consuming than the other methods in these experiments (Section 7.6.3). We compare with PEM-BO in the experiment that is most favorable for this algorithm, i.e., with the largest number of meta-observations and a discrete domain (refer to Section 7.6.2 for more details). We set $\eta = 1/N^{3/2}$, $\epsilon = 0.7$ and $r = 0.7$ in all real-world experiments to demonstrate the robustness of our algorithm against the choice of these parameters. In practice, the upper bound on the function gap, $\overline{d}_{i,t}$, from Lemma 7.1 may be too conservative; so, we replace the outer $\max$ operator over $j = 1, ..., N_i$ with the empirical mean in our experiments.[4] Some details and results are deferred to Appendix D.2 due to lack of space. All error bars represent standard errors.

[4]We explore the difference between these two choices in Appendix D.2.3.

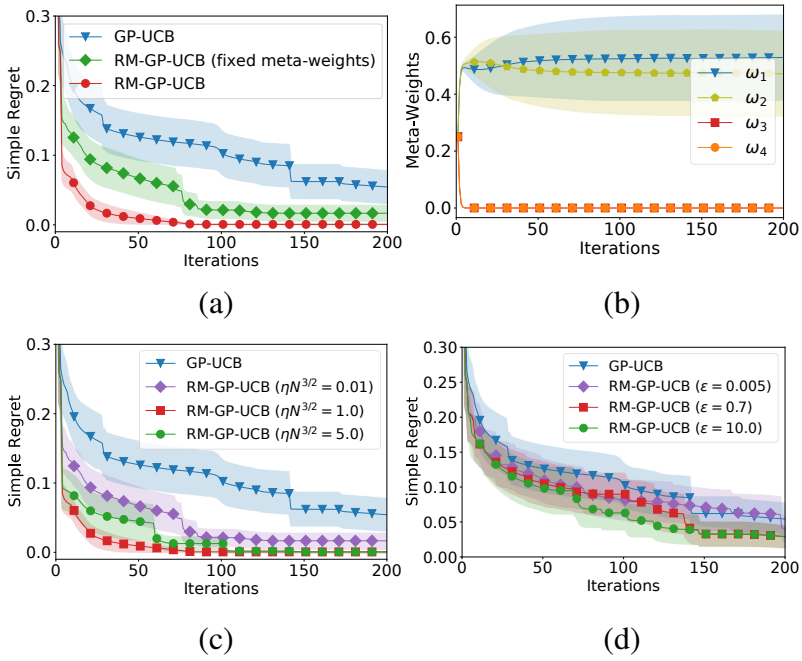


Figure 7.1: (a) The simple regret and (b) meta-weights optimized by RM-GP-UCB. The impact of (c) η and (d) ϵ.

7.6.1 Synthetic Experiments

We firstly optimize synthetic functions drawn from GPs. For each objective function, we construct $M = 4$ meta-tasks with $N = N_i = 20$ meta-observations each. The function gaps are chosen as $d_1 = d_2 = 0.05$ and $d_3 = d_4 = 4.0$ such that the last 2 meta-tasks are dissimilar to the target task. Fig. 7.1a plots the simple regrets averaged over 20 randomly drawn synthetic functions, with $\eta N^{3/2} = 1.0$, $\epsilon = 0.7$, and $r = 0.7$. The figure shows that RM-GP-UCB with fixed uniform meta-weights ($\omega_i = 1/4$ for all i) outperforms GP-UCB, and RM-GP-UCB with online meta-weight optimization (red) performs even better. Fig. 7.1b plots the meta-weights optimized by RM-GP-UCB for the red curve in Fig. 7.1a, showing that online meta-weight optimization (Section 7.5) rapidly reduces the weights given to the last two meta-tasks which are dissimilar to the target task. This shows the effectiveness of online meta-weight optimization in diminishing the impact of dissimilar meta-tasks (Fig. 7.1b), and verifies that it leads to performance improvement (Fig. 7.1a).

We also investigate the impact of η and ϵ. Fig. 7.1c shows the performances of different values of η, with fixed $\epsilon = 0.7$ and $r = 0.7$. The figure demonstrates that an excessively small η (purple curve) negatively impacts the performance, since RM-GP-UCB is unable to quickly reduce the weights of dissimilar meta-tasks (Fig. D.2a in Appendix D.2.1). Moreover, an overly large η is also slightly detrimental (green curve) since it rapidly assigns a large weight to one of the two useful meta-tasks (Fig. D.2c in Appendix D.2.1), thus failing to utilize the other useful meta-task. Fig. 7.1d illustrates the impact of ϵ when all function gaps are large: $d_i = 8.0$ for all i.[5] The figure shows that even when all meta-tasks are dissimilar, our adaptive selection of ν_t is able to diminish their negative impact and allow RM-GP-UCB to perform comparably to GP-UCB. Furthermore, in this adverse scenario, a faster decline of the impact of the meta-tasks (i.e., faster decay of η_t via larger ϵ) leads to slightly better performance. Fig. 7.1d verifies our robustness against dissimilar meta-tasks, which can be attributed to our robust convergence guarantee and online meta-weight optimization.

7.6.2 Real-world Experiments

Hyperparameter Tuning for Convolutional Neural Networks (CNN). Here, we apply meta-BO to hyperparameter tuning of ML models with the previous tasks using other datasets as the meta-tasks. We tune 3 hyperparameters of CNN using 4 widely used image datasets: MNIST, SVHN, CIFAR-10 and CIFAR-100. Specifically, in each experiment, one of the four datasets is selected to produce the target function f which maps a hyperparameter setting to a validation accuracy obtained using this dataset. The meta-observations are generated from 3 independent BO tasks (each with 50 iterations) using the other 3 datasets, i.e., $M = 3$ and $N_i = 50$ for $i = 1, 2, 3$ in all 4 experiments. The results for MNIST

[5] We use $\eta = 1/N^{3/2}$ and fix r at a large value (0.99) so that the decaying rate of ν_t is purely decided by ϵ.

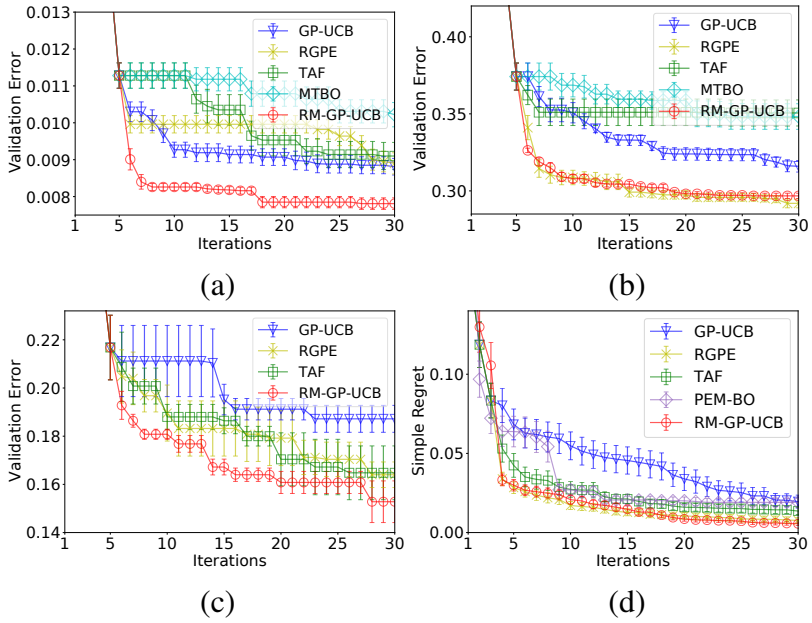


Figure 7.2: Best validation error of CNN for (a) MNIST and (b) CIFAR-10. (c) Best validation error of CNN using Omniglot. (d) Simple regret on SVM benchmark.

and CIFAR-10 are plotted in Figs. 7.2a and 7.2b while the remaining results are shown in Appendix D.2.2 (Fig. D.3). The performance of RM-GP-UCB is the best for MNIST (Fig. 7.2a) and comparable to RGPE in the other tasks (i.e., outperforming GP-UCB, TAF and MTBO). We also adopt the Omniglot dataset (Lake et al., 2015), which is widely used for meta-learning. Each task uses one alphabet, and involves tuning 3 hyperparameters of a Siamese network for one-shot classification, trained and validated using 75% and 25% of the alphabet respectively. We use 10 alphabets from background set as 10 meta-tasks and an alphabet from evaluation set as the target task. The 2-way validation errors (Fig. 7.2c) show that RM-GP-UCB outperforms all other algorithms under comparison.

Hyperparameter Tuning for Support Vector Machines (SVM). We also apply our algorithm to hyperparameter tuning for SVM using a tabular benchmark dataset (Wistuba et al., 2015a) which has also been adopted by RGPE (Feurer et al., 2018). The benchmark was constructed by evaluating a fixed grid of 288

SVM hyperparameter configurations using 50 diverse datasets (tasks). We follow the setting used by RGPE (Feurer et al., 2018): In every trial, we fix one of the tasks as the target task, and the remaining $M = 49$ tasks as the meta-tasks; for every meta-task i, we randomly select $N_i = 50$ hyperparameter configurations on the grid as the meta-observations. The results in Fig. 7.2d (averaged over 25 trials, each further averaged over 5 runs with random initializations) show that our RM-GP-UCB performs comparably to RGPE, outperforming GP-UCB, TAF and PEM-BO Wang et al. (2018). Of note, this experiment has the most favorable setting for PEM-BO because (a) PEM-BO has been shown to require a massive set of meta-observations (≥ 5000 in their experiments) to perform well (Wang et al., 2018), and this experiment has the largest number ($49 \times 50 = 2450$) of meta-observations among all our experiments; (b) the domain here is discrete, which is much easier for the application of PEM-BO.

Human Activity Recognition (HAR). HAR using mobile devices has promising applications in various domains such as healthcare (Reyes-Ortiz et al., 2013). When optimizing the configurations (hyperparameters) of the activity prediction model (ML model) for a subject, the previous optimization tasks for other subjects might be helpful. However, cross-subject transfer in HAR is challenging due to high individual variability (Soleimani and Nazerfard, 2019), which makes HAR suitable for evaluating the robustness of a meta-BO algorithm against dissimilar meta-tasks. We use the same dataset for human activity recognition using mobile phone sensors which has also been used in the experiments in Chapters 5 and 6. The dataset is collected through mobile phone sensors from 30 subjects performing 6 activities and we use SVM for activity prediction. Every task corresponds to tuning 2 SVM hyperparameters for a subject. We run a separate BO (30 iterations) for each of the first 21 subjects to generate the meta-observations ($M = 21$, $N_i = 30$ for $i = 1, \ldots, 21$) and use the other 9 subjects for validation. The validation error for each subject is averaged over

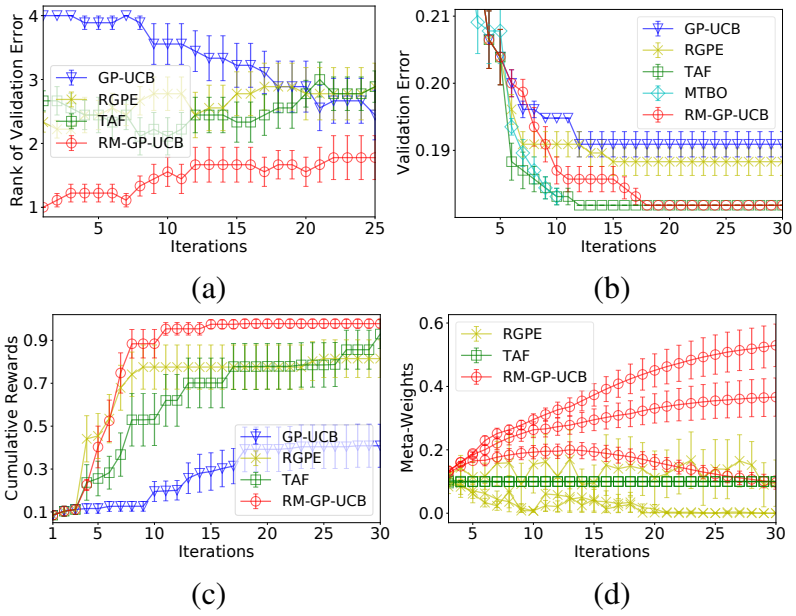


Figure 7.3: (a) Average rank of best validation errors for the HAR experiment. (b) Best validation error for diabetes diagnosis. (c) Best cumulative rewards and (d) learned meta-weights for the 3 similar meta-tasks for the RL experiment.

10 random initializations (shown in Fig. D.4, Appendix D.2.2). Then, for each tested algorithm and in every iteration, the rank (among the 4 algorithms) of their corresponding validation error is averaged over the 9 subjects (Fig. 7.3a), which shows that RM-GP-UCB outperforms all other algorithms.[6]

Non-stationary Bayesian Optimization. Meta-BO can be naturally applied to non-stationary BO problems in which the unknown objective function evolves over time since the previous (outdated) observations can be treated as the meta-observations. We consider here automated ML for clinical diagnosis. As the data from new patients becomes available regularly, clinicians often need to periodically update the dataset and re-run hyperparameter optimization for the ML model used for clinical diagnosis. This stimulates the question as to whether the previous hyperparameter tuning tasks using the outdated patients data can help accelerate the current task. We consider here the problem of diabetes prediction (Smith et al., 1988) with *logistic regression* (LR) and tune 3 LR

[6]The average ranks are shown since the subjects have different scales of validation errors.

hyperparameters. We create 5 progressively growing datasets (including the full dataset), treating (the hyperparameter tuning task using) the full dataset as the target task and the 4 smaller datasets as the meta-tasks. The results (Fig. 7.3b) show that MTBO and TAF perform the best, and RM-GP-UCB follows closely, outperforming both RGPE and GP-UCB.

Policy Search for Reinforcement Learning (RL). When optimizing the RL policy of an agent in an environment, the agent's experience in other related environments may help to make learning more efficient (Duan et al., 2016; Wang et al., 2016). We apply meta-BO to policy search in RL to maximize the cumulative rewards in an episode, using the Cart-Pole environment from OpenAI Gym (Brockman et al., 2016). We adopt a linear policy, i.e., every policy is represented by a 4×2 matrix which linearly maps a state vector of length 4 to an action vector of length 2. As a result, for a given policy matrix in a particular state, the action with the largest value in the mapped action vector is taken. The performance metric used in the experiment is the cumulative rewards (normalized to the range $[0, 1]$) in an episode (averaged over 10 independent episodes), and the maximum length of each episode is set to 200. We simulate different environments by setting the agent to different initial states. In particular, we choose $M = 10$ different initial states, among which 7 are randomly generated (i.e., dissimilar meta-tasks) and the other 3 are designed to be close to the initial state of the target task so that they are similar to the target task. An independent BO task with 50 iterations is run for every initial state, i.e., $N_i = 50$ for $i = 1, \ldots, 10$. Figs. 7.3c and 7.3d plot the (normalized) cumulative rewards of different algorithms and their learned meta-weights for the 3 similar meta-tasks. The results show that RM-GP-UCB achieves the best performance (Fig. 7.3c), because it is more effective at identifying the 3 similar meta-tasks and thus diminishing the impact of the remaining dissimilar meta-tasks (Fig. 7.3d). RGPE and TAF fail to correctly identify similar meta-tasks because they learn the

meta-weights based on how accurately each GP surrogate predicts the *pairwise ranking of the target observations* (more details in Section 3.4). However, in this case, many target observations have equal values, which confuses the pairwise ranking and makes the learned meta-weights unreliable.

7.6.3 Experimental Discussion

In most of our experimental results (Figs. 7.1, 7.2 and 7.3), the performance advantage of RM-GP-UCB is most evident at the initial stage. This is likely to corroborate our theoretical insights (Section 7.4) that the meta-tasks can help improve the convergence of RM-GP-UCB at the initial stage by reducing the degree of exploration. We also empirically demonstrate the scalability of our method (results in Appendix D.2.4) by (a) showing that the runtime of RM-GP-UCB is much smaller than MTBO in the non-stationary BO experiment, and (b) performing a more large-scale version of the RL experiment with 7800 meta-observations (60 meta-tasks, each containing 130 meta-observations).

A commonly raised issue in meta-BO is that the meta-functions may have different scales from the target function, i.e., the meta-functions may be shifted or scaled versions of the target function (Feurer et al., 2018). Nonetheless, this problem is not explicitly addressed here for the following reasons: Firstly, in some scenarios, the scale of the meta-functions is informative about task similarity and thus should not be removed. For example, in our clinical diagnosis experiment, the more recently completed meta-tasks (with larger training set, smaller validation errors, and thus smaller function gaps) are expected to be more similar to the target task. Secondly, as demonstrated by the green curve in Fig. 7.1a, even though the meta-weights are not optimized, RM-GP-UCB still performs favorably, thus justifying its robustness against mis-specification of the meta-weights. Moreover, RM-GP-UCB performs favorably throughout all experiments. Specifically, RM-GP-UCB outperforms standard GP-UCB in

all real-world experiments, while TAF (RGPE) fails to outperform standard GP-UCB in Figs. 7.2a and 7.2b (Figs. 7.2a and 7.3a), which is likely to result from the negative impact of harmful dissimilar meta-tasks. This might suggest RM-GP-UCB's superior ability to prevent the convergence of BO from being affected by dissimilar meta-tasks, which is believed to be largely due to its theoretically guaranteed convergence even when faced with dissimilar meta-tasks (Section 7.4) and its ability to identify dissimilar meta-tasks in a principled way (Section 7.5).

7.7 Conclusion

This chapter has introduced RM-GP-UCB, a scalable, principled and robust meta-BO algorithm that is asymptotically no-regret even when all meta-tasks are dissimilar to the target task. The regret upper bound of RM-GP-UCB is minimized via online learning to learn task similarity and identify harmful dissimilar tasks. RM-GP-UCB achieves competitive and consistent performances in a wide range of real-world experiments.

Chapter 8

Bayesian Optimization with Recursive Reasoning for Games

This chapter is based on the following paper published at ICML 2020:

Dai, Z., Chen, Y., Low, K. H., Jaillet, P., & Ho, T. H. (2020). R2-B2: Recursive reasoning-based Bayesian optimization for no-regret learning in games. In *Proc. ICML*.

8.1 Introduction

Several fundamental machine learning tasks in the real world involve intricate interactions between boundedly rational[1], self-interested agents that can be modeled as a form of repeated games with unknown, complex, and costly-to-evaluate payoff functions for the agents. For example, in adversarial *machine learning* (ML), the interactions between the *defender* $\mathcal{D}$ and the *attacker* $\mathcal{A}$ of an ML model can be modeled as a repeated game in which the payoffs to $\mathcal{D}$ and $\mathcal{A}$ are the performance of the ML model (e.g., validation accuracy) and its negation, respectively. Specifically, given a fully trained image classification

[1]Boundedly rational agents are subject to limited cognition and time in making decisions (Gigerenzer and Selten, 2002).

model (say, provided as an online service), $\mathcal{A}$ attempts to fool the ML model into misclassification through repeated queries of the model using perturbed input images. On the other hand, for each queried image that is perturbed by $\mathcal{A}$, $\mathcal{D}$ tries to ensure the correctness of its classification by transforming the perturbed image before feeding it into the ML model. As another example, *multi-agent reinforcement learning* (MARL) in an episodic environment can also be modeled as a repeated game in which the payoff to each agent is its return from the execution of all the agents' selected policies.

Solving such a form of repeated games in a cost-efficient manner is challenging since the payoff functions of the agents are unknown, complex (e.g., possibly noisy, non-convex, and/or with no closed-form expression/derivative), and costly to evaluate. Fortunately, the payoffs corresponding to different actions of each agent tend to be correlated. For example, in adversarial ML, the correlated perturbations performed by the attacker $\mathcal{A}$ (and correlated transformations executed by the defender $\mathcal{D}$) are likely to induce similar effects on the performance of the ML model. Such a correlation can be leveraged to *predict* the payoff associated with any action of an agent using a *surrogate* model such as the rich class of Bayesian nonparametric *Gaussian process* (GP) models (Rasmussen and Williams, 2006) which is expressive enough to represent a predictive belief of the unknown, complex payoff function over the action space of the agent. Then, in each iteration, the agent can select an action for evaluating its unknown payoff function that trades off between sampling at or near to a likely maximum payoff based on the current GP belief (exploitation) vs. improving the GP belief (exploration) until its cost/sampling budget is expended. To do this, the agent can use a sequential black-box optimizer such as the celebrated *Bayesian optimization* (BO) algorithm (Shahriari et al., 2016) based on the *GP-upper confidence bound* (GP-UCB) acquisition function (Srinivas et al., 2010), which guarantees asymptotic no-regret performance and is sample-efficient in practice. How then can we

design a BO algorithm to account for its interactions with boundedly rational[1], self-interested agents and still guarantee the trademark asymptotic no-regret performance?

Inspired by the cognitive hierarchy model of games (Camerer et al., 2004), we adopt a recursive reasoning formalism (i.e., typical among humans) to model the reasoning process in the interactions between boundedly rational[1], self-interested agents. It comprises k levels of reasoning which represents the cognitive limit of the agent. At level $k = 0$ of reasoning, the agent randomizes its choice of actions. At a higher level $k \geq 1$ of reasoning, the agent selects its best response to the actions of the other agents who are reasoning at lower levels $0, 1, \ldots, k-1$.

This chapter presents the first recursive reasoning formalism of BO to model the reasoning process in the interactions between boundedly rational[1], self-interested agents with unknown, complex, and costly-to-evaluate payoff functions in repeated games, which we call *Recursive Reasoning-Based BO* (R2-B2) (Section 8.3). R2-B2 provides these agents with principled strategies for performing effectively in this type of game. In this work, we consider repeated games with simultaneous moves and perfect monitoring[2]. Our R2-B2 algorithm is general in that it does not constrain the relationship among the payoff functions of different agents and can thus be applied to various types of games such as constant-sum games (e.g., adversarial ML in which the attacker $\mathcal{A}$ and defender $\mathcal{D}$ have opposing objectives), general-sum games (e.g., MARL where all agents have possibly different yet not necessarily conflicting goals), and common-payoff games (i.e., all agents have identical payoff functions). We prove that by reasoning at level $k \geq 2$ and one level higher than the other agents, our R2-B2 agent can achieve faster asymptotic convergence to no regret than that without utilizing recursive reasoning (Section 8.3.1.3). We also propose a

[2]In each iteration of a repeated game with (a) simultaneous moves and (b) perfect monitoring, every agent, respectively, (a) chooses its action simultaneously without knowing the other agents' selected actions, and (b) has access to the entire history of game plays, which includes all actions selected and payoffs observed by every agent in the previous iterations.

computationally cheaper variant of R2-B2 called R2-B2-Lite at the expense of a weaker convergence guarantee (Section 8.3.2). The performance and generality of R2-B2 are demonstrated through extensive experiments using synthetic games, adversarial ML, and MARL (Section 8.4). Interestingly, we empirically show that by reasoning at a higher level, our R2-B2 defender is able to effectively defend against the attacks from the state-of-the-art black-box adversarial attackers (Section 8.4.2.2), which can be of independent interest to the adversarial ML community.

8.2 Background and Problem Formulation

For simplicity, we will mostly focus on repeated games between two agents, but have extended our R2-B2 algorithm to games involving *more than two* agents, as detailed in Appendix E.2. To ease exposition, throughout this chapter, we will use adversarial ML as the running example and thus refer to the two agents as the *attacker* $\mathcal{A}$ and the *defender* $\mathcal{D}$. For example, the input action space $\mathcal{X}_1 \subset \mathbb{R}^{d_1}$ of $\mathcal{A}$ can be a set of allowed perturbations of a test image while the input action space $\mathcal{X}_2 \subset \mathbb{R}^{d_2}$ of $\mathcal{D}$ can represent a set of feasible transformations of the perturbed test image. We consider both input domains $\mathcal{X}_1$ and $\mathcal{X}_2$ to be discrete for simplicity; generalization of our theoretical results in Section 8.3 to continuous, compact domains can be easily achieved through a suitable discretization of the domains (Srinivas et al., 2010). When the ML model is an image classification model, the payoff function $f_1 : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathbb{R}$ of $\mathcal{A}$, which takes in its perturbation $\mathbf{x}_1 \in \mathcal{X}_1$ and $\mathcal{D}$'s transformation $\mathbf{x}_2 \in \mathcal{X}_2$ as inputs, can be the maximum predictive probability among all incorrect classes for a test image since $\mathcal{A}$ intends to cause misclassification. Since $\mathcal{A}$ and $\mathcal{D}$ have opposing objectives (i.e., $\mathcal{D}$ intends to prevent misclassification), the payoff function $f_2 : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathbb{R}$ of $\mathcal{D}$ can be the negation of that of $\mathcal{A}$, thus resulting

in a constant-sum game between $\mathcal{A}$ and $\mathcal{D}$.

In each iteration $t = 1, \ldots, T$ of the repeated game with simultaneous moves and perfect monitoring[2][3], $\mathcal{A}$ and $\mathcal{D}$ select their respective input actions $\mathbf{x}_{1,t}$ and $\mathbf{x}_{2,t}$ simultaneously using our R2-B2 algorithm (Section 8.3) for evaluating their payoff functions f_1 and f_2. Then, $\mathcal{A}$ and $\mathcal{D}$ receive the respective noisy observed payoffs $y_{1,t} \triangleq f_1(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}) + \epsilon_1$ and $y_{2,t} \triangleq f_2(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}) + \epsilon_2$ with i.i.d. Gaussian noises $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ and noise variances σ_i^2 for $i = 1, 2$.

A common practice in game theory is to measure the performance of $\mathcal{A}$ via its *(external) regret* (Nisan et al., 2007):

$$R_{1,T} \triangleq \sum_{t=1}^{T} [f_1(\mathbf{x}_1^*, \mathbf{x}_{2,t}) - f_1(\mathbf{x}_{1,t}, \mathbf{x}_{2,t})] \tag{8.1}$$

where $\mathbf{x}_1^* \triangleq \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \sum_{t=1}^{T} f_1(\mathbf{x}_1, \mathbf{x}_{2,t})$. The external regret $R_{2,T}$ of $\mathcal{D}$ is defined in a similar manner. An algorithm is said to achieve asymptotic *no regret* if $R_{1,T}$ grows sub-linearly in T, i.e., $\lim_{T\to\infty} R_{1,T}/T = 0$. Intuitively, by following a no-regret algorithm, $\mathcal{A}$ is guaranteed to eventually find its optimal input action $\mathbf{x}_1^*$ in hindsight, regardless of $\mathcal{D}$'s sequence of input actions.

To guarantee no regret (Section 8.3), $\mathcal{A}$ represents a predictive belief of its unknown, complex payoff function f_1 using the rich class of *Gaussian process* (GP) models by modeling f_1 as a sample of a GP (Rasmussen and Williams, 2006). $\mathcal{D}$ does likewise with its unknown f_2. A detailed background on GP has been given in Section 2.1, except that the input $\mathbf{x}$ in Section 2.1 need to be replaced with $[\mathbf{x}_1, \mathbf{x}_2]$. In particular, $\mathcal{A}$ uses the GP predictive/posterior belief of f_1 to compute the GP-UCB (Section 2.2) at any joint input actions $(\mathbf{x}_1, \mathbf{x}_2)$, which will be exploited by our R2-B2 algorithm (Section 8.3):

$$\alpha_{1,t}(\mathbf{x}_1, \mathbf{x}_2) \triangleq \mu_{t-1}(\mathbf{x}_1, \mathbf{x}_2) + \beta_t^{1/2} \sigma_{t-1}(\mathbf{x}_1, \mathbf{x}_2) \tag{8.2}$$

[3]Note that in some tasks such as adversarial ML, the requirement of perfect monitoring can be relaxed considerably. Refer to Section 8.4.2.2 for more details.

for iteration t where $\mu_{t-1}(\mathbf{x}_1, \mathbf{x}_2)$ and $\sigma^2_{t-1}(\mathbf{x}_1, \mathbf{x}_2)$ denote, respectively, the GP posterior mean and variance at $(\mathbf{x}_1, \mathbf{x}_2)$ (2.1) conditioned on the history of game plays up till iteration $t-1$ that includes $\mathcal{A}$'s observed payoffs and the actions selected by both agents in iterations $1, \ldots, t-1$. The GP-UCB acquisition function $\alpha_{2,t}$ for $\mathcal{D}$ is defined likewise. Supposing $\mathcal{A}$ knows the input action $\mathbf{x}_{2,t}$ selected by $\mathcal{D}$ and chooses an input action $\mathbf{x}_1$ to maximize the GP-UCB acquisition function $\alpha_{1,t}$ (8.2), its choice involves trading off between sampling close to an expected maximum payoff (i.e., with large GP posterior mean) given the current GP belief of f_1 (exploitation) vs. that of high predictive uncertainty (i.e., with large GP posterior variance) to improve the GP belief of f_1 (exploration) where the parameter β_t is set to trade off between exploitation vs. exploration for bounding its external regret (8.1), as specified later in Theorem 8.1.

8.3 Recursive Reasoning-Based Bayesian Optimization (R2-B2)

Algorithm 8.1 describes the R2-B2 algorithm from the perspective of *attacker* $\mathcal{A}$ which we will adopt in this section. Our R2-B2 algorithm for *defender* $\mathcal{D}$ can be derived analogously. We will now discuss the recursive reasoning formalism of BO for $\mathcal{A}$'s action selection in step 2 of Algorithm 8.1.

8.3.1 Recursive Reasoning Formalism of BO

Our recursive reasoning formalism of BO follows a similar principle as the cognitive hierarchy model (Camerer et al., 2004): At level $k = 0$ of reasoning, $\mathcal{A}$ adopts some randomized/mixed strategy of selecting its action. At level $k \geq 1$ of reasoning, $\mathcal{A}$ best-responds to the strategy of $\mathcal{D}$ who is reasoning at a lower level. Let $\mathbf{x}^k_{1,t}$ denote the input action $\mathbf{x}_{1,t}$ selected by $\mathcal{A}$'s strategy from reasoning at

Algorithm 8.1 R2-B2 for attacker $\mathcal{A}$'s level-k reasoning

1: **for** $t = 1, 2, \ldots, T$ **do**
2: Select input action $\mathbf{x}_{1,t}$ using its level-k strategy (while defender $\mathcal{D}$ selects input action $\mathbf{x}_{2,t}$)
3: Observe noisy payoff $y_{1,t} = f_1(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}) + \epsilon_1$
4: Update GP posterior belief using $\langle (\mathbf{x}_{1,t}, \mathbf{x}_{2,t}), y_{1,t} \rangle$

level k in iteration t. Depending on the (a) degree of knowledge about $\mathcal{D}$ and (b) available computational resource, $\mathcal{A}$ can choose one of the following three types of strategies of selecting its action with varying levels of reasoning, as shown in Fig. 8.1:

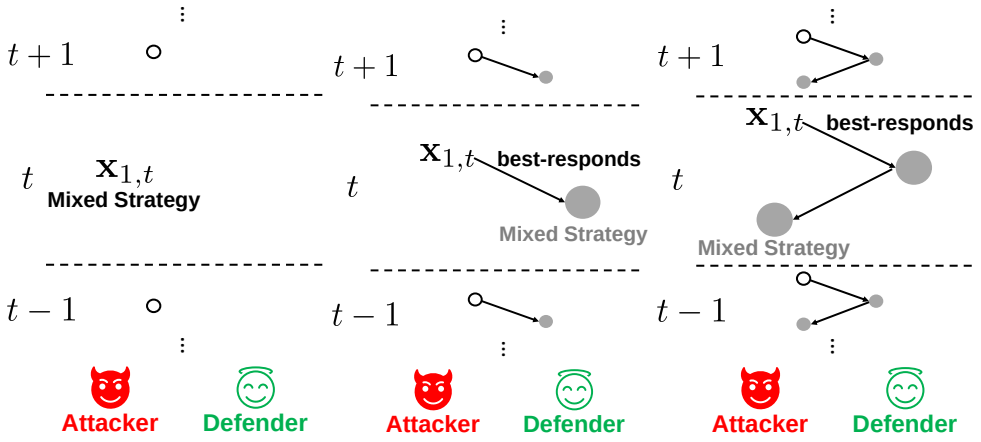
Level-$k = 0$ Strategy. Without knowledge of $\mathcal{D}$'s level of reasoning nor its level-0 strategy, $\mathcal{A}$ by default can reason at level 0 and play a mixed strategy $\mathcal{P}^0_{1,t}$ of selecting its action by sampling $\mathbf{x}^0_{1,t}$ from the probability distribution $\mathcal{P}^0_{1,t}$ over its input action space $\mathcal{X}_1$, as discussed in Section 8.3.1.1.

Level-$k = 1$ Strategy. If $\mathcal{A}$ thinks that $\mathcal{D}$ reasons at level 0 and has knowledge of $\mathcal{D}$'s level-0 mixed strategy $\mathcal{P}^0_{2,t}$, then $\mathcal{A}$ can reason at level 1 and play a pure strategy that best-responds to the level-0 strategy of $\mathcal{D}$, as explained in Section 8.3.1.2. Such a level-1 reasoning of $\mathcal{A}$ is general since it caters to *any* level-0 strategy of $\mathcal{D}$ and hence does not require $\mathcal{D}$ to perform recursive reasoning.

Level-$k \geq 2$ Strategy. If $\mathcal{A}$ thinks that $\mathcal{D}$ reasons at level $k-1$, then $\mathcal{A}$ can reason at level k and play a pure strategy that best-responds to $\mathcal{D}$'s level-$(k-1)$ action, as detailed in Section 8.3.1.3. Different from the level-1 reasoning of $\mathcal{A}$, its level-k reasoning assumes that $\mathcal{D}$'s level-$(k-1)$ action is derived using the same recursive reasoning process.

8.3.1.1 Level-$k = 0$ Strategy

Level 0 is a conservative, default choice for $\mathcal{A}$ since it does not require *any* knowledge about $\mathcal{D}$'s strategy of selecting its input action and is computationally



(a) Level 0 (b) Level 1 (c) Level 2

Figure 8.1: Illustration of attacker $\mathcal{A}$'s strategies of selecting its input action from reasoning at levels $k = 0$, 1, and 2.

lightweight. At level 0, $\mathcal{A}$ plays a mixed strategy $\mathcal{P}^0_{1,t}$ by sampling $\mathbf{x}^0_{1,t}$ from the probability distribution $\mathcal{P}^0_{1,t}$ over its input action space $\mathcal{X}_1$: $\mathbf{x}^0_{1,t} \sim \mathcal{P}^0_{1,t}$. A mixed/randomized strategy (instead of a pure/deterministic strategy) is considered because without knowledge of $\mathcal{D}$'s strategy, $\mathcal{A}$ has to treat $\mathcal{D}$ as a black-box adversary. This setting corresponds to that of an *adversarial bandit* problem in which any deterministic strategy suffers from linear worst-case regret (Lattimore and Szepesvári, 2020) and *randomization* alleviates this issue. Such a randomized design of our level-0 strategies is consistent with that of the cognitive hierarchy model in which a level-0 thinker does not make any assumption about the other agent and selects its action via a probability distribution without using strategic thinking (Camerer et al., 2004). We will now present a few reasonable choices of level-0 mixed strategies. However, in both theory (Theorems 8.2, 8.3 and 8.4) and practice, *any* strategy of action selection (including existing methods (Section 8.4.2.2)) can be considered as a level-0 strategy.

In the simplest setting where $\mathcal{A}$ has no knowledge of $\mathcal{D}$'s strategy, a natural choice for its level-0 mixed strategy is *random search*. That is, $\mathcal{A}$ samples its action from a uniform distribution over $\mathcal{X}_1$. An alternative choice is to use the *EXP3 algorithm* for the adversarial linear bandit problem, which requires the

GP to be transformed via a random features approximation (Rahimi and Recht, 2007) into linear regression with random features as inputs. Since the regret of EXP3 algorithm is bounded from above by $\mathcal{O}(\sqrt{d_1' T \log |\mathcal{X}_1|})$ (Lattimore and Szepesvári, 2020) where d_1' denotes the number of random features, it incurs sub-linear regret and can thus achieve asymptotic no regret.

In a more relaxed setting where $\mathcal{A}$ has access to the history of actions selected by $\mathcal{D}$, $\mathcal{A}$ can use the *GP-MW algorithm* (Sessa et al., 2019) to derive its level-0 mixed strategy; for completeness, GP-MW is briefly described in Appendix E.1. The result below bounds the regret of $\mathcal{A}$ when using GP-MW for level-0 reasoning and its proof is slightly modified from that of (Sessa et al., 2019) to account for its payoff function f_1 being sampled from a GP (Section 8.2):

Theorem 8.1. *Let $\delta \in (0,1)$, $\beta_t \triangleq 2\log(|\mathcal{X}_1| t^2 \pi^2/(3\delta))$, and γ_T denotes the maximum information gain about payoff function f_1 from any history of actions selected by both agents and corresponding noisy payoffs observed by $\mathcal{A}$ up till iteration T. Suppose that $\mathcal{A}$ uses GP-MW to derive its level-0 strategy. Then, with probability of at least $1-\delta$,*

$$R_{1,T} = \mathcal{O}(\sqrt{T \log |\mathcal{X}_1|} + \sqrt{T \log(2/\delta)} + \sqrt{T \beta_T \gamma_T}) \, .$$

From Theorem 8.1, $R_{1,T}$ is sub-linear in T for some commonly used kernels such as the SE kernel and Matérn kernel.[4] Therefore, $\mathcal{A}$ using GP-MW for level-0 reasoning achieves asymptotic no regret.

8.3.1.2 Level-$k = 1$ Strategy

If $\mathcal{A}$ thinks that $\mathcal{D}$ reasons at level 0 and has knowledge of $\mathcal{D}$'s level-0 strategy $\mathcal{P}^0_{2,t}$, then $\mathcal{A}$ can reason at level 1. Specifically, $\mathcal{A}$ selects its level-1 action $\mathbf{x}^1_{1,t}$

[4] Recall we have introduced in Section 2.2 that the asymptotic growth of γ_T has been analyzed for some commonly used kernels: $\gamma_T = \mathcal{O}((\log T)^{d_1+1})$ for the SE kernel and $\gamma_T = \mathcal{O}(T^{d_1(d_1+1)/(2\nu+d_1(d_1+1))} \log T)$ for Matérn kernel with parameter $\nu > 1$. For both kernels, the last term in the regret bound in Theorem 8.1 grows sub-linearly in T.

that maximizes the expected value of GP-UCB (8.2) w.r.t. $\mathcal{D}$'s level-0 strategy:

$$\mathbf{x}_{1,t}^1 \triangleq \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \mathbb{E}_{\mathbf{x}_{2,t}^0 \sim \mathcal{P}_{2,t}^0}[\alpha_{1,t}(\mathbf{x}_1, \mathbf{x}_{2,t}^0)] \ . \tag{8.3}$$

If input action space $\mathcal{X}_2$ of $\mathcal{D}$ is discrete and not too large, then (8.3) can be solved exactly. Otherwise, (8.3) can be solved approximately via sampling from $\mathcal{P}_{2,t}^0$. Such a level-1 reasoning of $\mathcal{A}$ to solve (8.3) only requires access to the history of actions selected by $\mathcal{D}$ but not its observed payoffs, which is the same as that needed by GP-MW. Our first main result (see its proof in Appendix E.3) bounds the expected regret of $\mathcal{A}$ when using R2-B2 for level-1 reasoning:

Theorem 8.2. *Let $\delta \in (0,1)$ and $C_1 \triangleq 8/\log(1+\sigma_1^{-2})$. Suppose that $\mathcal{A}$ uses R2-B2 (Algorithm 8.1) for level-1 reasoning and $\mathcal{D}$ uses mixed strategy $\mathcal{P}_{2,t}^0$ for level-0 reasoning. Then, with probability of at least $1-\delta$, $\mathbb{E}[R_{1,T}] \leq \sqrt{C_1 T \beta_T \gamma_T}$ where the expectation is with respect to the history of actions selected and payoffs observed by $\mathcal{D}$.*

It follows from Theorem 8.2 that $\mathbb{E}[R_{1,T}]$ is sublinear in T.[4] So, $\mathcal{A}$ using R2-B2 for level-1 reasoning achieves asymptotic no expected regret, which holds for *any* level-0 strategy of $\mathcal{D}$ regardless of whether $\mathcal{D}$ performs recursive reasoning.

8.3.1.3 Level-$k \geq 2$ Strategy

If $\mathcal{A}$ thinks that $\mathcal{D}$ reasons at level 1, then $\mathcal{A}$ can reason at level 2 and select its level-2 action $\mathbf{x}_{1,t}^2$ (8.4) to best-respond to level-1 action $\mathbf{x}_{2,t}^1$ (8.5) selected by $\mathcal{D}$, the latter of which can be computed/simulated by $\mathcal{A}$ in a similar manner as (8.3):

$$\mathbf{x}_{1,t}^2 \triangleq \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \alpha_{1,t}(\mathbf{x}_1, \mathbf{x}_{2,t}^1) \ , \tag{8.4}$$

$$\mathbf{x}_{2,t}^{1} \triangleq \arg\max_{\mathbf{x}_2 \in \mathcal{X}_2} \mathbb{E}_{\mathbf{x}_{1,t}^{0} \sim \mathcal{P}_{1,t}^{0}}[\alpha_{2,t}(\mathbf{x}_{1,t}^{0}, \mathbf{x}_2)] \; . \tag{8.5}$$

In the general case, if $\mathcal{A}$ thinks that $\mathcal{D}$ reasons at level $k-1 \geq 2$, then $\mathcal{A}$ can reason at level $k \geq 3$ and select its level-k action $\mathbf{x}_{1,t}^{k}$ (8.6) that best-responds to level-$(k-1)$ action $\mathbf{x}_{2,t}^{k-1}$ (8.7) selected by $\mathcal{D}$:

$$\mathbf{x}_{1,t}^{k} \triangleq \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \alpha_{1,t}(\mathbf{x}_1, \mathbf{x}_{2,t}^{k-1}) \; , \tag{8.6}$$

$$\mathbf{x}_{2,t}^{k-1} \triangleq \arg\max_{\mathbf{x}_2 \in \mathcal{X}_2} \alpha_{2,t}(\mathbf{x}_{1,t}^{k-2}, \mathbf{x}_2) \; . \tag{8.7}$$

Since $\mathcal{A}$ thinks that $\mathcal{D}$'s level-$(k-1)$ action $\mathbf{x}_{2,t}^{k-1}$ (8.7) is derived using the same recursive reasoning process, $\mathbf{x}_{2,t}^{k-1}$ best-responds to level-$(k-2)$ action $\mathbf{x}_{1,t}^{k-2}$ selected by $\mathcal{A}$, the latter of which in turn best-responds to level-$(k-3)$ action $\mathbf{x}_{2,t}^{k-3}$ selected by $\mathcal{D}$ and can be computed in the same way as (8.6). This recursive reasoning process continues until it reaches the base case of the level-1 action selected by either (a) $\mathcal{A}$ (8.3) if k is odd (in this case, recall from Section 8.3.1.2 that $\mathcal{A}$ requires knowledge of $\mathcal{D}$'s level-0 strategy $\mathcal{P}_{2,t}^{0}$ to compute (8.3)), or (b) $\mathcal{D}$ (8.5) if k is even. Note that $\mathcal{A}$ has to perform the computations made by $\mathcal{D}$ to derive $\mathbf{x}_{2,t}^{k-1}$ (8.7) as well as the computations to best-respond to $\mathbf{x}_{2,t}^{k-1}$ via (8.6). Our next main result (see its proof in Appendix E.3) bounds the regret of $\mathcal{A}$ when using R2-B2 for level-$k \geq 2$ reasoning:

Theorem 8.3. *Let $\delta \in (0,1)$. Suppose that $\mathcal{A}$ and $\mathcal{D}$ use R2-B2 (Algorithm 8.1) for level-$k \geq 2$ and level-$(k-1)$ reasoning, respectively. Then, with probability of at least $1-\delta$, $R_{1,T} \leq \sqrt{C_1 T \beta_T \gamma_T}$.*

Theorem 8.3 reveals that $R_{1,T}$ grows sublinearly in T.[4] So, $\mathcal{A}$ using R2-B2 for level-$k \geq 2$ reasoning achieves asymptotic no regret regardless of $\mathcal{D}$'s level-0 strategy $\mathcal{P}_{2,t}^{0}$. By comparing Theorems 8.1 and 8.3, we can observe that if $\mathcal{A}$ uses

GP-MW as its level-0 strategy, then it can achieve faster asymptotic convergence to no regret by using R2-B2 to reason at level $k \geq 2$ and one level higher than $\mathcal{D}$. However, when $\mathcal{A}$ reasons at a higher level k, its computational cost grows due to an additional optimization of the GP-UCB acquisition function per increase in level of reasoning. So, $\mathcal{A}$ is expected to favor reasoning at a lower level, which agrees with the observation in the work of (Camerer et al., 2004) on the cognitive hierarchy model that humans usually reason at a level no higher than 2.

8.3.2 R2-B2-Lite

We also propose a computationally cheaper variant of R2-B2 for level-1 reasoning called R2-B2-Lite at the expense of a weaker convergence guarantee. When using R2-B2-Lite for level-1 reasoning, instead of following (8.3), $\mathcal{A}$ selects its level-1 action $\mathbf{x}^1_{1,t}$ by sampling $\widetilde{\mathbf{x}}^0_{2,t}$ from level-0 strategy $\mathcal{P}^0_{2,t}$ of $\mathcal{D}$ and best-responding to this sampled action:

$$\mathbf{x}^1_{1,t} \triangleq \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \alpha_{1,t}(\mathbf{x}_1, \widetilde{\mathbf{x}}^0_{2,t}) \,. \tag{8.8}$$

Our final main result (its proof is in Appendix E.4) bounds the expected regret of $\mathcal{A}$ using R2-B2-Lite for level-1 reasoning:

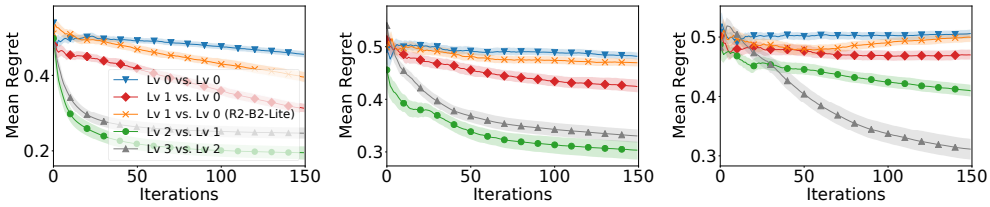
Theorem 8.4. *Let $\delta \in (0,1)$. Suppose that $\mathcal{A}$ uses R2-B2-Lite for level-1 reasoning and $\mathcal{D}$ uses mixed strategy $\mathcal{P}^0_{2,t}$ for level-0 reasoning. If the trace of the covariance matrix of $\mathbf{x}^0_{2,t} \sim \mathcal{P}^0_{2,t}$ is not more than ω_t for $t = 1, \ldots, T$, then with probability of at least $1-\delta$, $\mathbb{E}[R_{1,T}] = \mathcal{O}(\sum_{t=1}^{T} \sqrt{\omega_t} + \sqrt{T\beta_T\gamma_T})$ where the expectation is with respect to the history of actions selected and payoffs observed by $\mathcal{D}$ as well as $\widetilde{\mathbf{x}}^0_{2,t}$ for $t = 1, \ldots, T$.*

From Theorem 8.4, the expected regret bound tightens if $\mathcal{D}$'s level-0 mixed strategy $\mathcal{P}^0_{2,t}$ has a smaller variance for each dimension of input action $\mathbf{x}^0_{2,t}$. As a result, the level-0 action $\widetilde{\mathbf{x}}^0_{2,t}$ of $\mathcal{D}$ that is sampled by $\mathcal{A}$ tends to be closer to the

true level-0 action $\mathbf{x}^0_{2,t}$ selected by $\mathcal{D}$. Then, $\mathcal{A}$ can select level-1 action $\mathbf{x}^1_{1,t}$ that best-responds to a more precise estimate $\widetilde{\mathbf{x}}^0_{2,t}$ of the level-0 action $\mathbf{x}^0_{2,t}$ selected by $\mathcal{D}$, hence improving its expected payoff. Theorem 8.4 also reveals that $\mathcal{A}$ using R2-B2-Lite for level-1 reasoning achieves asymptotic no expected regret if the sequence $(\omega_t)_{t\in\mathbb{Z}^+}$ uniformly decreases to 0 (i.e., $\omega_{t+1} < \omega_t$ for $t \in \mathbb{Z}^+$ and $\lim_{T\to\infty} \omega_T = 0$). Interestingly, such a sufficient condition for achieving asymptotic no expected regret has a natural and elegant interpretation in terms of the exploration-exploitation trade-off: This condition is satisfied if $\mathcal{D}$ uses a level-0 mixed strategy $\mathcal{P}^0_{2,t}$ with a decreasing variance for each dimension of input action $\mathbf{x}^0_{2,t}$, which corresponds to transitioning from exploration (i.e., a large variance results in a diffused $\mathcal{P}^0_{2,t}$ and hence many actions being sampled) to exploitation (i.e., a small variance results in a peaked $\mathcal{P}^0_{2,t}$ and hence fewer actions being sampled).

8.4 Experiments and Discussion

This section empirically evaluates the performance of our R2-B2 algorithm and demonstrates its generality using synthetic games, adversarial ML, and MARL. Some of our experimental comparisons can be interpreted as comparisons with existing baselines used as level-0 strategies (Section 8.3.1.1). Specifically, we can compare the performance of our level-1 agent with that of a baseline method when they are against the same level-0 agent. Moreover, in constant-sum games, we can perform a more direct comparison by playing our level-1 agent against an opponent using a baseline method as a level-0 strategy (Section 8.4.2.2). Additional experimental details and results are reported in Appendix E.6 due to lack of space. All error bars represent standard error.



(a) (b) (c)

Figure 8.2: Mean regret of agent 1 in synthetic games for the (a) common-payoff game, (b) general-sum game, and (c) constant-sum game. The legend in (a) represents the levels of reasoning of agents 1 vs. 2.

8.4.1 Synthetic Games

Firstly, we empirically evaluate the performance of R2-B2 using synthetic games with two agents whose payoff functions are sampled from GP over a discrete input domain. Both agents use GP-MW and R2-B2/R2-B2-Lite for level-0 and level-$k \geq 1$ reasoning, respectively. We consider 3 types of games: common-payoff, general-sum, and constant-sum games. Figs. 8.2a to 8.2c show results of the mean regret[5] of agent 1 averaged over 10 random samples of GP and 5 initializations of 1 randomly selected action with observed payoff per sample: In all types of games, when agent 1 reasons at one level higher than agent 2, it incurs a smaller mean regret than when reasoning at level 0 (blue curve), which demonstrates the performance advantage of recursive reasoning and corroborates our theoretical results (Theorems 8.2 and 8.3). The same can be observed for agent 1 using R2-B2-Lite for level-1 reasoning (orange curve) but it does not perform as well as that using R2-B2 (red curve), which again agrees with our theoretical result (Theorem 8.4). Moreover, comparing the red (orange) and blue curves shows that when against the same level-0 agent, our R2-B2 (R2-B2-Lite) level-1 agent outperforms the baseline method of GP-MW (as a level-0 strategy).

Figs. 8.2a and 8.2c also reveal the effect of incorrect thinking of the level of

[5]The mean regret $T^{-1}\sum_{t=1}^{T}(\max_{\mathbf{x}_1\in\mathcal{X}_1,\mathbf{x}_2\in\mathcal{X}_2} f_1(\mathbf{x}_1,\mathbf{x}_2) - f_1(\mathbf{x}_{1,t},\mathbf{x}_{2,t}))$ of agent 1 pessimistically estimates (i.e., upper bounds) $R_{1,T}/T$ (8.1) and is thus not expected to converge to 0. Nevertheless, it serves as an appropriate performance metric here.

reasoning of the other agent on its performance: Since agent 2 uses recursive reasoning at level 1 or more, agent 2 thinks that it is reasoning at one level higher than agent 1. However, it is in fact reasoning at one level lower in these two figures. In common-payoff games, since agents 1 and 2 have identical payoff functions, the mean regret of agent 2 is the same as that of agent 1 in Fig. 8.2a. So, from agent 2's perspective, it benefits from such an incorrect thinking in common-payoff games. In constant-sum games, since the payoff function of agent 2 is negated from that of agent 1, the mean regret of agent 2 increases with a decreasing mean regret of agent 1 in Fig. 8.2c. So, from agent 2's viewpoint, it hurts from such an incorrect thinking in constant-sum games. Further experimental results on such incorrect thinking are reported in Appendix E.6.1.1b.

An intriguing observation from Figs. 8.2a to 8.2c is that when agent 1 reasons at level $k \geq 2$, it incurs a smaller mean regret than when reasoning at level 1. A possible explanation is that when agent 1 reasons at level $k \geq 2$, its selected level-k action (8.6) best-responds to the actual level-$(k-1)$ action (8.7) selected by agent 2. In contrast, when agent 1 reasons at level 1, its selected level-1 action (8.3) maximizes the *expected* value of GP-UCB w.r.t. agent 2's level-0 *mixed* strategy rather than the actual level-0 action selected by agent 2. However, as we shall see in the experiments on adversarial ML in Section 8.4.2.1, when the expectation in level-1 reasoning (8.3) needs to be approximated via sampling but insufficient samples are used, the performance of level-$k \geq 2$ reasoning can be potentially diminished due to propagation of the approximation error from level 1.

Moreover, Fig. 8.2c shows another interesting observation that is unique for constant-sum games: Agent 1 achieves a significantly better performance when reasoning at level 3 (i.e., agent 2 reasons at level 2) than at level 2 (i.e., agent 2 reasons at level 1). This can be explained by the fact that when agent 2 reasons at level 2, it best-responds to the level-1 action of agent 1, which

is most likely different from the actual action selected by agent 1 since agent 1 is in fact reasoning at level 3. In contrast, when agent 2 reasons at level 1, instead of best-responding to a single (most likely wrong) action of agent 1, it best-responds to the expected behavior of agent 1 by attributing a distribution over all actions of agent 1. As a result, agent 2 suffers from a smaller performance deficit when reasoning at level 1 (i.e., agent 1 reasons at level 2) compared with reasoning at level 2 (i.e., agent 1 reasons at level 3) or higher. Therefore, agent 1 obtains a more dramatic performance advantage when reasoning at level 3 (gray curve) due to the constant-sum nature of the game. A deeper implication of this insight is that although level-1 reasoning may not yield a better performance than level-$k \geq 2$ reasoning as analyzed in the previous paragraph, it is more robust against incorrect estimates of the opponent's level of reasoning in constant-sum games.

Experimental results on the use of random search and EXP3 (Section 8.3.1.1) for level-0 reasoning (instead of GP-MW) are reported in Appendix E.6.1.1c; the resulting observations and insights are consistent with those presented here. This demonstrates the robustness of R2-B2 and corroborates the generality of our theoretical results (Theorems 8.2 and 8.3) which hold for any level-0 strategy of the other agent. We have also performed experiments using synthetic games involving *more than two* agents (Appendix E.6.1.2), which yield some interesting observations that are consistent with our theoretical analysis.

8.4.2 Adversarial Machine Learning

8.4.2.1 R2-B2 for Adversarial ML

We apply our R2-B2 algorithm to black-box adversarial ML for image classification problems with *deep neural networks* (DNNs) using the MNIST and CIFAR-10 image datasets. We consider *evasion attacks* (Goodfellow et al., 2015; Yuan

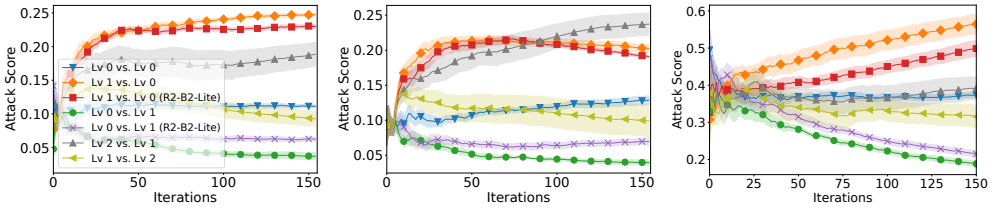
et al., 2019): The attacker $\mathcal{A}$ perturbs a test image in order to fool a fully trained DNN (referred to as the *target ML model* hereafter) into misclassifying the image, while the defender $\mathcal{D}$ transforms the perturbed image with the goal of ensuring the correct prediction by the classifier. To improve query efficiency, dimensionality reduction techniques such as autoencoders have been commonly used for black-box adversarial attacks (Tu et al., 2019). In our experiments, *variational autoencoders* (VAE) (Kingma and Welling, 2014) are used by both $\mathcal{A}$ and $\mathcal{D}$ to project the images to a lower-dimensional space (i.e., 2D for MNIST and 8D for CIFAR-10).[6] Following a common practice in adversarial ML, we focus on perturbations with bounded infinity norm as actions of $\mathcal{A}$ and $\mathcal{D}$: The maximum allowed perturbation to each pixel added by either $\mathcal{A}$ or $\mathcal{D}$ is no more than a pre-defined value ϵ where $\epsilon = 0.2$ for MNIST and $\epsilon = 0.05$ for CIFAR-10. We consider *untargeted attacks* whereby the goal of $\mathcal{A}$ ($\mathcal{D}$) is to cause (prevent) misclassification by the target ML model. So, the payoff function of $\mathcal{A}$ is the maximum predictive probability among all incorrect classes (referred to as *attack score* hereafter) and its negation is the payoff function of $\mathcal{D}$. As a result, the application of R2-B2 to black-box adversarial ML represents a *constant-sum game*. An attack is considered *successful* if the attack score is larger than the predictive probability of the correct class, hence resulting in misclassification of the test image. Both $\mathcal{A}$ and $\mathcal{D}$ use GP-MW/random search[7] and R2-B2/R2-B2-Lite for level-0 and level-$k \geq 1$ reasoning, respectively.

Figs. 8.3a to 8.3c show results of the attack score of $\mathcal{A}$ in adversarial ML for both image datasets while Table 8.1 shows results of the number of successful attacks by $\mathcal{A}$ over 150 iterations of the game; the results are averaged over 10 initializations of 5 randomly selected actions with observed payoffs.[8] It can be

[6] We have detailed in Appendix E.6.2.1a how VAE can be realistically incorporated into our algorithm.

[7] For CIFAR-10 dataset, $\mathcal{A}$ uses only random search for level-0 reasoning due to high dimensions, as explained in Appendix E.6.2.1a.

[8] The results here use a test image from each dataset that can clearly illustrate the effects of both attack and defense. Refer to Appendix E.6.2.1b for more details and results using more test



(a) (b) (c)

Figure 8.3: Attack score of $\mathcal{A}$ in adversarial ML for MNIST using (a) random search and (b) GP-MW as the level-0 strategy, and (c) CIFAR-10 using random search as the level-0 strategy. The legend in (a) represents the levels of reasoning of $\mathcal{A}$ vs. $\mathcal{D}$.

observed from Figs. 8.3a to 8.3c that when $\mathcal{A}$ reasons at one level higher than $\mathcal{D}$ (orange, red, and gray curves), its attack score is higher than when reasoning at level 0 (blue, green, and purple curves). Similarly, when $\mathcal{D}$ reasons at one level higher (green, purple, and yellow curves), the attack score of $\mathcal{A}$ is reduced. These observations demonstrate the performance advantage of using recursive reasoning in adversarial ML. Such an advantage of recursive reasoning can also be seen from Table 8.1: For MNIST, when random search is used for level-0 reasoning and $\mathcal{A}$ reasons at one level higher than $\mathcal{D}$, it achieves a larger number of successful attacks (12.8, 10.2, and 3.0) than when reasoning at level 0 (2.6, 0.8, and 1.8). Similarly, when $\mathcal{D}$ reasons at one level higher, it reduces the number of successful attacks by $\mathcal{A}$ (0.8, 1.8, and 0.9) than when reasoning at level 0 (2.6, 12.8, and 10.2). The observations are similar for MNIST with GP-MW for level-0 reasoning as well as for CIFAR-10 (Table 8.1).

The performance advantage of $\mathcal{A}$ reasoning at level 2 is observed to be smaller than that at level 1; this may be explained by the propagation of error of approximating the expectation in level-1 reasoning (8.3), as explained previously in Section 8.4.1. We investigate and report the effect of the number of samples for such an approximation in Appendix E.6.2.1c, which reveals that the performance improves with more samples, albeit with higher computational cost. Moreover,

images; the observations are consistent with those presented here.

Table 8.1: Average number of successful attacks by $\mathcal{A}$ over 150 iterations in adversarial ML for MNIST and CIFAR-10 datasets where the levels of reasoning are in the form of $\mathcal{A}$ vs. $\mathcal{D}$.

Levels of reasoning	MNIST (random)	MNIST (GP-MW)	CIFAR-10
0 vs. 0	2.6	4.3	70.1
1 vs. 0	12.8	6.0	113.1
1 vs. 0 (R2-B2-Lite)	10.2	6.8	99.7
0 vs. 1	0.8	0.4	25.2
0 vs. 1 (R2-B2-Lite)	1.8	1.0	29.7
2 vs. 1	3.0	5.2	70.9
1 vs. 2	0.9	0.4	54.0

some insights can also be drawn regarding the consequence of an incorrect thinking about the opponent's level of reasoning in constant-sum games. For example, for the gray curves in Figs. 8.3a to 8.3c, $\mathcal{D}$ reasons at level 1 because it thinks that $\mathcal{A}$ reasons at level 0. However, $\mathcal{A}$ is in fact reasoning at level 2. As a result, in this constant-sum game, $\mathcal{D}$'s incorrect thinking about the opponent's level of reasoning negatively impacts $\mathcal{D}$'s performance since the attack scores are increased. This is consistent with the corresponding analysis in synthetic games regarding the effect of incorrect thinking about the level of reasoning of the other agent (Section 8.4.1).

8.4.2.2 Comparison with State-of-the-art Adversarial Attack Methods

It was mentioned in Section 8.3.1 that our theoretical results hold for *any* level-0 strategy of the other agent. So, any existing adversarial attack (defense) method can be used as the level-0 strategy of $\mathcal{A}$ ($\mathcal{D}$). In this experiment, we perform a direct comparison of R2-B2 with the state-of-the-art black-box adversarial attack method called *Parsimonious* (Moon et al., 2019): We use Parsimonious as the level-0 strategy of $\mathcal{A}$ and let $\mathcal{D}$ use R2-B2 for level-1 reasoning. We consider a realistic setting where in each iteration, $\mathcal{D}$ only needs to receive the image perturbed by $\mathcal{A}$ and choose its action that best-responds to this perturbed image. In this manner, $\mathcal{D}$ naturally has access to the history of actions selected by $\mathcal{A}$ (as

required by *perfect monitoring* in our repeated game) since it receives all images perturbed by $\mathcal{A}$. Additional details of the experimental setting are reported in Appendix E.6.2.2a.

We randomly select 70 images from the CIFAR-10 dataset that are successfully attacked by Parsimonious using $\epsilon = 0.05$ over 500 iterations without the defender $\mathcal{D}$.[9] Our level-1 R2-B2 defender manages to *completely prevent any successful attacks* for 53 of these images and requires Parsimonious to use *more than* 3.5 *times* more queries on average to succeed for 10 other images.[10] Fig. 8.4 shows results of the loss incurred by Parsimonious (i.e., its original attack objective) with and without our level-1 R2-B2 defender for 4 of the successfully defended images; results for other images are shown in Appendix E.6.2.2a. This experiment not only demonstrates the generality of our R2-B2 algorithm, but can also be of significant independent interest to the adversarial ML community as a defense method against black-box adversarial attacks.

In addition, as another comparison, we use the same experimental setting with the CIFAR-10 dataset in Section 8.4.2.1 and play Parsimonious against a level-0 defender using random search. The results show that when against the same level-0 defender, Parsimonious achieves a significantly smaller average number of successful attacks (27.6) compared with our level-1 attacker (113.1, as shown in Table 8.1). In other words, our level-1 defender can defend effectively against Parsimonious, while our level-1 attacker can attack better than Parsimonious. Note that the unsatisfactory performances of Parsimonious in our experiments might be largely explained the fact that it does not consider the presence of a defender. Moreover, our level-1 R2-B2 defender can also defend against black-box adversarial attacks from standard BO algorithms (Appendix E.6.2.2b)[11], which

[9] Compared to the work of (Moon et al., 2019), we use fewer iterations and a larger ϵ, which we think is more realistic as attacks with an excessively large no. of queries may be easily detected.

[10] The remaining 7 images are so easy to attack such that the attacks are already successful during the initial exploration phase of our level-1 R2-B2 defender.

[11] The BO attacker here only takes its perturbations as inputs and thus does not consider the defender.

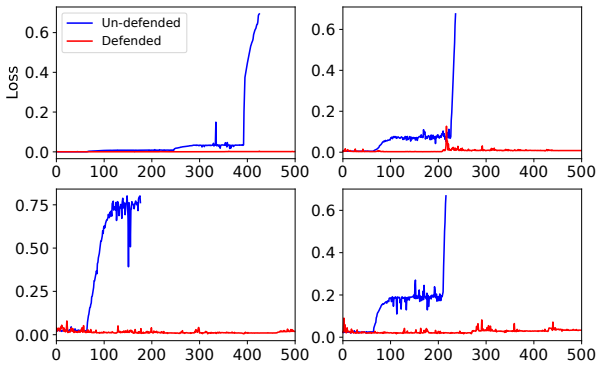


Figure 8.4: Loss incurred by Parsimonious with and without our level-1 R2-B2 defender on 4 randomly selected images that are successfully attacked by Parsimonious.

have become popular recently (Ru et al., 2020).

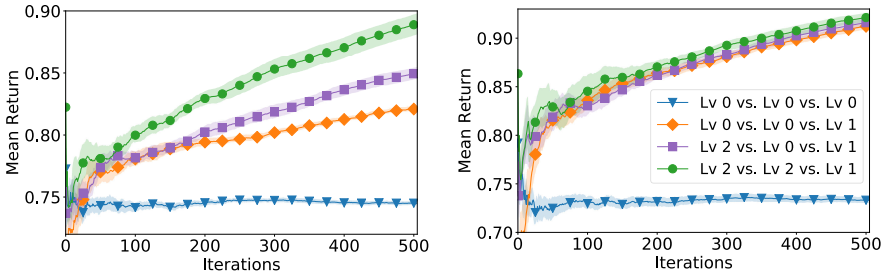
8.4.3 Multi-Agent Reinforcement Learning

We apply R2-B2 to policy search for MARL with *more than two* agents. Each action of an agent represents a particular set of policy parameters controlling the behavior of the agent in an environment. The payoff to each agent corresponding to a selected set of its policy parameters (i.e., action) is its mean return (i.e., cumulative reward) from the execution of all the agents' selected policies across 5 independent episodes. Since the agents interact in the environment, the payoff function of each agent depends on the policies (actions) selected by all agents. We use the predator-prey game from the widely used multi-agent particle environment in (Lowe et al., 2017). This 3-agent game (see Fig. E.11 in Appendix E.6.3) contains two predators who are trying to catch a prey. The prey is rewarded for being far from the predators and penalized for stepping outside the boundary. The two predators have identical payoff functions and are rewarded for being close to the prey (if the prey stays within the boundary). So, the predator-prey game represents a *general-sum game*. All agents use random search[12] and R2-B2

[12]All agents use only random search for level-0 reasoning due to high dimensions, as explained in Appendix E.6.3.

for level-0 and level-$k \geq 1$ reasoning, respectively.

Fig. 8.5 shows results of the (scaled) mean return of the agents averaged over 10 initializations of 5 randomly selected actions with observed payoffs. It can be observed from Fig. 8.5b that when the prey reasons at level 1 and both predators reason at level 0 (orange curve), its mean return is much higher than when reasoning at level 0 (blue curve); this results from the prey's ability to learn to stay within the boundary. Specifically, there exist some "dominated actions" in this game, namely, those causing the prey to step beyond the boundary. Regardless of the predators' policies, such dominated actions never give large returns to the prey and are thus likely to yield small values of GP-UCB for any actions (policies) selected by the predators. So, by reasoning at level 1 (i.e., by maximizing the expected value of GP-UCB), the prey is able to eliminate those dominated actions and thus learn to stay within the boundary. From Fig. 8.5a, the mean return of the predators is also improved (orange curve) because the prey's ability to stay within the boundary allows the predators to improve their rewards by being close to the prey despite using random search for level-0 reasoning. In contrast, when the prey reasons at level 0, the predators rarely get rewarded (blue curve) since the prey repeatedly steps beyond the boundary. On the other hand, when predator 1 reasons at level 2 (purple curve), the mean return of the predators is further increased since predator 1 is now able to learn to actively move close to the prey instead of moving around using random search for level-0 reasoning (orange curve). When both predators reason at level 2 (green curve), their mean return is improved even further. In both of these scenarios, the mean return of the prey stays close to that associated with the orange curve: Although the predators are able to actively approach the prey, this also further helps to prevent the prey from moving beyond the boundary, which compensates for the loss in its mean return due to the more strategic predators.



(a) predators (b) prey

Figure 8.5: Mean return of predators and prey in predator-prey game where the legend in (b) represents the levels of reasoning of predator 1 vs. predator 2 vs. prey.

8.5 Conclusion

This chapter describes the first BO algorithm called R2-B2 that is endowed with the capability of recursive reasoning to model the reasoning process in the interactions between boundedly rational[1], self-interested agents with unknown, complex, and expensive-to-evaluate payoff functions in repeated games. We prove that by reasoning at level $k \geq 2$ and one level higher than the other agents, our R2-B2 agent can achieve faster asymptotic convergence to no regret than that without utilizing recursive reasoning. We empirically demonstrate the competitive performance and generality of R2-B2 through extensive experiments using synthetic games, adversarial ML, and MARL.

Chapter 9

Conclusion

9.1 Summary

This thesis has presented five works that improve the sample efficiency of BO for AutoML or extend the applicability of BO to other important ML applications. As illustrated in Fig. 1.1, these five works can be categorized according to whether they focus on AutoML or other ML applications, and by whether they are developed for the single-agent or multi-agent setting. Each of the five new BO algorithms presented in this thesis is equipped with a theoretical convergence guarantee and has been shown to perform effectively in real-world experiments.

Chapter 4. Many ML models require an iterative training process such as stochastic gradient descent. When using BO for hyperparameter tuning of ML models, the training of some ML models that end up under-performing may be early-stopped to save resource and improve the epoch efficiency. In Chapter 4, we have introduced the *BO with Bayesian optimal stopping* (BO-BOS) algorithm, which incorporates BOS into BO in a principled manner, equipping BO with the ability to early-stop the training of unpromising hyperparameters while maintaining its trademark asymptotic no-regret property.

Chapter 5. The typical federated learning (FL) setting uses first-order

(gradient-based) optimization techniques. However, some ML tasks in the FL setting, such as hyperparameter tuning, lack access to gradients and hence require zeroth-order optimization methods such as BO. In Chapter 5, we have introduced the *federated Thompson sampling* (FTS) algorithm, which modifies BO for the FL setting. FTS overcomes a number of major challenges faced by federated BO: FTS uses random Fourier features approximation to avoid the sharing of raw data, employs Thompson sampling to improve its communication efficiency, and ensures its robustness against heterogeneous agents by offering a robust convergence guarantee.

Chapter 6. Although the above-mentioned FTS algorithm is able to avoid sharing the raw data, it is not equipped with a rigorous privacy guarantee, which is an important consideration in FL. In Chapter 6, we have integrated user-level *differential privacy* (DP) into FTS by following a general DP framework. In addition, we have leveraged the ability of the general DP framework to handle different parameter vectors, as well as the technique of local modeling for BO, to further improve the utility of our algorithm through *distributed exploration* (DE). The resulting *differentially private FTS with DE* (DP-FTS-DE) algorithm is amenable to a number of theoretical insights on the *privacy-utility trade-off*, and achieves competitive utilities with strong privacy guarantees in real-world experiments.

Chapter 7. When using BO to optimize the hyperparameters of an ML model using a dataset, we often have access to previous completed hyperparameter optimization tasks using other potentially related datasets. This prompts the question as to whether we can leverage these previous experiences to accelerate the current BO task through *meta-learning*, while ensuring robustness against harmful dissimilar tasks. In Chapter 7, we have introduced a scalable, principled and robust meta-BO algorithm: *robust meta-GP-UCB* (RM-GP-UCB), which uses a weighted combination of the acquisition functions from different tasks for

query selection. RM-GP-UCB is asymptotically no-regret even when all previous tasks are dissimilar to the current task, and is amenable to a principled method to learn the weights assigned to each previous task through regret minimization via online learning. RM-GP-UCB performs competitively and consistently across a wide range of real-world experiments.

Chapter 8. Some ML tasks such as adversarial ML can be modeled as repeated games between boundedly rational, self-interested agents with unknown, complex, and costly-to-evaluate payoff functions. In Chapter 8, we have introduced the *Recursive Reasoning-Based BO* (R2-B2) algorithm, which is a recursive reasoning formalism of BO that provides efficient strategies for players in this type of game. Under certain conditions, using R2-B2 to reason at one level higher than the other agents leads to improved convergence compared with not using recursive reasoning. R2-B2 is also shown to be practically effective in adversarial ML and multi-agent reinforcement learning experiments.

9.2 Future Outlook

9.2.1 Towards More Practical Collaborative/Federated BO

The ever-increasing computational capability of edge devices, coupled with increasing awareness of and regulations on data privacy, has prompted a surging interest in extending ML to the collaborative/federated setting (Kairouz et al., 2019; Li et al., 2019b). Many AutoML tasks, such as hyperparameter tuning and neural architecture search, require zeroth-order optimization methods such as BO. In this regard, we have extended BO to the federated setting (Chapters 5 and 6), hence allowing multiple agents (e.g., edge devices, hospitals, etc.) to collaborate in zeroth-order optimization tasks without violating the privacy of individual agents. However, to further promote the practical application of our FTS and DP-FTS-DE algorithms (Chapters 5 and 6) in the federated BO setting, a number

of other important issues remain to be addressed. For example, *fairness* in ML has been receiving increasing attention in recent years, which makes fairness among agents an important consideration in collaborative/federated ML (Li et al., 2020c); providing *incentives* for agents to participate is critical for initiating and sustaining the collaboration/federation (Sim et al., 2020); *robustness* against malicious agents is required to ensure the reliability of collaborative/federated ML algorithms (Blanchard et al., 2017). Therefore, integrating these important considerations into our algorithms for the FBO setting represents a promising direction for future research. Moreover, in addition to BO, another interesting potential direction is to extend other classes of algorithms for *sequential decision-making under uncertainty* (e.g., multi-armed bandit, reinforcement learning, active learning, etc.) to the collaborative/federated setting, to allow a wider range of tasks to be performed in this setting.

9.2.2 More Applications of BO beyond ML

Although BO has gained most of its visibility through its applications in AutoML, BO is a *general* black-box optimization method with impressive sample efficiency and is hence suitable for a wide range of applications beyond ML. Firstly, as we have discussed in Section 1.1 (second last paragraph), our algorithms in this thesis can also find other interesting applications beyond ML such as precision agriculture, patient selection for medical tests, etc. Thus, it would be interesting to apply our algorithms in this thesis to these scenarios and other potential applications. Secondly, some real-world experimental design problems are amenable to the application of BO yet require customized modifications to standard BO algorithms to adapt to their problem structures. For example, applying BO to molecule design requires a customized kernel to measure the similarity between molecules (Korovina et al., 2020). Therefore, it is promising to explore novel applications of BO to more real-world experimental design

problems to improve the efficiency of scientific discovery, where it would be interesting to investigate the required modifications to existing BO algorithms as well as their implications on the existing theoretical guarantees.

Bibliography

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318.

Anguita, D., Ghio, A., Oneto, L., Parra, X., and Reyes-Ortiz, J. L. (2013). A public domain dataset for human activity recognition using smartphones. In *Proc. ESANN*.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 17(6):26–38.

Baker, B., Gupta, O., Raskar, R., and Naik, N. (2017). Accelerating neural architecture search using performance prediction. In *Proc. NIPS Workshop on Meta-Learning*.

Bardenet, R., Brendel, M., Kégl, B., and Sebag, M. (2013). Collaborative hyperparameter tuning. In *Proc. ICML*, pages 199–207.

Beyer, H.-G. and Sendhoff, B. (2007). Robust optimization–a comprehensive survey. *Computer methods in applied mechanics and engineering*, 196(33-34):3190–3218.

Bittau, A., Erlingsson, Ú., Maniatis, P., Mironov, I., Raghunathan, A., Lie, D.,

Rudominer, M., Kode, U., Tinnes, J., and Seefeld, B. (2017). Prochlo: Strong privacy for analytics in the crowd. In *Proc. SOSP*, pages 441–459.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. In *Proc. NeurIPS*, pages 118–128.

Bogunovic, I., Scarlett, J., Jegelka, S., and Cevher, V. (2018). Adversarially robust optimization with Gaussian processes. In *Proc. NeurIPS*, pages 5760–5770.

Brochu, E., Cora, V. M., and de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. arXiv:1012.2599.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI Gym. arXiv:1606.01540.

Brockwell, A. E. and Kadane, J. B. (2003). A gridding method for Bayesian sequential decision problems. *J. Computational and Graphical Statistics*, 12(3):566–584.

Bubeck, S. (2011). Introduction to online optimization. Lecture notes.

Burger, B., Maffettone, P. M., Gusev, V. V., Aitchison, C. M., Bai, Y., Wang, X., Li, X., Alston, B. M., Li, B., Clowes, R., et al. (2020). A mobile robotic chemist. *Nature*, 583(7815):237–241.

Camerer, C. F., Ho, T.-H., and Chong, J.-K. (2004). A cognitive hierarchy model of games. *Quarterly J. Economics*, 119(3):861–898.

Chang, H., Shejwalkar, V., Shokri, R., and Houmansadr, A. (2019). Cronus: Robust and heterogeneous collaborative learning with black-box knowledge transfer. arXiv:1912.11279.

Chapelle, O. and Li, L. (2011). An empirical evaluation of Thompson sampling. In *Proc. NeurIPS*, pages 2249–2257.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proc. KDD*, pages 785–794.

Cheu, A., Smith, A., Ullman, J., Zeber, D., and Zhilyaev, M. (2019). Distributed differential privacy via shuffling. In *Proc. EUROCRYPT*, pages 375–403. Springer.

Chowdhury, S. R. and Gopalan, A. (2017). On kernelized multi-armed bandits. In *Proc. ICML*, pages 844–853.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. In *Proc. IJCNN*, pages 2921–2926. IEEE.

Contal, E., Buffoni, D., Robicquet, A., and Vayatis, N. (2013). Parallel Gaussian process optimization with upper confidence bound and pure exploration. In *Proc. ECML/PKDD*, pages 225–240. Springer.

Dai, Z., Chen, Y., Low, K. H., Jaillet, P., and Ho, T.-H. (2020a). R2-b2: Recursive reasoning-based Bayesian optimization for no-regret learning in games. In *Proc. ICML*.

Dai, Z., Low, K. H., and Jaillet, P. (2020b). Federated Bayesian optimization via Thompson sampling. In *Proc. NeurIPS*.

Dai, Z., Yu, H., Low, B. K. H., and Jaillet, P. (2019). Bayesian optimization meets Bayesian optimal stopping. In *Proc. ICML*, pages 1496–1506.

Davis, G. A. and Cairns, R. D. (2012). Good timing: The economics of optimal stopping. *Journal of Economic Dynamics and Control*, 36(2):255–265.

Daxberger, E. A. and Low, B. K. H. (2017). Distributed batch Gaussian process optimization. In *Proc. ICML*, pages 951–960.

Desautels, T., Krause, A., and Burdick, J. W. (2014). Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:3873–3923.

Dheeru, D. and Karra Taniskidou, E. (2017). UCI machine learning repository.

Domhan, T., Springenberg, J. T., and Hutter, F. (2015). Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proc. IJCAI*, pages 3460–3468.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. (2016). RL^2: Fast reinforcement learning via slow reinforcement learning. arXiv:1611.02779.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. (2006a). Our data, ourselves: Privacy via distributed noise generation. In *Proc. EUROCRYPT*, pages 486–503. Springer.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006b). Calibrating noise to sensitivity in private data analysis. In *Proc. TCC*, pages 265–284. Springer.

Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. (2019). Scalable global optimization via local Bayesian optimization. In *Proc. NeurIPS*.

Falkner, S., Klein, A., and Hutter, F. (2018). BOHB: Robust and efficient hyperparameter optimization at scale. In *Proc. ICML*, pages 1436–1445.

Ferguson, T. S. (2006). *Optimal stopping and applications*.

Feurer, M., Letham, B., and Bakshy, E. (2018). Scalable meta-learning for Bayesian optimization using ranking-weighted Gaussian process ensembles. In *Proc. ICML Workshop on Automatic Machine Learning*.

Feurer, M., Springenberg, J. T., and Hutter, F. (2015). Initializing Bayesian hyperparameter optimization via meta-learning. In *Proc. AAAI*, pages 1128–1135.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*, pages 1126–1135.

Frazier, P. I. (2018). A tutorial on Bayesian optimization. arXiv:1807.02811.

Frazier, P. I. and Wang, J. (2016). Bayesian optimization for materials design. In *Information Science for Materials Discovery and Design*, pages 45–75. Springer.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Ann. Statistics*, 29(5):1189–1232.

Garcia-Barcos, J. and Martinez-Cantin, R. (2019). Fully distributed bayesian optimization with stochastic policies. In *Proc. IJCAI*.

Gigerenzer, G. and Selten, R. (2002). *Bounded Rationality*. MIT Press.

Gmytrasiewicz, P. J. and Doshi, P. (2005). A framework for sequential planning in multi-agent settings. *J. Artif. Intell. Res.*, 24:49–79.

Goldman, A. I. (2012). Theory of mind. In Margolis, E., Samuels, R., and Stich, S. P., editors, *The Oxford Handbook of Philosophy of Cognitive Science*. Oxford Univ. Press.

Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., and Sculley, D. (2017). Google Vizier: A service for black-box optimization. In *Proc. ACM SIGKDD*, pages 1487–1495.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *Proc. ICLR*.

Hall, M. A. (2000). Correlation-based feature selection of discrete and numeric class machine learning. In *Proc. ICML*, pages 359–366.

Hall, M. A. and Smith, L. A. (1999). Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper. In *Proc. FLAIRS conference*, pages 235–239.

Hennig, P. and Schuler, C. J. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. (2014). Predictive entropy search for efficient global optimization of black-box functions. In *Proc. NeurIPS*, pages 918–926.

Hernández-Lobato, J. M., Requeima, J., Pyzer-Knapp, E. O., and Aspuru-Guzik, A. (2017). Parallel and distributed Thompson sampling for large-scale accelerated exploration of chemical space. In *Proc. ICML*.

Hoang, T. N. and Low, K. H. (2013). Interactive POMDP Lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. In *Proc. IJCAI*.

Hoffman, M., Brochu, E., de Freitas, N., et al. (2011). Portfolio allocation for bayesian optimization. In *Proc. UAI*, pages 327–336. Citeseer.

Hutter, F., Kotthoff, L., and Vanschoren, J. (2019). *Automated machine learning: methods, systems, challenges*. Springer Nature.

Ji, Z., Lipton, Z. C., and Elkan, C. (2014). Differential privacy and machine learning: a survey and review. arXiv:1412.7584.

Jiang, F., Jack Lee, J., and Müller, P. (2013). A Bayesian decision-theoretic sequential response-adaptive randomization design. *Statistics in Medicine*, 32(12):1975–1994.

Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. (2019). Advances and open problems in federated learning. arXiv:1912.04977.

Kandasamy, K., Dasarathy, G., Oliva, J. B., Schneider, J., and Póczos, B. (2016). Gaussian process bandit optimisation with multi-fidelity evaluations. In *Proc. NIPS*, pages 992–1000.

Kandasamy, K., Dasarathy, G., Schneider, J., and Póczos, B. (2017). Multi-fidelity Bayesian optimisation with continuous approximations. In *Proc. ICML*, pages 1799–1808.

Kandasamy, K., Krishnamurthy, A., Schneider, J., and Póczos, B. (2018). Parallelised Bayesian optimisation via Thompson sampling. In *Proc. AISTATS*, pages 133–142.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. (2011). What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826.

Kharkovskii, D., Dai, Z., and Low, B. K. H. (2020). Private outsourced Bayesian optimization. In *Proc. ICML*.

Kim, J. and Choi, S. (2019). On local optimizers of acquisition functions in Bayesian optimization. arXiv:1901.08350.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational Bayes. In *Proc. ICLR*.

Klein, A., Falkner, S., Springenberg, J. T., and Hutter, F. (2017). Learning curve prediction with Bayesian neural networks. In *Proc. ICLR*.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *Proc. ICML*, volume 2. Lille.

Korovina, K., Xu, S., Kandasamy, K., Neiswanger, W., Poczos, B., Schneider, J., and Xing, E. (2020). Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *Proc. AISTATS*, pages 3393–3403. PMLR.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto.

Kusner, M., Gardner, J., Garnett, R., and Weinberger, K. (2015). Differentially private Bayesian optimization. In *Proc. ICML*, pages 918–927.

Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Lattimore, T. and Szepesvári, C. (2020). *Bandit Algorithms*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lewis, R. J. and Berry, D. A. (1994). Group sequential clinical trials: A classical evaluation of Bayesian decision-theoretic designs. *J. American Statistical Association*, 89(428):1528–1534.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. (2017). Hyperband: A novel bandit-based approach to hyperparameter optimization. *JMLR*, 18(1):6765–6816.

Li, Q., Wen, Z., and He, B. (2019a). Federated learning systems: Vision, hype and reality for data privacy and protection. arXiv:1907.09693.

Li, Q., Wen, Z., and He, B. (2020a). Practical federated gradient boosting decision trees. In *Proc. AAAI*, pages 4642–4649.

Li, Q., Wu, Z., Wen, Z., and He, B. (2020b). Privacy-preserving gradient boosting decision trees. In *Proc. AAAI*, volume 34, pages 784–791.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. (2019b). Federated learning: Challenges, methods, and future directions. arXiv:1908.07873.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2018). Federated optimization in heterogeneous networks. arXiv:1812.06127.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. (2020c). Fair resource allocation in federated learning. In *Proc. ICLR*.

Li, X., Huang, K., Yang, W., Wang, S., and Zhang, Z. (2020d). On the convergence of FedAvg on non-iid data. In *Proc. ICLR*.

Lizotte, D. J., Wang, T., Bowling, M. H., and Schuurmans, D. (2007). Automatic gait optimization with Gaussian process regression. In *IJCAI*, pages 944–949.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proc. NeurIPS*, pages 6379–6390.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. In *Proc. ICLR*.

Martinez-Cantin, R., de Freitas, N., Doucet, A., and Castellanos, J. A. (2007). Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and systems*, volume 3, pages 321–328.

McMahan, H. B., Andrew, G., Erlingsson, U., Chien, S., Mironov, I., Papernot, N., and Kairouz, P. (2018a). A general approach to adding differential privacy to iterative training procedures. In *Proc. NeurIPS Workshop on Privacy Preserving Machine Learning*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., et al. (2017). Communication-efficient learning of deep networks from decentralized data. In *Proc. AISTATS*.

McMahan, H. B., Ramage, D., Talwar, K., and Zhang, L. (2018b). Learning differentially private recurrent language models. In *Proc. ICLR*.

Meng, D. and Chen, H. (2017). MagNet: A two-pronged defense against adversarial examples. In *Proc. CCS*, pages 135–147.

Moon, S., An, G., and Song, H. O. (2019). Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *Proc. ICML*.

Müller, P., Berry, D. A., Grieve, A. P., Smith, M., and Krams, M. (2007). Simulation-based sequential Bayesian design. *J. Statistical Planning and Inference*, 137(10):3140–3150.

Mutny, M. and Krause, A. (2018). Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. In *Proc. NeurIPS*, pages 9005–9016.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *Proc. NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Nguyen, D. T., Gupta, S., Rana, S., and Venkatesh, S. (2018). A privacy preserving Bayesian optimization with high efficiency. In *Proc. PAKDD*, pages 543–555. Springer.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge Univ. Press.

Pang, K., Dong, M., Wu, Y., and Hospedales, T. (2018). Meta-learning transferable active learning policies by deep reinforcement learning. arXiv:1806.04798.

Perrone, V., Jenatton, R., Seeger, M. W., and Archambeau, C. (2018). Scalable hyperparameter transfer learning. In *Proc. NIPS*, pages 6845–6855.

Perrone, V., Shen, H., Seeger, M. W., Archambeau, C., and Jenatton, R. (2019). Learning search spaces for bayesian optimization: Another view of hyperparameter transfer learning. In *Proc. NeurIPS*, pages 12771–12781.

Picheny, V., Binois, M., and Habbal, A. (2019). A Bayesian optimization approach to find Nash equilibria. *Journal of Global Optimization*, 73(1):171–192.

Poloczek, M., Wang, J., and Frazier, P. (2017). Multi-information source optimization. In *Proc. NeurIPS*, pages 4288–4298.

Poloczek, M., Wang, J., and Frazier, P. I. (2016). Warm starting Bayesian optimization. In *Proc. WSC*, pages 770–781.

Powell, W. B. and Ryzhov, I. O. (2012). *Optimal learning*. John Wiley & Sons.

Pynadath, D. V. and Marsella, S. C. (2005). PsychSim: Modeling theory of mind with decision-theoretic agents. In *Proc. IJCAI*, pages 1181–1186.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Proc. NeurIPS*, pages 1177–1184.

Rahman, S. A., Merck, C., Huang, Y., and Kleinberg, S. (2015). Unintrusive eating recognition using Google glass. In *Proc. PervasiveHealth*, pages 108–111. IEEE.

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Reyes-Ortiz, J. L., Ghio, A., Parra, X., Anguita, D., Cabestany, J., and Catala, A. (2013). Human activity and motion disorder recognition: Towards smarter interactive cognitive environments. In *Proc. ESANN*.

Ru, B., Cobb, A., Blaas, A., and Gal, Y. (2020). BayesOpt adversarial attack. In *Proc. ICLR*.

Russo, D. and Van Roy, B. (2014). Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243.

Russo, D., Van Roy, B., Kazerouni, A., Osband, I., and Wen, Z. (2017). A tutorial on Thompson sampling. arXiv:1707.02038.

Salinas, D., Shen, H., and Perrone, V. (2020). A quantile-based approach for hyperparameter transfer learning. In *Proc. ICML*, pages 8438–8448. PMLR.

Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *Proc. ICLR*.

Schilling, N., Wistuba, M., and Schmidt-Thieme, L. (2016). Scalable hyperparameter optimization with products of Gaussian process experts. In *Proc. ECML/PKDD*, pages 33–48.

Sessa, P. G., Bogunovic, I., Kamgarpour, M., and Krause, A. (2019). No-regret learning in unknown games with correlated payoffs. In *Proc. NeurIPS*.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Shi, E., Chan, T. H., Rieffel, E., Chow, R., and Song, D. (2011). Privacy-preserving aggregation of time-series data. In *Proc. NDSS*, volume 2, pages 1–17. Citeseer.

Shilton, A., Gupta, S., Rana, S., and Venkatesh, S. (2017). Regret bounds for transfer learning in Bayesian optimisation. In *Proc. AISTATS*, pages 1–9.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Sim, R. H. L., Zhang, Y., Chan, M. C., and Low, B. K. H. (2020). Collaborative machine learning with incentive-aware model rewards. In *Proc. ICML*, pages 8927–8936. PMLR.

Smith, J. W., Everhart, J. E., Dickson, W. C., Knowler, W. C., and Johannes, R. S. (1988). Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proc Annu. Symp. Comput. Appl. Med. Care*, pages 261–265.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017). Federated multi-task learning. In *Proc. NeurIPS*, pages 4424–4434.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Proc. NeurIPS*, pages 2951–2959.

Soleimani, E. and Nazerfard, E. (2019). Cross-subject transfer learning in

human activity recognition systems using generative adversarial networks. arxiv:1903.12489.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pages 1015–1022.

Swersky, K., Snoek, J., and Adams, R. P. (2013). Multi-task Bayesian optimization. In *Proc. NeurIPS*, pages 2004–2012.

Swersky, K., Snoek, J., and Adams, R. P. (2014). Freeze-thaw Bayesian optimization. arXiv:1406.3896.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2014). Intriguing properties of neural networks. In *Proc. ICLR*.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Todorov, E., Erez, T., and Tassa, Y. (2012). MuJoCo: A physics engine for model-based control. In *Proc. IEEE/RSJ IROS*, pages 5026–5033.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2018). Ensemble adversarial training: Attacks and defenses. In *Proc. ICLR*.

Tu, C.-C., Ting, P., Chen, P.-Y., Liu, S., Zhang, H., Yi, J., Hsieh, C.-J., and Cheng, S.-M. (2019). AutoZOOM: Autoencoder-based zeroth order optimization method for attacking black-box neural networks. In *Proc. AAAI*, pages 742–749.

Vanschoren, J. (2018). Meta-learning: A survey. arXiv:1810.03548.

Volpp, M., Froehlich, L., Fischer, K., Doerr, A., Falkner, S., Hutter, F., and Daniel, C. (2020). Meta-learning acquisition functions for transfer learning in Bayesian optimization. In *Proc. ICLR*.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. arXiv:1611.05763.

Wang, Y.-X., Balle, B., and Kasiviswanathan, S. P. (2019). Subsampled Rényi differential privacy and analytical moments accountant. In *Proc. AISTATS*, pages 1226–1235. PMLR.

Wang, Z. and Jegelka, S. (2017). Max-value entropy search for efficient Bayesian optimization. In *Proc. ICML*.

Wang, Z., Kim, B., and Kaelbling, L. P. (2018). Regret bounds for meta Bayesian optimization with an unknown Gaussian process prior. In *Proc. NeurIPS*, pages 10477–10488.

Warner, S. L. (1965). Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69.

Wathen, J. K. and Thall, P. F. (2008). Bayesian adaptive model selection for optimizing group sequential clinical trials. *Statistics in Medicine*, 27(27):5586–5604.

Wen, Y., Yang, Y., Luo, R., Wang, J., and Pan, W. (2019). Probabilistic recursive reasoning for multi-agent reinforcement learning. In *Proc. ICLR*.

Wilson, A., Fern, A., and Tadepalli, P. (2014). Using trajectory data to improve Bayesian optimization for reinforcement learning. *Journal of Machine Learning Research*, 15(1):253–282.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015a). Learning hyperparameter optimization initializations. In *Proc. DSAA*, pages 1–10. IEEE.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2015b). Sequential model-free hyperparameter tuning. In *Proc. ICDM*, pages 1033–1038.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2016). Two-stage transfer surrogate model for automatic hyperparameter optimization. In *Proc. ECML/PKDD*, pages 199–214.

Wistuba, M., Schilling, N., and Schmidt-Thieme, L. (2018). Scalable Gaussian process-based transfer surrogates for hyperparameter optimization. *Machine Learning*, 107(1):43–78.

Wu, J. and Frazier, P. I. (2018). Continuous-fidelity bayesian optimization with knowledge gradient.

Wu, J., Toscano-Palmerin, S., Frazier, P. I., and Wilson, A. G. (2020). Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *Proc. UAI*, pages 788–798. PMLR.

Xu, Z., van Hasselt, H. P., and Silver, D. (2018). Meta-gradient reinforcement learning. In *Proc. NeurIPS*, pages 2396–2407.

Xue, Y., Liao, X., Carin, L., and Krishnapuram, B. (2007). Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63.

Yogatama, D. and Mann, G. (2014). Efficient transfer learning method for automatic hyperparameter tuning. In *Proc. AISTATS*, pages 1077–1085.

Yu, S., Farooq, F., Van Esbroeck, A., Fung, G., Anand, V., and Krishnapuram, B. (2015). Predicting readmission risk with institution-specific prediction models. *Artificial intelligence in medicine*, 65(2):89–96.

Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks

and defenses for deep learning. *IEEE Trans. Neural Netw. Learning Syst.*, 30(9):2805–2824.

Zhang, Y., Dai, Z., and Low, B. K. H. (2020). Bayesian optimization with binary auxiliary information. In *Proc. UAI*, pages 1222–1232. PMLR.

Zhang, Y., Hoang, T. N., Low, B. K. H., and Kankanhalli, M. (2017). Information-based multi-fidelity Bayesian optimization. In *Proc. NeurIPS Workshop on Bayesian Optimization*.

Zhou, X. and Tan, J. (2020). Local differential privacy for Bayesian optimization. arXiv:2010.06709.

Appendix A

Appendix for Chapter 4

A.1 Approximate Backward Induction for Bayesian Optimal Stopping

In this section, we will present a commonly-used approximate backward induction algorithm for solving the BOS problem. The algorithm uses summary statistics to compactly represent the posterior belief $\mathbb{P}(\theta_t|\mathbf{y}_{t,n})$ which is computed from the prior belief $\mathbb{P}(\theta_t)$ and the noisy outputs $\mathbf{y}_{t,n}$ observed up till epoch n in iteration t.

In the approximate backward induction algorithm of (Müller et al., 2007), the entire space of summary statistics is firstly partitioned into a number of discrete intervals in each epoch, which results in a two-dimensional domain with one axis being the number of epochs and the other axis representing the discretized intervals of the summary statistic (i.e., assuming the summary statistic is one-dimensional). In the beginning, a number of sample paths are generated through *forward simulation*: Firstly, a large number of samples are drawn from the prior belief $\mathbb{P}(\theta_t)$. Then, for each sample drawn from $\mathbb{P}(\theta_t)$, an entire sample path is generated from epochs 1 to N through repeated sampling. In this manner, each sample path leads to a curve in the 2-D domain and fully defines N posterior

beliefs with one in each epoch. Starting from the last epoch N, for each interval, the expected loss of a terminal decision d_1 or d_2 is evaluated for every sample path ending in this interval (since each such sample path ends with a particular posterior belief in epoch N), and their empirical average is used to approximate the expected loss of the particular terminal decision for this interval. The minimum of the expected losses among the two terminal decisions is the expected loss for this particular interval, which is equivalent to (4.1) except that decision d_0 is not available in the last epoch N.

Next, the algorithm proceeds backwards from epoch $n = N - 1$ all the way to epoch $n = 1$. In each epoch n, the expected loss of each terminal decision is evaluated in the same way as that in the last epoch N, as described above. To evaluate the expected loss of the continuation decision for an interval, for each sample path passing through this interval, the expected loss for the interval that it passes through in the next epoch $n + 1$ is recorded and an average of all the recorded expected losses in the next epoch $n + 1$ is summed with the cost c_{d_0} of observing the noisy output $y_{t,n+1}$ to yield the expected loss of the continuation decision d_0 for this particular interval; this is equivalent to approximating the $\mathbb{E}_{y_{t,n+1}|\mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n+1})] + c_{d_0}$ term in (4.1) via Monte Carlo sampling of the posterior belief $\mathbb{P}(y_{t,n+1}|\mathbf{y}_{t,n})$. Following (4.1), the minimum of the expected losses among all terminal and continuation decisions is the expected loss for this particular interval and the corresponding decision is recorded as the optimal decision when the summary statistic falls into this interval. Then, the algorithm continues backwards until epoch $n = 1$ is reached. After the algorithm has finished running, the optimal decision computed in every pair of epoch and interval will form the optimal decision rules which serve as the output of the approximate backward induction algorithm.

A.2 Approximate Backward Induction Algorithm for Solving BOS Problem in BO-BOS

In this section, we will describe the approximate backward induction algorithm for solving the BOS problem (line 5) in each iteration of BO-BOS (Algorithm 4.1), which is adapted from the algorithm introduced in Appendix A.1.

To account for Assumption 4.1b in the approximate backward induction algorithm, we adopt the kernel k introduced in (Swersky et al., 2014) to incorporate the inductive bias that the learning curve (in the form of validation error) of the ML model is approximately exponentially decreasing in the number of training epochs, which can be expressed as

$$k(n, n') \triangleq \int_0^\infty \exp(-\lambda n) \exp(-\lambda n') \, \phi(\lambda) \, \mathrm{d}\lambda = \frac{\beta^\alpha}{(n + n' + \beta)^\alpha} \tag{A.1}$$

for all epochs $n, n' = 1, \ldots, N$ where ϕ is a probability measure over λ that is chosen to be a Gamma prior with parameters α and β. The above kernel (A.1) is used to fit a GP model to the validation errors $\mathbf{1} - \mathbf{y}_{t,N_0}$ of the ML model trained using $\mathbf{x}_t$ for a fixed number N_0 of initial epochs (e.g., $N_0 = 8$ in all our experiments when $N = 50$), specifically, by computing the values of parameters α and β in (A.1) via Bayesian update (i.e., assuming that the validation errors follow the Gamma conjugate prior with respect to an exponential likelihood). Samples are then drawn from the resulting GP posterior belief for forward simulation of sample paths from epochs $N_0 + 1$ to N, which are used to estimate the $\mathbb{P}(\theta_t|\mathbf{y}_{t,n})$ and $\mathbb{P}(y_{t,n+1}|\mathbf{y}_{t,n})$ terms necessary for approximate backward induction. Fig. A.1 plots some of such sample paths and demonstrates that the GP kernel in (A.1) can characterize a monotonic learning curve (Assumption 4.1b) well.

Following the practices in related applications of BOS (Brockwell and Kadane, 2003; Jiang et al., 2013; Müller et al., 2007), the average validation

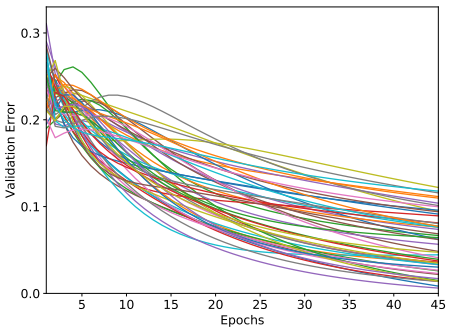


Figure A.1: Forward simulation of some sample paths drawn from a GP posterior belief based on the kernel in (A.1).

error (or, equivalently, average validation accuracy) over epochs 1 to n is used as the summary statistics. Firstly, the entire space of summary statistics is partitioned into a number of discrete intervals in each epoch, which results in a two-dimensional domain with one axis being the number of epochs and the other axis representing the discretized intervals of the summary statistic (i.e., average validation error). Next, a forward simulation of a large number (i.e., $100,000$ in all our experiments) of sample paths is performed using the GP kernel in (A.1), as described above. Each sample path corresponds to a curve in the 2-D domain. Starting from the last epoch N, for each interval, we consider all sample paths ending in this interval and use the proportion of such sample paths with a validation accuracy (from model training for N epochs) larger than the currently found maximum (offset by a noise correction term) to estimate the posterior probability $\mathbb{P}(\theta_t = \theta_{t,2}|\mathbf{y}_{t,N}) = \mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,N})$, which is in turn used to evaluate the expected losses of the terminal decisions d_1 and d_2 for this interval.[1] The minimum of the expected losses among the two terminal decisions is the expected loss for this particular interval.

Next, the algorithm proceeds backwards from epoch $n = N - 1$ all the way to epoch $n = N_0 + 1$. In each epoch n, the expected loss of each terminal

[1]In contrast to the approximate backward induction algorithm of (Müller et al., 2007) (Appendix A.1), we employ a computationally cheaper way to approximate the expected losses of the terminal decisions for an interval.

decision is evaluated in the same way as that in the last epoch N, as described above. The expected loss of the continuation decision d_0 is evaluated in the same way as that in Appendix A.1: For each sample path passing through an interval in epoch n, the expected loss for the interval that it passes through in the next epoch $n+1$ is recorded and an average of all the recorded expected losses in the next epoch $n+1$ is summed with the cost c_{d_0} of observing the validation accuracy $y_{t,n+1}$ to yield the expected loss of the continuation decision d_0 for this particular interval. Note that this step is equivalent to approximating the $\mathbb{E}_{y_{t,n+1}|\mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n})]$ term in (4.1) via Monte Carlo sampling of the posterior belief $\mathbb{P}(y_{t,n+1}|\mathbf{y}_{t,n})$. Following (4.1), the minimum of expected losses among all terminal and continuation decisions is the expected loss for this particular interval and the corresponding decision is recorded as the optimal decision to be recommended when the summary statistic falls into this particular interval. Then, the algorithm continues backwards until epoch $n = N_0 + 1$ is reached. We present in Algorithm 1.1 the pseudocode for the above-mentioned approximate backward induction algorithm for ease of understanding.

After solving our BOS problem for early stopping in BO using the approximate backward induction algorithm described above, Bayes-optimal decision rules are obtained in every pair of epoch and interval. Fig. A.2 shows an example of optimal decision rules obtained from solving an instance of our BOS problem where the white, yellow, and red regions correspond to recommending optimal continuation decision d_0 and terminal decisions d_1 and d_2, respectively. In particular, after model training under $\mathbf{x}_t$ to yield the validation error $1 - y_{t,n}$ in epoch n, the summary statistic is updated to the average validation error over epochs 1 to n. The updated summary statistic falls into an interval with a corresponding optimal decision to be recommended. For example, Fig. A.2 shows that if the summary statistic falls into the yellow region in any epoch n, then the optimal terminal decision d_1 is recommended to early-stop model training under $\mathbf{x}_t$ (assuming

Algorithm 1.1 Approximate Backward Induction Algorithm for Solving BOS Problem in BO-BOS

Partition the domain of summary statistics into M discrete intervals
Train the ML model using $\mathbf{x}_t$ for N_0 epochs
Generate a large number of forward simulation samples using kernel (A.1)
Let $n = N$
for $m = 1, 2, \ldots, M$ **do**
    Find all sample paths ending in interval m at epoch n, denoted as $\mathcal{S}$
    Estimate $\mathbb{P}(\theta_t = \theta_{t,2}|\mathbf{y}_{t,n}) = \mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n})$ by the proportion of $\mathcal{S}$ that end up (after N epochs) having larger validation accuracy than $y^*_{t-1} - \xi_t$
    Calculate the expected losses of the terminal decisions d_1 and d_2 using (4.3)
    Use the minimum of these two expected losses as the expected loss of epoch n and interval m
for $n = N-1, N-2, \ldots, N_0+1$ **do**
    for $m = 1, 2, \ldots, M$ **do**
        Find all sample paths passing through interval m at epoch n, denoted as $\mathcal{S}$
        Estimate $\mathbb{P}(\theta_t = \theta_{t,2}|\mathbf{y}_{t,n}) = \mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n})$ by the proportion of $\mathcal{S}$ that end up (after N epochs) having larger validation accuracy than $y^*_{t-1} - \xi_t$
        Calculate the expected losses of the terminal decisions d_1 and d_2 using (4.3)
        $l_{d_0,n+1,m} = 0$
        for each sample path s in $\mathcal{S}$ **do**
            $l_{d_0,n+1,m} = l_{d_0,n+1,m}+$ the expected loss of the interval reached by s at epoch $n+1$
        $l_{d_0,n+1,m} = l_{d_0,n+1,m}/|\mathcal{S}|$
        Calculate the expected loss of the continuation decision d_0 as: $\mathbb{E}_{y_{t,n+1}|\mathbf{y}_{t,n}}[\rho_{t,n+1}(\mathbf{y}_{t,n})] + c_{d_0} = l_{d_0,n+1,m} + c_{d_0}$
        Use the minimum expected losses among d_1, d_2 and d_0 as the expected loss of epoch n and interval m (following (4.1)), and record the corresponding decision as the optimal decision

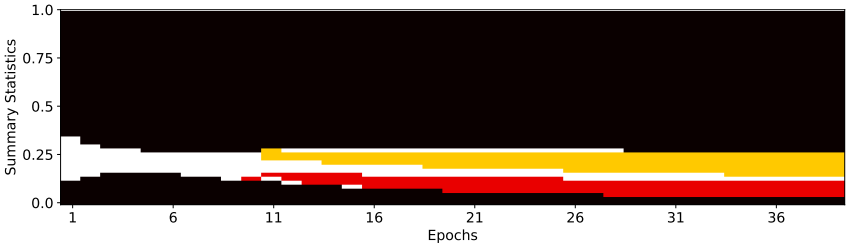


Figure A.2: An example of optimal decision rules obtained from solving an instance of our BOS problem: White, yellow, and red regions correspond to recommending optimal continuation decision d_0 and terminal decisions d_1 and d_2, respectively. The sample paths cannot reach the black regions due to the use of the GP kernel in (A.1) for characterizing a monotonic learning curve (Assumption 4.1b).

that C2 is satisfied). If the summary statistic falls into any other region, then model training continues under $\mathbf{x}_t$ for one more epoch and the above procedure is repeated in epoch $n+1$ until the last epoch $n = N$ is reached. This procedure, together with C2, constitutes lines 6 to 9 in Algorithm 4.1.

A.3 Proof of Theorems 4.1 and 4.2

In this section, we prove the theoretical results in this paper.

A.3.1 Regret Decomposition

In this work, it is natural and convenient to define the instantaneous regret at step t as $r_t = f(\mathbf{z}^*) - f_t^*$, in which $\mathbf{z}^*$ is the location of the global maximum: $\mathbf{z}^* = \text{argmax}_{\mathbf{z}} f(\mathbf{z})$, and f_t^* is the maximum observed function value from iterations 1 to t: $f_t^* = \max_{t'=1,\ldots,t} f(\mathbf{z}_{t'})$. Subsequently, the cumulative regret and simple regret after T iterations are defined as $R_T = \sum_{t=1}^{T} r_t$ and $S_T = \min_{t=1,\ldots,T} r_t$ respectively. As a result, as long as we can show that R_T grows sub-linearly in T, then we can conclude that the average regret $\frac{R_T}{T}$ asymptotically goes to 0; therefore, S_T vanishes asymptotically since it is upper-bounded by the average regret: $S_T \leq \frac{R_T}{T}$. In contrast to the more commonly used definition

of instantaneous regret: $r_t = f(\mathbf{z}^*) - f(\mathbf{z}_t)$, the slightly modified definition introduced here is justified in the sense that the induced definition of simple regrets, which is the ultimate goal of the theoretical analysis, obtained in both cases are equivalent, i.e., $\min_{t=1,\dots,T} f(\mathbf{z}^*) - f_t^* = \min_{t=1,\dots,T} f(\mathbf{z}^*) - f(\mathbf{z}_t)$.

The instantaneous regret defined above can be further decomposed as

$$\begin{aligned} r_t =& f(\mathbf{z}^*) - f_t^* = f(\mathbf{z}^*) - \max_{t'=1,\dots,t} f(\mathbf{z}_{t'}) \\ =& f(\mathbf{z}^*) - \max\{f_{t-1}^*, f(\mathbf{z}_t)\} \end{aligned} \tag{A.2}$$

Note that in our algorithm, the BO iterations can be divided into two types: **1)** t^+ such that $n_{t^+} = N$: those iterations that are not early-stopped; and **2)** t^- such that $n_{t^-} < N$: those that are early-stopped. For all t^+, it follows from Equation A.2 that $r_t = f(\mathbf{z}^*) - \max\{f_{t-1}^*, f(\mathbf{z}_t)\} \leq f(\mathbf{z}^*) - f(\mathbf{z}_t) = f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t]) = f(\mathbf{z}^*) - f([\mathbf{z}_t, N]) \triangleq r_{t^+}$; for all t^-, from Equation A.2, we have that $r_t = f(\mathbf{z}^*) - \max\{f_{t-1}^*, f(\mathbf{z}_t)\} \leq f(\mathbf{z}^*) - f_{t-1}^* \triangleq r_{t^-}$. In the following, we will focus on the analysis of the sum of all r_{t^+} and all r_{t^-}: $R'_T = \sum_{t^+} r_{t^+} + \sum_{t^-} r_{t^-}$. As a result of the definition, R'_T is an upper bound of R_T, therefore, sub-linear growth of R'_T implies that R_T also grows sub-linearly.

Next, note that for all t^- such that $n_{t^-} < N$ (when $\mathbf{x}_t$ is early-stopped),

$$\begin{aligned} r_{t^-} = f(\mathbf{z}^*) - f_{t-1}^* &= \underline{f(\mathbf{z}^*) - f([\mathbf{x}_t, N])} + \underline{f([\mathbf{x}_t, N]) - f_{t-1}^*} \\ &\overset{(1)}{\leq} \underline{f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])} + \underline{f([\mathbf{x}_t, N]) - f_{t-1}^*} \end{aligned} \tag{A.3}$$

in which (1) results from Assumption 4.1. As a result, R'_T can be re-written as

$$
\begin{aligned}
R'_T &\overset{(1)}{=} \sum_{\{t|n_t=N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t<N\}} [f(\mathbf{z}^*) - f^*_{t-1}] \\
&= \sum_{\{t|n_t=N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t<N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \\
&\quad \sum_{\{t|n_t<N\}} [f([\mathbf{x}_t, N]) - f^*_{t-1}] \\
&\overset{(2)}{\leq} \sum_{\{t|n_t=N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, N])] + \sum_{\{t|n_t<N\}} [f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])] + \\
&\quad \sum_{\{t|n_t<N\}} [f([\mathbf{x}_t, N]) - f^*_{t-1}] \\
&\overset{(3)}{=} \sum_{t=1}^{T} [f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])] + \sum_{\{t|n_t<N\}} [f([\mathbf{x}_t, N]) - f^*_{t-1}] \\
&\triangleq \sum_{t=1}^{T} r_{t,1} + \sum_{\{t|n_t<N\}} r_{t,2} \\
&\triangleq R_{T,1} + R_{T,2}
\end{aligned}
\tag{A.4}
$$

in which (1) makes use of the definition of R'_T, (2) results from Equation A.3 and (3) follows by combining the first two terms on the previous line. The first term following (3) of Equation A.4 is summed over all time steps, whereas the second term is only summed over those time steps that are early-stopped ($n_t < N$). As mentioned earlier, in the sequel, we will attempt to prove an upper bound on the expected value of R'_T,

$$
\mathbb{E}[R'_T] \leq \mathbb{E}[R_{T,1}] + \mathbb{E}[R_{T,2}] \tag{A.5}
$$

in which the expectation is taken with respect to the posterior probabilities used in the BOS problems, corresponding to those iterations that are early-stopped:

$$
\Pi_{t \in \{t'|t'=1,\ldots,T, n_{t'}<N\}} \mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t | \mathbf{y}_{t,n_t}).
$$

Note that the probability distributions are independent across all the early-stopped iterations, therefore, for each early-stopped iteration t, the expectations of both $r_{t,1}$ and $r_{t,2}$ are only taken over the specific distribution: $\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t | \mathbf{y}_{t,n_t})$; whereas for each not-early-stopped iteration t, $\mathbb{E}[r_{t,1}] = r_{t,1}$ (whereas $r_{t,2}$ is absent). In the next two sections, we will prove upper bounds on $\mathbb{E}[R_{T,1}]$ and $\mathbb{E}[R_{T,2}]$ respectively.

A.3.2 Upper Bound on $\mathbb{E}[R_{T,1}]$

In this section, we will upper-bound the term $\mathbb{E}[R_{T,1}]$. As mentioned in the main text, for simplicity, we will focus on the case in which the underlying domain $\mathcal{Z}$ is discrete, i.e., $|\mathcal{Z}| < \infty$. To begin with, we will need a supporting lemma showing a uniform upper bound over the entire domain.

Lemma A.1. *Suppose that $\delta \in (0, 1)$ and $\beta_t \triangleq 2\log(|\mathcal{Z}|t^2\pi^2/6\delta)$. Then, with probability $\geq 1 - \delta$*

$$|f(\mathbf{z}) - \mu_{t-1}(\mathbf{z})| \leq \beta_t^{1/2}\sigma_{t-1}(\mathbf{z}) \quad \forall \mathbf{z} \in \mathcal{Z}, t \geq 1 \, .$$

The proof of lemma D.2 makes use of standard Gaussian tail bounds and a number of union bounds, and the proof is identical to the proof of lemma 5.1 in (Srinivas et al., 2010). The next supporting lemma makes use of the Lipschitz continuity of f to bound the differences between function values whose inputs only differ by the dimension corresponding to the number of training epochs.

Lemma A.2. *Suppose that Assumption 4.2 holds and let $\delta' \in (0, 1)$. Then, with probability $\geq 1 - \delta'$,*

$$|f([\mathbf{x}, N]) - f([\mathbf{x}, n])| \leq Nb\sqrt{\log \frac{da}{\delta'}} \qquad \forall \mathbf{x}, n = 1, \ldots, N \, .$$

Proof. Let $\mathbf{z} = [\mathbf{x}, n]$ denote the input to the objective function f. Assumption

4.2, together with a union bound over $j = 1, \ldots, d$, implies that with probability $\geq 1 - dae^{-(\frac{L}{b})^2}$,

$$|f(\mathbf{z}) - f(\mathbf{z}')| \leq L||\mathbf{z} - \mathbf{z}'||_1 \qquad \forall \mathbf{z} \in \mathcal{Z}$$

Since $[\mathbf{x}, N]$ and $[\mathbf{x}, n]$ differ only by the dimension corresponding to the number of training epochs, we have that

$$|f([\mathbf{x}, N]) - f([\mathbf{x}, n])| \leq LN$$

Then, the lemma follows by letting $\delta' = dae^{-(\frac{L}{b})^2}$. □

The next lemma bounds $E[r_{t,1}]$ by the Gaussian process posterior standard deviation with some scaling constants.

Lemma A.3. *Let $\delta, \delta' \in (0, 1)$ and $\kappa \geq 1$ be the constant used in C2 in the BO-BOS algorithm. Then, at iteration t of the BO-BOS algorithm, we have that, with probability $\geq 1 - \delta - \delta'$,*

$$\mathbb{E}[r_{t,1}] \leq 2\kappa\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) + Nb\sqrt{\log\frac{da}{\delta'}}\mathbb{1}_{n_t<N} .$$

Proof. Firstly, with probability $\geq 1 - \delta$,

$$\begin{aligned} f(\mathbf{z}^*) \overset{(1)}{=} f([\mathbf{x}^*, N]) &\overset{(2)}{\leq} \mu_{t-1}([\mathbf{x}^*, N]) + \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}^*, N]) \\ &\overset{(3)}{\leq} \mu_{t-1}([\mathbf{x}_t, N]) + \beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) \end{aligned} \tag{A.6}$$

in which (1) follows from Assumption 4.1 which states that, for each $\mathbf{x}$, the function value is monotonically non-decreasing in the number of training epochs, which implies that at the (unknown) global maximum $\mathbf{z}^*$, the dimension corresponding to the number of epochs is equal to N. (2) makes use of Lemma D.2, whereas (3) is due to the way $\mathbf{x}_t$ is selected in the algorithm, i.e., $\mathbf{x}_t = \text{argmax}_{\mathbf{x}}\mu_{t-1}([\mathbf{x}, N]) +$

$\sqrt{\beta_t}\sigma_{t-1}([\mathbf{x}, N])$. As a result, we have that with probability $\geq 1-\delta-\delta'$

$$\begin{aligned}
\mathbb{E}[r_{t,1}] &= \mathbb{E}[f(\mathbf{z}^*) - f([\mathbf{x}_t, n_t])] \overset{(1)}{\leq} \mathbb{E}[\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) + \mu_{t-1}([\mathbf{x}_t, N]) \\
&\quad - f([\mathbf{x}_t, n_t])] \\
&= \mathbb{E}[\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) + \mu_{t-1}([\mathbf{x}_t, N]) - f([\mathbf{x}_t, N]) \\
&\quad + f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \\
&\overset{(2)}{\leq} \mathbb{E}[2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N])] + \mathbb{E}[f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \\
&\overset{(3)}{\leq} \mathbb{E}[2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N])] + Nb\sqrt{\log\frac{da}{\delta'}}\mathbb{1}_{n_t<N} \\
&\overset{(4)}{\leq} 2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, N]) + Nb\sqrt{\log\frac{da}{\delta'}}\mathbb{1}_{n_t<N} \\
&\overset{(5)}{\leq} 2\kappa\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) + Nb\sqrt{\log\frac{da}{\delta'}}\mathbb{1}_{n_t<N}
\end{aligned} \tag{A.7}$$

in which (1) follows from Equation A.6, and (2) results from Lemma D.2 and the linearity of the expectation operator. $\mathbb{1}_{n_t<N}$ in (3) is the indicator function, which takes the value of 1 if the event $n_t < N$ is true and 0 otherwise. (3) is obtained by analyzing two different cases separately: if $n_t = N$ ($\mathbf{x}_t$ is not early-stopped), then $\mathbb{E}[f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] = 0$; if $n_t < N$ (x_t is early-stopped), then $\mathbb{E}[f([\mathbf{x}_t, N]) - f([\mathbf{x}_t, n_t])] \leq \mathbb{E}[Nb\sqrt{\log\frac{da}{\delta'}}] = Nb\sqrt{\log\frac{da}{\delta'}}$ with probability $\geq 1-\delta'$ following Lemma A.2. (4) is due to the fact that $\sigma_{t-1}([\mathbf{x}_t, N])$ only depends on the observations up to step $t-1$ and is not dependent on the probability $\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t})$. (5) follows from the design of the algorithm; in particular, if $n_t < N$, then $\kappa\sigma_{t-1}([\mathbf{x}_t, n_t]) \geq \sigma_{t-1}([\mathbf{x}_t, N])$ is guaranteed by C2; otherwise, if $n_t = N$, then $\kappa\sigma_{t-1}([\mathbf{x}_t, n_t]) \geq \sigma_{t-1}([\mathbf{x}_t, n_t]) = \sigma_{t-1}([\mathbf{x}_t, N])$ since $\kappa \geq 1$. □

Subsequently, we can upper bound $\mathbb{E}[R_{T,1}] = \sum_{t=1}^{T}\mathbb{E}[r_{t,1}]$ by extensions of Lemma 5.3 and 5.4 from (Srinivas et al., 2010), which are presented here for completeness. The following lemma connects the information gain about the objective function with the posterior predictive variance, whose proof results

from straightforward extension of Lemma 5.3 of (Srinivas et al., 2010).

Lemma A.4. *Let $\mathbf{y}_T$ be a set of observations of size T, and let $\mathbf{f}_T$ be the corresponding function values. The information gain about $\mathbf{f}_T$ from observing $\mathbf{y}_T$ is*

$$I(\mathbf{y}_T; \mathbf{f}_T) = \frac{1}{2}\sum_{t=1}^{T} \log[1 + \sigma^{-2}\sigma_{t-1}^2([\mathbf{x}_t, n_t])] \,.$$

Next, we use the following lemma to bound the sum of the first term of the expected instantaneous regret as given in Lemma A.3.

Lemma A.5. *Let $\delta \in (0,1)$, $C_1 \triangleq \frac{8}{\log(1+\sigma^{-2})}$, $\beta_t \triangleq 2\log(|\mathcal{Z}|t^2\pi^2/6\delta)$, and $\gamma_T \triangleq \max_{A\in\mathcal{Z},|A|=T} I(\mathbf{y}_A; \mathbf{f}_A)$ is the maximum information gain about f from any subset of size T. Then,*

$$\sum_{t=1}^{T} 2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) \leq \sqrt{TC_1\beta_T I(\mathbf{y}_T; \mathbf{f}_T)} \leq \sqrt{TC_1\beta_T\gamma_T} \,.$$

Proof. Firstly, we have that

$$\begin{aligned}
(2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]))^2 = 4\beta_t\sigma_{t-1}^2([\mathbf{x}_t, n_t]) &\overset{(1)}{\leq} 4\beta_T\sigma^2[\sigma^{-2}\sigma_{t-1}^2([\mathbf{x}_t, n_t])] \\
&\overset{(2)}{\leq} 4\beta_T\sigma^2\frac{\sigma^{-2}}{\log(1+\sigma^{-2})}\log[1 + \sigma^{-2}\sigma_{t-1}^2([\mathbf{x}_t, n_t])] \\
&\leq \beta_T\frac{8}{\log(1+\sigma^{-2})}\frac{1}{2}\log[1 + \sigma^{-2}\sigma_{t-1}^2([\mathbf{x}_t, n_t])]
\end{aligned} \tag{A.8}$$

in which (1) holds since β_t is monotonically increasing in t; (2) results from the fact that $\sigma^{-2}x \leq \frac{\sigma^{-2}}{\log(1+\sigma^{-2})}\log[1 + \sigma^{-2}x]$ for $x \in (0,1]$, whereas $0 < \sigma_{t-1}^2([\mathbf{x}_t, n_t]) \leq 1$. Next, summing over $t = 1, \ldots, T$, we get

$$\begin{aligned}
\sum_{t=1}^{T}(2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]))^2 &\leq \beta_T\frac{8}{\log(1+\sigma^{-2})}\frac{1}{2}\sum_{t=1}^{T}\log[1 + \sigma^{-2}\sigma_{t-1}^2([\mathbf{x}_t, n_t])] \\
&\overset{(1)}{=} \beta_T\frac{8}{\log(1+\sigma^{-2})}I(\mathbf{y}_T; \mathbf{f}_T) \overset{(2)}{\leq} C_1\beta_T\gamma_T
\end{aligned} \tag{A.9}$$

in which (1) results from Lemma A.4, and (2) follows from the definitions of C_1 and γ_T. Next, making use of the Cauchy-Schwarz inequality, we get

$$\begin{aligned}\sum_{t=1}^{T} 2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) &\leq \sqrt{T}\sqrt{\sum_{t=1}^{T}(2\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]))^2} \\ &\leq \sqrt{C_1 T \beta_T \gamma_T}\end{aligned} \tag{A.10}$$

which completes the proof. □

Next, putting everything together, we get the follow lemma on the upper bound on $\mathbb{E}[R_{T,1}]$.

Lemma A.6. *Suppose that Assumptions 4.1 and 4.2 hold. Let $\delta, \delta' \in (0,1)$, $C_1 = 8/\log(1+\sigma^{-2})$, $\beta_t = 2\log(|\mathcal{Z}|t^2\pi^2/6\delta)$, $\kappa \geq 1$ be the constant used in C2 in the BO-BOS algorithm, and $\gamma_T = \max_{A\in\mathcal{Z},|A|=T} I(\mathbf{y}_A; \mathbf{f}_A)$. Let $\tau_T = \sum_{t=1}^{T} \mathbb{1}_{n_t<N}$ be the number of BO iterations in which early stopping happens from iterations 1 to T. Assume that f is a sample from a GP, and $y(\mathbf{z}) = f(\mathbf{z}) + \epsilon \forall z \in \mathcal{Z}$ in which $\epsilon \sim N(0, \sigma^2)$. Then, with probability $\geq 1-\delta-\delta'$,*

$$\mathbb{E}[R_{T,1}] = \sum_{t=1}^{T}\mathbb{E}[r_{t,1}] \leq \kappa\sqrt{TC_1\beta_T\gamma_T} + Nb\sqrt{\log\frac{da}{\delta'}}\tau_T \quad \forall T \geq 1\,.$$

Proof.

$$\begin{aligned}\mathbb{E}[R_{T,1}] \overset{(1)}{=} \sum_{t=1}^{T}\mathbb{E}[r_{t,1}] &\overset{(2)}{\leq} \sum_{t=1}^{T}[2\kappa\beta_t^{1/2}\sigma_{t-1}([\mathbf{x}_t, n_t]) + Nb\sqrt{\log\frac{da}{\delta'}}\mathbb{1}_{n_t<N}] \\ &\overset{(3)}{\leq} \kappa\sqrt{TC_1\beta_T\gamma_T} + \sum_{t=1}^{T} Nb\sqrt{\log\frac{da}{\delta'}}\mathbb{1}_{n_t<N} \\ &= \kappa\sqrt{TC_1\beta_T\gamma_T} + Nb\sqrt{\log\frac{da}{\delta'}}\sum_{t=1}^{T}\mathbb{1}_{n_t<N} \\ &= \kappa\sqrt{TC_1\beta_T\gamma_T} + Nb\sqrt{\log\frac{da}{\delta'}}\tau_T\end{aligned} \tag{A.11}$$

in which (1) follows from the linearity of the expectation operator, (2) results from Lemma A.3, and (3) follows from Lemma A.5. □

A.3.3 Upper Bound on $\mathbb{E}[R_{T,2}]$

In this section, we prove an upper bound on $\mathbb{E}[R_{T,2}]$. A few supporting lemmas will be presented and proved first. To begin with, the next lemma derives the appropriate choice of the incumbent values used in the BOS problems in different iterations of the BO-BOS algorithm.

Lemma A.7. *Let the objective function f be a sample from a GP and $y(\mathbf{z}) = f(\mathbf{z}) + \epsilon \, \forall \mathbf{z} \in \mathcal{Z}$ in which $\epsilon \sim N(0, \sigma^2)$. Let $\delta'' \in (0, 1)$. At iteration $t > 1$, define $f^*_{t-1} \triangleq \max_{t'=1,\ldots,t-1} f(\mathbf{z}_{t'})$ and $y^*_{t-1} \triangleq \max_{t'=1,\ldots,t-1} y_{t'}$; for iteration $t = 1$, define $f^*_0 \triangleq 0$ and $y^*_0 \triangleq 0$. Then with probability $\geq 1 - \delta''$,*

$$f^*_{t-1} \geq y^*_{t-1} - \xi_t \quad \forall t \geq 1$$

in which

$$\xi_t = \sqrt{2\sigma^2 \log \frac{\pi^2 t^2 (t-1)}{6\delta''}} \quad \forall t > 1$$

and $\xi_1 = 0$.

Proof. The lemma trivially holds for $t = 1$. Assume we are at iteration $t > 1$ of the BO-BOS algorithm, and let $t' \in \{1, 2, \ldots, t-1\}$. Since $y_{t'} = f(\mathbf{z}_{t'}) + \epsilon$, in which $\epsilon \sim N(0, \sigma^2)$, we have that $y_{t'} \sim N(f(\mathbf{z}_{t'}), \sigma^2)$. Making use of the *upper deviation inequality* for Gaussian distribution and the definition of ξ_t, we get

$$\mathbb{P}[y_{t'} \geq f(\mathbf{z}_{t'}) + \xi_t] \leq e^{-\frac{{\xi_t}^2}{2\sigma^2}} = \frac{6\delta''}{\pi^2 t^2 (t-1)} \tag{A.12}$$

Denote the event that $\{\exists\, t' \in \{1, 2, \ldots, t-1\} \text{ s.t. } y_{t'} \geq f(\mathbf{z}_{t'}) + \xi_t\}$ as $\mathcal{A}_t$. Next, taking a union bound over the entire observation history $t' \in \{1, 2, \ldots, t-1\}$,

we get

$$\mathbb{P}[\mathcal{A}_t] \leq \sum_{t'=1}^{t-1} \mathbb{P}[y_{t'} \geq f(\mathbf{z}_{t'}) + \xi_t] \leq (t-1)\frac{6\delta''}{\pi^2 t^2 (t-1)} = \frac{6\delta''}{\pi^2 t^2} \tag{A.13}$$

which implies that at iteration t, with probability $\geq 1 - \frac{6\delta''}{\pi^2 t^2}$, $y_{t'} - f(\mathbf{z}_{t'}) < \xi_t \, \forall t' \in \{1, 2, \ldots, t-1\}$, which further suggests that $y^*_{t-1} - f^*_{t-1} \leq \xi_t$ at iteration t. Next, taking a union bound over $t \geq 1$, we get

$$\mathbb{P}[\exists t \geq 1 \text{ s.t. } \mathcal{A}_t \text{ holds}] \leq \sum_{t \geq 1} \mathbb{P}[\mathcal{A}_t] \leq \sum_{t \geq 1} \frac{6\delta''}{\pi^2 t^2} = \delta'' \tag{A.14}$$

which suggests that, with probability $\geq 1 - \delta''$, $y^*_{t-1} - f^*_{t-1} \leq \xi_t \; \forall t \geq 1$, and thus completes the proof. □

The next lemma shows that, with appropriate choices of the incumbent value, the posterior probability used in Bayesian optimal stopping is upper-bounded.

Lemma A.8. *If in iteration t of the BO-BOS algorithm, the BOS algorithm is run with the incumbent value $y^*_{t-1} - \gamma_t$ and the corresponding cost parameters K_1, K_2 and c_{d_0}, and the algorithm early-stops after $n_t < N$ epochs, then with probability $\geq 1 - \delta''$,*

$$\mathbb{P}(f([\mathbf{x}_t, N]) > f^*_{t-1} | \mathbf{y}_{t,n_t}) \leq \frac{K_2 + c_{d_0}}{K_1} \quad \forall t \geq 1 \, . \tag{A.15}$$

Proof. Recall that when running the Bayesian optimal stopping algorithm in iteration t of BO-BOS, we only early-stop the experiment ($n_t < N$) when we can safely conclude that the performance of the currently evaluated hyperparameter $\mathbf{x}_t$ will end up having smaller (or equal) validation accuracy than the currently observed optimum offset by a noise correction term: $y^*_{t-1} - \xi_t$; i.e., when the expected loss of decision d_1 is the smallest among all decisions. Therefore, when

the evaluation of $\mathbf{x}_t$ is early-stopped after $n_t < N$ epochs, we can conclude that

$$
\begin{aligned}
&K_1\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t}) \\
&\leq \mathbb{E}_{y_{t,n_t+1}|\mathbf{y}_{t,n_t}}\Big[\min\{K_1\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t+1}), \\
&K_2\mathbb{P}(f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t+1}), \mathbb{E}_{y_{t,n_t+2}|\mathbf{y}_{t,n_t+1}}[\rho_{t,n_t+2}(\mathbf{y}_{t,n_t+2})] + c_{d_0}\}\Big] + c_{d_0} \\
&\leq \mathbb{E}_{y_{t,n_t+1}|\mathbf{y}_{t,n_t}}[K_2\mathbb{P}(f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t+1})] + c_{d_0} \\
&\leq K_2\mathbb{E}_{y_{t,n_t+1}|\mathbf{y}_{t,n_t}}[\mathbb{P}(f([\mathbf{x}_t, N]) \leq y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t+1})] + c_{d_0} \\
&\leq K_2 + c_{d_0}
\end{aligned}
\tag{A.16}
$$

Equation A.16, together with Lemma A.7, implies that

$$
\begin{aligned}
\mathbb{P}(f([\mathbf{x}_t, N]) > f^*_{t-1}|\mathbf{y}_{t,n_t}) \leq &\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t}) \\
\leq &\frac{K_2 + c_{d_0}}{K_1}
\end{aligned}
\tag{A.17}
$$

which holds uniformly for all $t \geq 1$ with probability $\geq 1 - \delta''$. □

Subsequently, we use the next Lemma to upper-bound $\mathbb{E}[R_T^2]$ by the BOS cost parameters. We set K_2 and c_{d_0} as constants, and use different values of K_1 in different iterations t of the BO-BOS algorithm, which is represented by $K_{1,t}$.

Lemma A.9. *In iteration t of the BO-BOS algorithm, define $\frac{K_2+c_{d_0}}{K_{1,t}} \triangleq \eta_t$. Then, with probability $\geq 1 - \delta''$,*

$$
\mathbb{E}[R_{T,2}] \leq \sum_{t=1}^{T} \eta_t \quad \forall T \geq 1 \, .
$$

Proof. Recall that according to Assumption 4.1, the value of the objective function f is bounded in the range $[0, 1]$. In iteration t, assume we early-stop the

evaluation of $\mathbf{x}_t$ after $n_t < N$ epochs, then

$$\begin{aligned}\mathbb{E}[f([\mathbf{x}_t, N]) - f^*_{t-1}|\mathbf{y}_{t,n_t}] &\overset{(1)}{\leq} \mathbb{E}[\mathbb{1}_{f([\mathbf{x}_t,N])-f^*_{t-1}>0}|\mathbf{y}_{t,n_t}] \\ &= \mathbb{P}(f([\mathbf{x}_t, N]) > f^*_{t-1}|\mathbf{y}_{t,n_t})\end{aligned} \tag{A.18}$$

Step (1) in Equation A.18 is because $x \leq \mathbb{1}_{x>0}\ \forall x \in [-1, 1]$ and substituting $x = f([\mathbf{x}_t, N]) - f^*_{t-1}$. As a result, with probability $\geq 1 - \delta''$

$$\begin{aligned}\mathbb{E}[R_{T,2}] &\overset{(1)}{=} \sum_{\{t|n_t<N\}} \mathbb{E}[r_{t,2}] \overset{(2)}{=} \sum_{\{t|n_t<N\}} \mathbb{E}[f([\mathbf{x}_t, N]) - f^*_{t-1}|\mathbf{y}_{t,n_t}] \\ &\overset{(3)}{\leq} \sum_{\{t|n_t<N\}} \mathbb{P}(f([\mathbf{x}_t, N]) > f^*_{t-1}|\mathbf{y}_{t,n_t}) \overset{(4)}{\leq} \sum_{\{t|n_t<N\}} \eta_t \leq \sum_{t=1}^{T} \eta_t\end{aligned} \tag{A.19}$$

in which (1) follows from the linearity of expectation, (2) holds because the Expectation of $r_{t,2}$ is taken over $\mathbb{P}(f([\mathbf{x}_t, N]) > y^*_{t-1} - \xi_t|\mathbf{y}_{t,n_t})$, (3) results from Equation A.18, and (4) follows from Lemma A.8. This completes the proof. □

A.3.4 Putting Things Together

In this section, we put everything from the previous two sections together to prove the main theorems.

A.3.4.1 Proof of Theorem 4.1

Theorem 4.1 can be proven by combining Lemmas A.6 and A.9, and making use of the fact that $S_T \leq \frac{R_T}{T}$.

A.3.4.2 Proof of Theorem 4.2

Below we analyze the asymptotic behavior of each of the three terms in the upper bound of $\mathbb{E}[S_T]$ in Theorem 4.1, which is re-presented here for ease of reference.

$$\mathbb{E}[S_T] \leq \frac{\kappa\sqrt{TC_1\beta_T\gamma_T}}{T} + \frac{\sum_{t=1}^{T}\eta_t}{T} + \frac{1}{T}Nb\sqrt{\log\frac{da}{\delta'}}\tau_T\ . \tag{A.20}$$

A.3.4.2.1 The first term in the upper bound of $\mathbb{E}[S_T]$ Firstly, the first term in the upper bound matches the upper bound on the simple regret of the GP-UCB algorithm (Srinivas et al., 2010) (up to the constant κ). The maximum information gain, γ_T, has been analyzed for a few of the commonly used kernels in GP (Srinivas et al., 2010). For example, for the Square Exponential kernel, $\gamma_T = O((\log T)^{d+1})$, whereas for the Matérn kernel with $\nu > 1$, $\gamma_T = O(T^{d(d+1)/(2\nu+d(d+1))} \log T)$. Plugging both expressions of γ_T into Theorem 4.1, together with the expression of β_T as given in Theorem 4.1, shows that both kernels lead to sub-linear growth of the term $\sqrt{TC_1\beta_T\gamma_T}$, which implies that the first term in the upper bound of $\mathbb{E}[S_T]$ asymptotically goes to 0.

A.3.4.2.2 The second term in the upper bound of $\mathbb{E}[S_T]$ Given that $K_{1,t}$ is an increasing sequence with $K_{1,1} \geq K_2 + c_{d_0}$, the series $\sum_{t=1}^{T} \eta_t = \sum_{t=1}^{T} \frac{K_2 + c_{d_0}}{K_{1,t}}$ grows sub-linearly, thus making the second term in the upper bound of $\mathbb{E}[S_T]$ given in Theorem 4.1, $\frac{\sum_{t=1}^{T} \eta_t}{T}$, asymptotically go to 0.

A.3.4.2.3 The third term in the upper bound of $\mathbb{E}[S_T]$ Next, suppose that $K_{1,t}$ becomes $+\infty$ for the first time at iteration T_0. Since $K_{1,t}$ is a non-decreasing sequence, $K_{1,t} = +\infty$ for all $t \geq T_0$. Therefore, for $t \geq T_0$, decision d_1 will never be taken and the algorithm will never early-stop. In other words, $n_t = N$ for all $t \geq T_0$.

Therefore, we can conclude that $\tau_T \leq T_0$ for all $T \geq 1$. As a result, the last term in the upper bound on $\mathbb{E}[S_T]$ in Theorem 4.1 can be upper-bounded by

$$\frac{\tau_T}{T} Nb \sqrt{\log \frac{da}{\delta'}} \leq \frac{T_0 Nb \sqrt{\log \frac{da}{\delta'}}}{T} = O\left(\frac{1}{T}\right) \tag{A.21}$$

which asymptotically goes to 0 as T goes to $+\infty$, because the numerator term is a constant. Therefore, this term also asymptotically vanishes in the upper bound.

To summarize, if the BOS parameters are selected according to Theorem 4.2,

we have that

$$\mathbb{E}[S_T] = O\left(\frac{\sqrt{T\beta_T\gamma_T}}{T} + \frac{\sum_{t=1}^{T}\eta_t}{T} + \frac{1}{T}\right) \tag{A.22}$$

and $\mathbb{E}[S_T]$ goes to zero asymptotically.

A.4 Additional Experimental Details

In each experiment, the same initializations (6 initial points if not further specified) are used for all BO-based methods: GP-UCB, BOCA, LC Prediction, and BO-BOS. The Square Exponential kernel is used for BOCA since the algorithm is only given for this kernel (Kandasamy et al., 2017), the other BO-based algorithms use the Matérn kernel; the kernel hyperparameters are updated by maximizing the Gaussian process marginal likelihood after every 10 BO iterations. In the BO-BOS algorithm, since the number of training epochs is an input to the GP surrogate function, some of the intermediate observations ($n < N$) can be used as additional input to GP to improve the modeling of the objective function. However, using the observation after every epoch as input leads to poor scalability. Therefore, for all experiments with $N = 50$ (which include most of the experiments), we use the observations after first, 10-th, 20-th, 30-th and 40-th epochs as additional inputs to the GP surrogate function; whereas for the RL experiment with $N = 100$ in section 4.5.3.1, we use the 1-th, 20-th, 40-th, 60-th and 80-th intermediate observations as additional inputs. 100,000 forward simulation samples are used for each BOS algorithm; the grid size of the discretized summary statistics is set to 100; for simplicity, the incumbent value at iteration t is chosen as $y^*_{t-1} = \max_{t'=1,\ldots,t-1} y_{t'}$, thus ignoring the observation noise. In the LC Prediction algorithm (Domhan et al., 2015), learning curve prediction is performed after every 2 epochs. In Hyperband (Li et al., 2017), the successive halving parameter η is set to 3 as recommended by the original authors, and the maximum number of epochs is set to $N = 80$ (we observed

that setting $N = 80$ led to better performance than $N = 50$ since it allows the Hyperband algorithm to run for more epochs overall).

A.4.1 Hyperparameter Tuning for Logistic Regression

In the first set of experiments, we perform hyperparameter tuning for a simple ML model, logistic regression (LR). The LR model is trained using the MNIST image dataset, which consists of 70,000 images of the 10 digits, corresponding to a 10-class classification problem. Three hyperparameters are tuned: the batch size (20 to 500), L2 regularization parameter (10^{-6} to 1.0), and learning rate (10^{-3} to 0.1). We use 80% of the images as the training set and the remaining 20% as the validation set.

Some of the learning curves during a particular run of the BO-BOS algorithm is shown in Fig. A.3. It can be observed that the learning curves that show minimal potential in achieving small validation errors are early-stopped, whereas the promising hyperparameter settings are run for larger number of epochs. The reliability of the early stopping achieved by the BO-BOS algorithm is demonstrated in Fig. A.4. In this figure, the green triangles correspond to the learning curves that are not early-stopped ($n_t = N$), and the red circles represent the final validation errors (after training for the maximum number of epochs N) that *could have been* reached by the early-stopped learning curves ($n_t < N$). Note that the red circles are shown only for the purpose of illustration and are not observed in practice. As displayed in the figure, the early stopping decisions made during the BO-BOS algorithm are reliable, since those early-stopped learning curves all end up having large validation errors.

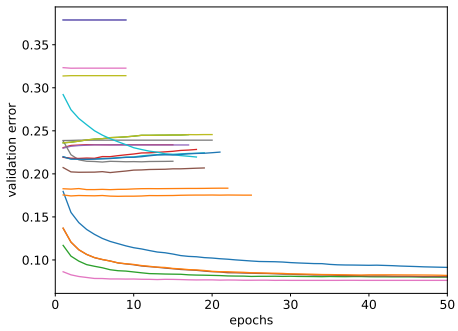


Figure A.3: Some learning curves during the BO-BOS algorithm.

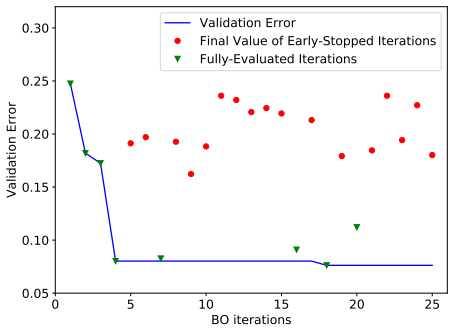


Figure A.4: Illustration of the effectiveness of the early stopping decisions made during the BO-BOS algorithm.

A.4.2 Hyperparameter Tuning for Convolutional Neural Networks

Next, we tune the hyperparameters of convolutional neural networks (CNN) using the CIFAR-10 (Krizhevsky, 2009) and Street View House Numbers (SVHN) dataset (Netzer et al., 2011). Both tasks correspond to 10-class classification problems. For CIFAR-10, $50,000$ images are used as the training set and 10,000 images are used as the validation set; for SVHN, 73,257 and 26032 images are used as the training and validation sets respectively following the original dataset partition. The CNN model consists of three convolutional layers (each followed by a max-pooling layer) followed by one fully-connected layer. We tune six

hyperparameters in both experiments: the batch size (32 to 512), learning rate (10^{-7} to 0.1), learning rate decay (10^{-7} to 10^{-3}), L2 regularization parameter (10^{-7} to 10^{-3}), the number of convolutional filters in each layer (128 to 256), and the number of units in the fully-connected layer (256 to 512).

A.4.3 Policy Search for Reinforcement Learning

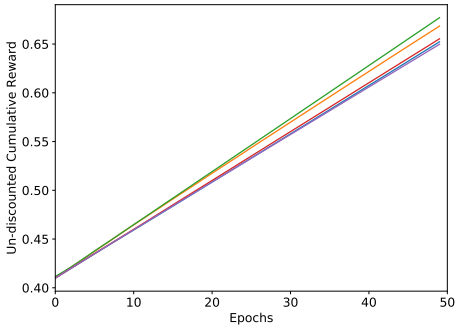
We apply our algorithm to a continuous control task: the Swimmer-v2 environment from OpenAI Gym, MuJoCo (Brockman et al., 2016; Todorov et al., 2012). The task involves controlling two joints of a swimming robot to make it swim forward as fast as possible. The state of the robot is represented by an 8-dimensional feature vector, and the action space is 2-dimensional corresponding to the two joints. We use a linear policy, in which the policy is represented by an 8×2 matrix that maps each state vector to the corresponding action vector. In this setting, the input parameters, $\mathbf{x}$, to the GP-UCB and BO-BOS algorithms are the 16 parameters of the policy matrix, and the objective function is the discounted cumulative rewards in an episode. Each episode of the task consists of 1,000 steps. We set N (the maximum number of epochs) to be smaller than $1,000$ by treating a fixed number of consecutive steps as one single epoch. E.g., we can set $N = 50$ or $N = 100$ by treating every 20 or 10 consecutive steps as one epoch respectively. The rewards are clipped, scaled, and normalized such that the discounted cumulative rewards of each episode is bounded in the range $[0, 1]$; for each evaluated policy, we also record the un-discounted and un-scaled cumulative rewards, which are the ultimate objective to be maximized and reported in Fig. 4.3a in the main text. Each policy evaluation consists of running 5 independent episodes with the given policy, and returning the average discounted cumulative rewards, i.e., average return, as the observed function value.

As mentioned in the main text, the rewards are discounted in order to make the objective function, the discounted cumulative rewards, resemble the learning

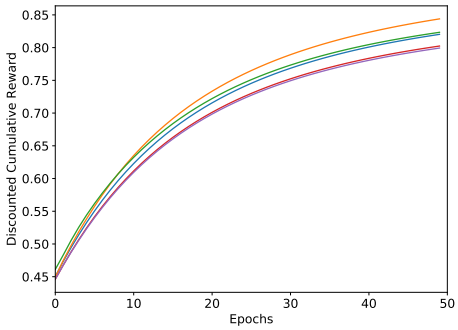
curves of ML models, such that the BO-BOS algorithm can be naturally applied. This rationale is illustrated in Fig. A.5, which plots some example un-discounted ($\gamma = 1.0$) and discounted ($\gamma = 0.9$) cumulative rewards respectively. The figures indicate that, compared with un-discounted cumulative rewards, discounted cumulative rewards bear significantly closer resemblance to the learning curves of ML models, thus supporting the claim made in the main text motivating the use of discounted rewards, as well as the experimental results shown in Fig. 4.3a (specifically, the poor performance of the curve corresponding to $N = 50$ and $\gamma = 1.0$). In addition to the results presented in the main text in section 4.5.3.1, we further present the results with standard errors in Fig. A.6, to emphasize the significant performance advantage offered by BO-BOS compared with GP-UCB. To avoid clutter, we only present the results with error bar for GP-UCB with $\gamma = 1.0$ and BO-BOS with $N = 50$ and $\gamma = 0.9$, which are best-performing settings for GP-UCB and BO-BOS respectively.

A.4.4 Joint Hyperparameter Tuning and Feature Selection

In this set of experiments, we use the gradient boosting model (XGBoost (Chen and Guestrin, 2016)), tuning four hyperparameters: the learning rate (10^{-3} to 0.5), maximum depth of each decision tree (2 to 15), feature sub-sampling ratio for each tree (0.3 to 1.0), and L1 regularization parameter (0.0 to 5.0). We use the email spam dataset from the UCI Machine Learning Repository (Dheeru and Karra Taniskidou, 2017), which represents a binary classification problem: whether the email is a spam or not. We use 3065 emails as the training set and the remaining 1536 emails as the validation set; each email consists of 57 features. The maximum number of features for each hyperparameter setting is set as $N = 50$.



(a) Un-discounted ($\gamma = 1.0$).



(b) Discounted ($\gamma = 0.9$).

Figure A.5: Example curves of un-discounted and discounted cumulative rewards in RL.

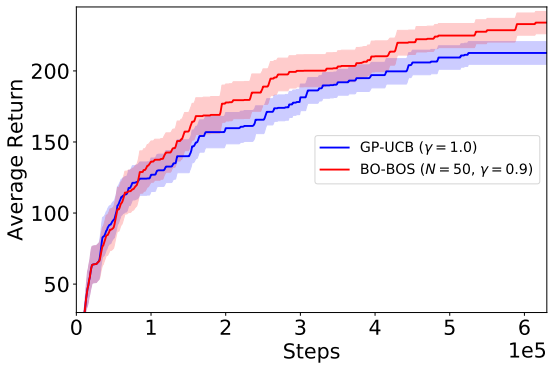


Figure A.6: Best-found return (averaged over 5 episodes) v.s. the total number of steps of the robot in the environment (averaged over 30 random initializations) using the Swimmer-v2 task, with standard error.

Appendix B

Appendix for Chapter 5

B.1 Construction of Random Fourier Features

As mentioned in Section 5.2, in this work, we focus on the widely used Squared Exponential (SE) kernel: $k(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp(-\|\mathbf{x} - \mathbf{x}'\|_2^2 / (2l^2))$ in which l is the length scale and σ_0^2 is the signal variance. $\sigma_0^2 = 1$ is usually the default value, which we use in all experiments. We construct the random features following the work of (Rahimi and Recht, 2007). Specifically, for the SE kernel with length scale l, the spectral density follows a d-dimensional Gaussian distribution: $p(s) = \mathcal{N}(0, \frac{1}{l^2}\mathbf{I}_{d\times d})$. To begin with, we draw M independent samples of $\{\mathbf{s}_i\}_{i=1,\dots,M}$ from $p(\mathbf{s})$ (every $\mathbf{s}_i$ is a d-dimensional vector), and M independent samples of $\{b_i\}_{i=1,\dots,M}$ from the uniform distribution over $[0, 2\pi]$ (every b_i is a scalar). Next, for an input $\mathbf{x}$, the corresponding M-dimensional random features (basis functions) can be constructed as $\boldsymbol{\phi}(\mathbf{x})^\top = [\sqrt{2/M}\cos(\mathbf{s}_i^\top \mathbf{x} + b_i)]_{i=1,\dots,M}$. Each set of random features $\boldsymbol{\phi}(\mathbf{x})$ is then normalized such that $\left\|\boldsymbol{\phi}(\mathbf{x})\right\|_2^2 = \sigma_0^2$. As a result, sharing the random features $\boldsymbol{\phi}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$ among all agents (Section 5.2) can be achieved by simply sharing the parameters $\{\mathbf{s}_i\}_{i=1,\dots,M}$ and $\{b_i\}_{i=1,\dots,M}$. This is easily achievable since it is equivalent to sharing the parameters of the first layer of a neural network model with M units in the hidden layer, in which

$\{\mathbf{s}_i\}_{i=1,\dots,M}$ are the weights (which form a $d \times M$-dimensional weight matrix) and $\{b_i\}_{i=1,\dots,M}$ are the biases.

B.2 GP Posterior Prediction with RFF Approximation

Here we derive the expressions of the posterior predictive mean and variance of a GP with random Fourier features (RFF) approximation (Section 5.2). Recall that we have defined $\boldsymbol{\Phi}(\mathbf{X}_t) = [\boldsymbol{\phi}(\mathbf{x}_1), \dots, \boldsymbol{\phi}(\mathbf{x}_t)]^\top$ which is a $t \times M$-dimensional matrix.

With the RFF approximation, the kernel function is approximated by $k(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$. Define $\hat{\mathbf{K}}_t = [\boldsymbol{\phi}(\mathbf{x}_{t'})^\top \boldsymbol{\phi}(\mathbf{x}_{t''})]_{t',t''=1,\dots,t} = \boldsymbol{\Phi}(\mathbf{X}_t)\boldsymbol{\Phi}(\mathbf{X}_t)^\top$ and $\hat{\mathbf{k}}_t(\mathbf{x}) = [\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}_{t'})]^\top_{t'=1,\dots,t} = \boldsymbol{\Phi}(\mathbf{X}_t)\boldsymbol{\phi}(\mathbf{x})$, which are analogous to $\mathbf{K}_t$ and $\mathbf{k}_t(\mathbf{x})$ in Equation (2.1) with the kernel values $k(\mathbf{x}, \mathbf{x}')$ replaced by the approximate kernel values $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$. With these definitions, we have that

$$\begin{aligned}
\boldsymbol{\Phi}(\mathbf{X}_t)^\top \left[\hat{\mathbf{K}}_t + \sigma^2 \mathbf{I}\right] &= \boldsymbol{\Phi}(\mathbf{X}_t)^\top \left[\boldsymbol{\Phi}(\mathbf{X}_t)\boldsymbol{\Phi}(\mathbf{X}_t)^\top + \sigma^2 \mathbf{I}\right] \\
&= \boldsymbol{\Phi}(\mathbf{X}_t)^\top \boldsymbol{\Phi}(\mathbf{X}_t)\boldsymbol{\Phi}(\mathbf{X}_t)^\top + \sigma^2 \boldsymbol{\Phi}(\mathbf{X}_t)^\top \\
&= \left[\boldsymbol{\Phi}(\mathbf{X}_t)^\top \boldsymbol{\Phi}(\mathbf{X}_t) + \sigma^2 \mathbf{I}\right] \boldsymbol{\Phi}(\mathbf{X}_t)^\top \\
&= \boldsymbol{\Sigma}_t \boldsymbol{\Phi}(\mathbf{X}_t)^\top.
\end{aligned} \tag{B.1}$$

Multiplying both sides by $\boldsymbol{\Sigma}_t^{-1}$ from the left and $(\hat{\mathbf{K}}_t + \sigma^2 \mathbf{I})^{-1}$ from the right, we get

$$\boldsymbol{\Sigma}_t^{-1} \boldsymbol{\Phi}(\mathbf{X}_t)^\top = \boldsymbol{\Phi}(\mathbf{X}_t)^\top (\hat{\mathbf{K}}_t + \sigma^2 \mathbf{I})^{-1}. \tag{B.2}$$

Then multiplying both sides by $\boldsymbol{\phi}(\mathbf{x})^\top$ from the left and $\mathbf{y}_t$ from the right, we get

$$\begin{aligned}\hat{\mu}_t(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\nu}_t = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\Phi}(\mathbf{X}_t)^\top \mathbf{y}_t = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Phi}(\mathbf{X}_t)^\top (\hat{\mathbf{K}}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_t \\ = \hat{\mathbf{k}}_t(\mathbf{x})^\top (\hat{\mathbf{K}}_t + \sigma^2 \mathbf{I})^{-1} \mathbf{y}_t,\end{aligned} \tag{B.3}$$

which proves that the expression of the approximate posterior mean with RFF approximation: $\hat{\mu}_t(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\nu}_t$ matches the expression of the posterior mean of standard GP without RFF approximation, except that the kernel values $k(\mathbf{x}, \mathbf{x}')$ are replaced by the approximate kernel values $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$.

Next, we derive the expression of the approximate posterior variance. Making use of the matrix inversion lemma, we get

$$\begin{aligned}&\hat{\sigma}_t^2(\mathbf{x}) = \sigma^2 \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\phi}(\mathbf{x}) = \sigma^2 \boldsymbol{\phi}(\mathbf{x})^\top (\boldsymbol{\Phi}(\mathbf{X}_t)^\top \boldsymbol{\Phi}(\mathbf{X}_t) + \sigma^2 \mathbf{I})^{-1} \boldsymbol{\phi}(\mathbf{x}) \\ &= \sigma^2 \boldsymbol{\phi}(\mathbf{x})^\top \left[\frac{1}{\sigma^2} \mathbf{I} - \frac{1}{\sigma^2} \boldsymbol{\Phi}(\mathbf{X}_t)^\top \left(\mathbf{I} + \boldsymbol{\Phi}(\mathbf{X}_t) \frac{1}{\sigma^2} \boldsymbol{\Phi}(\mathbf{X}_t)^\top \right)^{-1} \boldsymbol{\Phi}(\mathbf{X}_t) \frac{1}{\sigma^2} \right] \boldsymbol{\phi}(\mathbf{x}) \\ &= \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}) - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Phi}(\mathbf{X}_t)^\top \left(\sigma^2 \mathbf{I} + \boldsymbol{\Phi}(\mathbf{X}_t)^\top \boldsymbol{\Phi}(\mathbf{X}_t) \right)^{-1} \boldsymbol{\Phi}(\mathbf{X}_t) \boldsymbol{\phi}(\mathbf{x}) \\ &= \hat{k}(\mathbf{x}, \mathbf{x}) - \hat{\mathbf{k}}_t(\mathbf{x})^\top \left(\hat{\mathbf{K}}_t + \sigma^2 \mathbf{I} \right)^{-1} \hat{\mathbf{k}}_t(\mathbf{x}),\end{aligned} \tag{B.4}$$

which gives the expression of the approximate posterior variance: $\hat{\sigma}_t^2(\mathbf{x}) = \sigma^2 \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\phi}(\mathbf{x})$. To conclude, Equations (B.3) and (B.4) prove that the expressions of the posterior mean and variance of GP with RFF approximation given in Section 5.2 (in the paragraph after Equation (5.2)) match the corresponding expressions of standard GP posterior mean and variance without RFF approximation (2.1), except that the original kernel values (i.e., $k(\mathbf{x}, \mathbf{x}')$) are replaced by the corresponding approximate kernel values (i.e., $\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$).

B.3 Proof of Theorem 5.1

As mentioned in Section 5.4, we analyze our FTS algorithm in the more general setting in which a message can be received from each agent $\mathcal{A}_n$ before every iteration t, instead of only before the first iteration. Therefore, throughout our theoretical analysis, we use $\boldsymbol{\omega}_{n,t}$, instead of $\boldsymbol{\omega}_n$, to denote the message received from agent $\mathcal{A}_n$ before iteration t. Similarly, we use $\hat{g}_{n,t}$, instead of $\hat{g}_n$, to denote the corresponding sampled function from agent $\mathcal{A}_n$ with RFF approximation in iteration t, obtained using $\boldsymbol{\omega}_{n,t}$: $\hat{g}_{n,t}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}, \forall \mathbf{x} \in \mathcal{X}$. Note that our theoretical analysis and results also hold in the most general setting where every agent $\mathcal{A}_n$ may collect more observations between different rounds of communication, in which the only difference is that every $t_n, \forall n = 1, \ldots, N$ may increase over different iterations.

Define $\mathcal{F}_t$ as the filtration containing agent $\mathcal{A}$'s history of selected inputs and observed outputs up to iteration t. Let $\delta \in (0, 1)$, we have defined in Theorem 5.1 that $\beta_t = B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)}$ and $c_t = \beta_t(1 + \sqrt{2\log(|\mathcal{X}|t^2)})$ for $t \geq 1$. Clearly, both β_t and c_t are increasing in t. Denote by A_t the event that agent $\mathcal{A}$ chooses $\mathbf{x}_t$ by maximizing a sampled function from its own GP posterior (i.e., $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$, as in line 4 of Algorithm 5.1), which happens with probability p_t; denote by B_t the event that $\mathcal{A}$ chooses $\mathbf{x}_t$ by maximizing the sampled function from any other agent $\mathcal{A}_1, \ldots, \mathcal{A}_N$ (line 6 of Algorithm 5.1), which happens with probability $(1 - p_t)$; denote by $B_{t,n}$ the event that $\mathcal{A}$ chooses $\mathbf{x}_t$ by maximizing the sampled function of agent $\mathcal{A}_n$ using RFF approximation (i.e., $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \hat{g}_{n,t}(\mathbf{x})$), which happens with probability $(1-p_t) \times P_N[n]$.

To begin with, we define two high-probability events through the following lemmas.

Lemma B.1. *Let $\delta \in (0, 1)$. Define $E^f(t)$ as the event that $|\mu_{t-1}(\mathbf{x}) - f(\mathbf{x})| \leq \beta_t\sigma_{t-1}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. We have that $\mathbb{P}\left[E^f(t)\right] \geq 1 - \delta/4$ for all $t \geq 1$.*

Lemma C.1 quantifies the concentration of the function f around its posterior mean and its proof follows directly from Theorem 2 of the work of (Chowdhury and Gopalan, 2017) by using an error probability of $\delta/4$.

Lemma B.2. *Define* $E^{f_t}(t)$ *as the event that* $|f_t(\mathbf{x}) - \mu_{t-1}(\mathbf{x})| \leq \beta_t \sqrt{2\log(|\mathcal{X}|t^2)}\sigma_{t-1}(\mathbf{x})$. *We have that* $\mathbb{P}\left[E^{f_t}(t)|\mathcal{F}_{t-1}\right] \geq 1 - 1/t^2$ *for any possible filtration* $\mathcal{F}_{t-1}$.

Lemma C.2 illustrates how concentrated a sampled function f_t from f is around its posterior mean and is a simpler version of Lemma 5 of the work of (Chowdhury and Gopalan, 2017). Specifically, we have assumed a discrete domain, whereas the work of (Chowdhury and Gopalan, 2017) deals with a compact domain. Note that both events $E^f(t)$ and $E^{f_t}(t)$ are $\mathcal{F}_{t-1}$-measurable.

Next, we define a set of inputs at every iteration t called *saturated points*, which represents the set of "bad" inputs at iteration t. These inputs are "bad" in the sense that the function values at these inputs have relatively large difference from the value of the global maximum of f. In the subsequent proof, we will lower-bound the probability that the selected input $\mathbf{x}_t$ is *unsaturated*, which will be a critical step in the proof.

Definition B.1. *Define the set of saturated points at iteration* t *as*

$$S_t = \{\mathbf{x} \in \mathcal{X} : \Delta(\mathbf{x}) > c_t\sigma_{t-1}(\mathbf{x})\}$$

in which $\Delta(\mathbf{x}) = f(\mathbf{x}^*) - f(\mathbf{x})$ *and* $\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.

Note that from this definition, $\mathbf{x}^*$ is always unsaturated since $\Delta(\mathbf{x}) = f(\mathbf{x}^*) - f(\mathbf{x}^*) = 0 < c_t\sigma_{t-1}(\mathbf{x}^*)$ for all $t \geq 1$. Also note that S_t is $\mathcal{F}_{t-1}$-measurable.

The next lemma bounds the deviation of the sampled function $\hat{g}_{n,t}(\mathbf{x})$ from the GP posterior of agent $\mathcal{A}_n$ with RFF approximation around its posterior mean $\hat{\mu}_{n,t}(\mathbf{x})$, whose proof is based on that of Lemma 11 of (Mutny and Krause, 2018).

Lemma B.3. *Given $\delta \in (0,1)$. We have that for all agents $\mathcal{A}_n, \forall n = 1, \ldots, N$ and for all $\mathbf{x} \in \mathcal{X}$ and all $t \geq 1$, with probability of at least $1 - \delta/4$*

$$|\hat{\mu}_{n,t}(\mathbf{x}) - \hat{g}_{n,t}(\mathbf{x})| \leq \sqrt{2 \log \frac{2\pi^2 t^2 N}{3\delta} + M}.$$

Proof. Recall from Section 5.2 that the sampled function $\hat{g}_{n,t}$ is obtained by firstly sampling $\boldsymbol{\omega}_{n,t} \sim \mathcal{N}(\boldsymbol{\nu}_{n,t}, \sigma^2 \boldsymbol{\Sigma}_{n,t}^{-1})$, and then setting $\hat{g}_{n,t}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}, \forall \mathbf{x} \in \mathcal{X}$. Moreover, we have shown in Section 5.2 that $\hat{\mu}_{n,t}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\nu}_{n,t}$. Denote $\boldsymbol{\omega}_{n,t} = \boldsymbol{\nu}_{n,t} + \sigma \boldsymbol{\Sigma}_{n,t}^{-1/2} \mathbf{z}$, in which $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is the $M \times 1$-dimensional standard Gaussian distribution. We have that

$$\begin{aligned}
|\hat{\mu}_{n,t}(\mathbf{x}) - \hat{g}_{n,t}(\mathbf{x})|^2 &= |\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\nu}_{n,t} - \boldsymbol{\phi}(\mathbf{x})^\top (\boldsymbol{\nu}_{n,t} + \sigma \boldsymbol{\Sigma}_{n,t}^{-1/2} \mathbf{z})|^2 \\
&= |\sigma \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_{n,t}^{-1/2} \mathbf{z}|^2 \\
&\leq \sigma^2 \left\| \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_{n,t}^{-1/2} \right\|_2^2 \|\mathbf{z}\|_2^2 \\
&= \sigma^2 \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\Sigma}_{n,t}^{-1} \boldsymbol{\phi}(\mathbf{x}) \|\mathbf{z}\|_2^2 \\
&= \hat{\sigma}_{n,t}^2(\mathbf{x}) \|\mathbf{z}\|_2^2 \leq \|\mathbf{z}\|_2^2 ,
\end{aligned} \tag{B.5}$$

in which we have made use of the assumption w.l.o.g. that the posterior variance is upper-bounded by 1 in the last inequality. Next, making use of the concentration of chi-squared distribution: $\mathbb{P}(\|\mathbf{z}\|_2^2 \geq M + 2\lambda) \leq \exp(-\lambda)$ (Mutny and Krause, 2018), we have that with probability of at least $1 - \frac{3\delta}{2\pi^2 t^2 N}$,

$$\|\mathbf{z}\|_2^2 \leq M + 2 \log \frac{2\pi^2 t^2 N}{3\delta}. \tag{B.6}$$

Taking a union bound over all agents $\mathcal{A}_1, \ldots, \mathcal{A}_N$ and all $t \geq 1$ completes the proof. □

The following lemma uniformly upper-bounds the difference between agent $\mathcal{A}_n$'s objective function g_n and sampled function $\hat{g}_{n,t}$ from its GP posterior with RFF approximation.

Lemma B.4. *Given any $\delta \in (0,1)$. Conditioned on the event $E^f(t)$, for agent $\mathcal{A}_n$'s sampled function $\hat{g}_{n,t}$ from its posterior GP with RFF approximation, we have that for all agents $\mathcal{A}_n, \forall n = 1,\ldots,N$ and for all $\mathbf{x} \in \mathcal{X}$ and all $t \geq 1$, with probability of at least $1-\delta/2$,*

$$|\hat{g}_{n,t}(\mathbf{x}) - g_n(\mathbf{x})| \leq \tilde{\Delta}_{n,t},$$

where $\beta'_t = B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(8N/\delta)}$, and

$$\tilde{\Delta}_{n,t} \triangleq \varepsilon\frac{(t_n+1)^2}{\sigma^2}\left(B + \sqrt{2\log\left(\frac{4\pi^2t^2N}{3\delta}\right)}\right) + \beta'_{t_n+1} + \sqrt{2\log\frac{2\pi^2t^2N}{3\delta} + M}.$$

Proof. Recall that ϵ is the accuracy of the RFF approximation, t_n is the number of iterations agent $\mathcal{A}_n$ has completed in its own BO task when it passes information to $\mathcal{A}$, M is the number of random features used in the RFF approximation. Denote by $\hat{\mu}_{n,t}(\mathbf{x})$ and $\mu_{n,t}(\mathbf{x})$ ($\hat{\sigma}_{n,t}(\mathbf{x})$ and $\sigma_{n,t}(\mathbf{x})$) the posterior mean (standard deviation) at $\mathbf{x}$ of agent $\mathcal{A}_n$'s GP after running its own BO task for t_n iterations with and without the RFF approximation respectively.

We have that for all $\mathbf{x} \in \mathcal{X}$, all agents $\mathcal{A}_n, \forall n = 1,\ldots,N$ and all $t \geq 1$, with probability of at least $1 - \frac{\delta}{8}$,

$$|\mu_{n,t}(\mathbf{x}) - \hat{\mu}_{n,t}(\mathbf{x})| \leq \varepsilon\frac{(t_n+1)^2}{\sigma^2}\left(B + \sqrt{2\log\left(\frac{4\pi^2t^2N}{3\delta}\right)}\right), \tag{B.7}$$

which can be proved by following the proof of Theorem 5 in the work of (Mutny and Krause, 2018) by substituting the error probability of $\frac{3\delta}{4\pi^2t^2N}$ and taking a union bound over all agents and all $t \geq 1$. Next, making use of Lemma C.1 (replacing f by g_n, and $\delta/4$ by $\delta/(8N)$), we get

$$|\mu_{n,t}(\mathbf{x}) - g_n(\mathbf{x})| \leq \beta'_{t_n+1}\sigma_{n,t}(\mathbf{x}) \leq \beta'_{t_n+1}, \tag{B.8}$$

which holds for all $\mathbf{x} \in \mathcal{X}$, agents $\mathcal{A}_n$ and $t_n \geq 1$, with probability of at least $1 - \delta/8$. The last inequality follows from our assumption w.l.o.g. that the posterior variance is upper-bounded by 1.

Combining the two equations above and making use of Lemma B.3 completes the proof. □

The next lemma shows a uniform upper bound on the difference between the sampled function f_t of agent $\mathcal{A}$ and that of agent $\mathcal{A}_n$ with RFF approximation $(\hat{g}_{n,t})$.

Lemma B.5. *At iteration t, conditioned on the events $E^f(t)$ and $E^{f_t}(t)$, we have that for all agents $\mathcal{A}_n, \forall n = 1, \ldots, N$ and for all $\mathbf{x} \in \mathcal{X}$ with probability $\geq 1 - \delta/2$*

$$|\hat{g}_{n,t}(\mathbf{x}) - f_t(\mathbf{x})| \leq \Delta_{n,t},$$

in which

$$\begin{aligned}\Delta_{n,t} \triangleq \varepsilon \frac{(t_n+1)^2}{\sigma^2} \left(B + \sqrt{2 \log \left(\frac{4\pi^2 t^2 N}{3\delta} \right)} \right) + \beta'_{t_n+1} + \\ \sqrt{2 \log \frac{2\pi^2 t^2 N}{3\delta} + M} + d_n + c_t.\end{aligned} \tag{B.9}$$

Proof. Firstly, note that since we condition on both events $E^f(t)$ and $E^{f_t}(t)$, we have that for all $\mathbf{x} \in \mathcal{X}$ and all $t \geq 1$

$$\begin{aligned}|f(\mathbf{x}) - f_t(\mathbf{x})| &\leq |f(\mathbf{x}) - \mu_{t-1}(\mathbf{x})| + |\mu_{t-1}(\mathbf{x}) - f_t(\mathbf{x})| \\ &= \beta_t \sigma_{t-1}(\mathbf{x}) + \beta_t \sqrt{2 \log(|\mathcal{X}| t^2)} \sigma_{t-1}(\mathbf{x}) = c_t \sigma_{t-1}(\mathbf{x})\end{aligned} \tag{B.10}$$

Next,

$$\begin{aligned}|\hat{g}_{n,t}(\mathbf{x}) - f_t(\mathbf{x})| &\leq |\hat{g}_{n,t}(\mathbf{x}) - g_n(\mathbf{x})| + |g_n(\mathbf{x}) - f(\mathbf{x})| + |f(\mathbf{x}) - f_t(\mathbf{x})| \\ &\leq \tilde{\Delta}_{n,t} + d_n + c_t\sigma_{t-1}(\mathbf{x}) \\ &\leq \tilde{\Delta}_{n,t} + d_n + c_t,\end{aligned} \tag{B.11}$$

in which we have made use of Lemma C.3, the definition of d_n: $d_n = \max_{\mathbf{x}\in\mathcal{X}} |f(\mathbf{x}) - g_n(\mathbf{x})|$ (Section 5.2, last paragraph), Equation (C.2), and the assumption that the posterior variance is upper-bounded by 1. Plugging in the expression of $\tilde{\Delta}_{n,t}$ from Lemma C.3 completes the proof. □

Lemma B.6. *For any filtration $\mathcal{F}_{t-1}$, conditioned on the events $E^f(t)$ and A_t, we have that for every $\mathbf{x} \in \mathcal{X}$,*

$$\mathbb{P}\left(f_t(\mathbf{x}) > f(\mathbf{x}) | \mathcal{F}_{t-1}, E^f(t), A_t\right) \geq p, \tag{B.12}$$

in which $p = \frac{1}{4e\sqrt{\pi}}$.

As shown in the proof of Lemma 8 of (Chowdhury and Gopalan, 2017), the proof of Lemma B.6 makes use of the fact that $f_t(\mathbf{x}) \sim \mathcal{N}(\mu_{t-1}(\mathbf{x}), \beta_t^2\sigma_{t-1}^2(\mathbf{x}))$ since we are conditioning on the event A_t, the confidence bound given in Lemma C.1 which holds since we are conditioning on the event $E^f(t)$, and the Gaussian anti-concentration lemma. That is, for a Gaussian random variable $X \sim \mathcal{N}(\mu, \sigma^2)$, for any $\beta > 0$, we have that

$$\mathbb{P}\left(\frac{X-\mu}{\sigma} > \beta\right) \geq \frac{\exp(-\beta^2)}{4\sqrt{\pi}\beta}.$$

The next lemma shows that in each iteration t, the probability that an unsaturated input is selected can be lower-bounded.

Lemma B.7. *For any filtration $\mathcal{F}_{t-1}$, conditioned on the event $E^f(t)$, we have that with probability $\geq 1-\delta/2$,*

$$\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}\right) \geq P_t,$$

in which

$$P_t \triangleq p_t(p - 1/t^2).$$

Proof. Note that all probabilities in this proof are conditioned on the event $E^f(t)$ and thus this conditioning is omitted for simplicity. At iteration t, the probability that the selected input $\mathbf{x}_t$ is unsaturated can be lower-bounded by:

$$\begin{aligned}\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}\right) &\geq \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) \mathbb{P}(A_t) \\ &= \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) p_t\end{aligned} \tag{B.13}$$

Next, we attempt to lower-bound $\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right)$.

Firstly, recall that conditioned on the event A_t, $\mathbf{x}_t$ is selected by maximizing f_t, which is sampled from the GP posterior of function f. This gives rise to:

$$\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) \geq \mathbb{P}\left(f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t | \mathcal{F}_{t-1}, A_t\right). \tag{B.14}$$

This inequality can be obtained by observing that the event on the right hand side is a subset of the event on the left hand side. Specifically, recall from Definition C.1 that $\mathbf{x}^*$ is always unsaturated. Therefore, if $f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t$, as a result of the way in which $\mathbf{x}_t$ is selected (i.e., $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$), this guarantees that an unsaturated input will be selected as $\mathbf{x}_t$ since at least one unsaturated input ($\mathbf{x}^*$) has a larger value of f_t than all saturated inputs.

Next, we assume that both events $E^f(t)$ and $E^{f_t}(t)$ are true, which allows us

to derive an upper bound on $f_t(\mathbf{x})$ for all $\mathbf{x} \in S_t$:

$$f_t(\mathbf{x}) \overset{(a)}{\leq} f(\mathbf{x}) + c_t\sigma_{t-1}(\mathbf{x}) \overset{(b)}{\leq} f(\mathbf{x}) + \Delta(\mathbf{x}) = f(\mathbf{x}) + f(\mathbf{x}^*) - f(\mathbf{x}) = f(\mathbf{x}^*), \tag{B.15}$$

in which (a) follows from (C.2) since here we also assume both events $E^f(t)$ and $E^{f_t}(t)$ are true, and (b) results from the definition of saturated set (Definition C.1). Therefore, (B.15) implies that

$$\begin{aligned} &\mathbb{P}\left(f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t | \mathcal{F}_{t-1}, A_t, E^{f_t}(t)\right) \\ &\geq \mathbb{P}\left(f_t(\mathbf{x}^*) > f(\mathbf{x}^*) | \mathcal{F}_{t-1}, A_t, E^{f_t}(t)\right). \end{aligned} \tag{B.16}$$

Next, we can show that

$$\begin{aligned} \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) &\geq \mathbb{P}\left(f_t(\mathbf{x}^*) > f_t(\mathbf{x}), \forall \mathbf{x} \in S_t | \mathcal{F}_{t-1}, A_t\right) \\ &\overset{(a)}{\geq} \mathbb{P}\left(f_t(\mathbf{x}^*) > f(\mathbf{x}^*) | \mathcal{F}_{t-1}, A_t\right) - \mathbb{P}\left(\overline{E^{f_t}(\mathbf{x})} | \mathcal{F}_{t-1}\right) \\ &\overset{(b)}{\geq} p - 1/t^2, \end{aligned} \tag{B.17}$$

in which (a) follows from some simple probabilistic manipulations and the fact that the event $E^{f_t}(t)$ is $\mathcal{F}_{t-1}$-measurable and thus independent of the event A_t, (b) results from Lemma B.6 and the fact that the event $E^{f_t}(t)$ holds with probability of at least $1 - 1/t^2$. Combining this inequality with (C.7) completes the proof.

□

The next lemma presents an upper bound on the expected instantaneous regret of the FTS algorithm.

Lemma B.8. *For any filtration $\mathcal{F}_{t-1}$, conditioned on the event $E^f(t)$, we have*

that with probability of $\geq 1-\delta/2$

$$\mathbb{E}[r_t|\mathcal{F}_{t-1}] \leq c_t\left(1+\frac{10}{pp_1}\right)\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2},$$

in which r_t *is the instantaneous regret:* $r_t = f(\mathbf{x}^*)-f(\mathbf{x}_t)$*, and* $\psi_t \triangleq 2(1-p_t)\sum_{n=1}^{N} P_N[n]\Delta_{n,t}$.

Proof. To begin with, we define $\overline{\mathbf{x}}_t$ as the unsaturated input at iteration t with the smallest posterior standard deviation:

$$\overline{\mathbf{x}}_t = \arg\min_{\mathbf{x}\in\mathcal{X}\setminus S_t}\sigma_{t-1}(\mathbf{x}). \tag{B.18}$$

Following this definition, for any $\mathcal{F}_{t-1}$ such that $E^f(t)$ is true, we have that

$$\begin{aligned}\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] &\geq \mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}, \mathbf{x}_t\in\mathcal{X}\setminus S_t\right]\mathbb{P}\left(\mathbf{x}_t\in\mathcal{X}\setminus S_t|\mathcal{F}_{t-1}\right)\\ &\geq \sigma_{t-1}(\overline{\mathbf{x}}_t)P_t,\end{aligned} \tag{B.19}$$

in which the last inequality follows from the definition of $\overline{\mathbf{x}}_t$ and Lemma C.5.

Now we condition on both events $E^f(t)$ and $E^{f_t}(t)$, and analyze the instantaneous regret as:

$$\begin{aligned}r_t = \Delta_t(\mathbf{x}_t) &= f(\mathbf{x}^*)-f(\overline{\mathbf{x}}_t)+f(\overline{\mathbf{x}}_t)-f(\mathbf{x}_t)\\ &\overset{(a)}{\leq} \Delta_t(\overline{\mathbf{x}}_t)+f_t(\overline{\mathbf{x}}_t)+c_t\sigma_{t-1}(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)+c_t\sigma_{t-1}(\mathbf{x}_t)\\ &\overset{(b)}{\leq} c_t\sigma_{t-1}(\overline{\mathbf{x}}_t)+c_t\sigma_{t-1}(\overline{\mathbf{x}}_t)+c_t\sigma_{t-1}(\mathbf{x}_t)+f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)\\ &= c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))+\underline{f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)},\end{aligned} \tag{B.20}$$

in which (a) follows from the definition of $\Delta_t(\mathbf{x})$ and $|f_t(\mathbf{x})-f(\mathbf{x})|\leq c_t\sigma_{t-1}(\mathbf{x})$ for all $\mathbf{x}\in\mathcal{X}$ since we assume both events $E^f(t)$ and $E^{f_t}(t)$ are true, and (b)

results from the fact that $\overline{\mathbf{x}}_t$ is unsaturated. Next, we analyze the expected value of the underlined term given a filtration $\mathcal{F}_{t-1}$:

$$\begin{aligned}
&\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] \\
&\quad = \mathbb{P}\left(A_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, A_t\right] + \\
&\qquad \mathbb{P}\left(B_t\right)\sum_{n=1}^{N} P_N[n]\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_{t,n}\right] \\
&\quad \overset{(a)}{\leq} (1-p_t)\sum_{n=1}^{N} P_N[n]\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_{t,n}\right] \\
&\quad \overset{(b)}{\leq} (1-p_t)\sum_{n=1}^{N} P_N[n]\mathbb{E}\left[\hat{g}_{n,t}(\overline{\mathbf{x}}_t) + \Delta_{n,t} - \hat{g}_{n,t}(\mathbf{x}_t) + \Delta_{n,t}|\mathcal{F}_{t-1}, B_{t,n}\right] \\
&\quad \overset{(c)}{\leq} 2(1-p_t)\sum_{n=1}^{N} P_N[n]\Delta_{n,t} \triangleq \psi_t,
\end{aligned} \tag{B.21}$$

in which (a) follows since when A_t is true, i.e., when $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$, $f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t) \leq 0$, (b) makes use of Lemma C.4 (note that here we are also conditioning on the events $E^f(t)$ and $E^{f_t}(t)$ which is the same as Lemma C.4, and that Lemma C.4 holds irrespective of the event $B_{t,n}$ since both E_t^f and $E^{f_t}(t)$ are $\mathcal{F}_{t-1}$-measurable) and thus holds with probability of $\geq 1-\delta/2$, and (c) follows since conditioned on the event $B_{t,n}$ (i.e., $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \hat{g}_{n,t}(\mathbf{x})$), $\hat{g}_{n,t}(\overline{\mathbf{x}}_t) - \hat{g}_{n,t}(\mathbf{x}_t) \leq 0$.

Subsequently, we can analyze the expected instantaneous regret by separately considering the two cases in which the event $E^{f_t}(t)$ is true and false respectively:

$$
\begin{aligned}
&\mathbb{E}\left[r_t|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))+f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+2B\mathbb{P}\left[\overline{E^{f_t}(t)}|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))|\mathcal{F}_{t-1}\right]+\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] \\
&\quad +2B\mathbb{P}\left[\overline{E^{f_t}(t)}|\mathcal{F}_{t-1}\right] \\
&\leq \frac{2c_t}{P_t}\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+c_t\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2} \\
&\leq c_t\left(1+\frac{2}{P_t}\right)\mathbb{E}\left[\sigma_{t-1}(x_t)|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2}.
\end{aligned}
\tag{B.22}
$$

Note that since $1/(p-1/t^2)\leq 5/p$ and $p_t\geq p_1$ for all $t\geq 1$,

$$
\frac{2}{P_t}=\frac{2}{p_t(p-\frac{1}{t^2})}\leq\frac{10}{pp_t}\leq\frac{10}{pp_1}. \tag{B.23}
$$

Therefore, (C.19) can be further analyzed as

$$
\mathbb{E}\left[r_t|\mathcal{F}_{t-1}\right]\leq c_t\left(1+\frac{10}{pp_1}\right)\mathbb{E}\left[\sigma_{t-1}(x_t)|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2}, \tag{B.24}
$$

which completes the proof. □

Subsequently, we make use of the concentration inequality of super-martingales to derive a bound on the cumulative regret.

Definition B.2. *Define $Y_0=0$, and for all $t=1,\ldots,T$,*

$$
\overline{r}_t=r_t\mathbb{I}\{E^f(t)\},
$$

$$
X_t=\overline{r}_t-c_t\left(1+\frac{10}{pp_1}\right)\sigma_{t-1}(\mathbf{x}_t)-\psi_t-\frac{2B}{t^2},
$$

$$
Y_t=\sum_{s=1}^{t}X_s.
$$

Lemma B.9. *Conditioned on Lemma C.6 (i.e., with probability of* $\geq 1-\delta/2$*),* $(Y_t : t = 0, \ldots, T)$ *is a super-martingale with respect to the filtration* $\mathcal{F}_t$*.*

Proof.

$$\begin{aligned}
\mathbb{E}[Y_t - Y_{t-1}|\mathcal{F}_{t-1}] &= \mathbb{E}\left[X_t|\mathcal{F}_{t-1}\right] \\
&= \mathbb{E}\left[\overline{r}_t - c_t\left(1+\frac{10}{pp_1}\right)\sigma_{t-1}(x_t) - \psi_t - \frac{2B}{t^2}|\mathcal{F}_{t-1}\right] \\
&= \mathbb{E}\left[\overline{r}_t|\mathcal{F}_{t-1}\right] - \left[c_t\left(1+\frac{10}{pp_1}\right)\mathbb{E}\left[\sigma_{t-1}(x_t)|\mathcal{F}_{t-1}\right] + \psi_t + \frac{2B}{t^2}\right] \\
&\leq 0,
\end{aligned} \tag{B.25}$$

in which the last inequality follows from Lemma C.6 when the event $E^f(t)$ is true; when $E^f(t)$ is false, $\overline{r}_t = 0$ and thus the inequality holds trivially. □

The Azuma-Hoeffding Inequality presented below will be useful for proving the concentration of the super-martingale $(Y_t : t = 0, \ldots, T)$.

Lemma B.10 (Azuma-Hoeffding Inequality)**.** *Given any* $\delta' \in (0,1)$*. If a super-martingale* $(Z_T : t = 1, \ldots, T)$*, defined with respect to the filtration* $\mathcal{F}_t$*, satisfies* $|Z_t - Z_{t-1}| \leq \alpha_t$ *for some constant* α_t*, then for all* $t = 1, \ldots, T$ *and with probability of at least* $1-\delta'$*,*

$$Z_T - Z_0 \leq \sqrt{2\log(1/\delta')\sum_{t=1}^{T}\alpha_t^2}.$$

Finally, we can derive an upper bound on the cumulative regret through the following lemma.

Lemma B.11. *Given* $\delta \in (0,1)$*, then with probability of at least* $1-\delta$*,*

$$\begin{aligned}
R_T \leq & c_T\left(1+\frac{10}{pp_1}\right)\mathcal{O}(\sqrt{T\gamma_T}) + \sum_{t=1}^{T}\psi_t + \frac{B\pi^2}{3} + \\
& \left[c_T\left(1+\frac{4B}{p}+\frac{10}{pp_1}\right) + \psi_1 + \mathcal{O}(\sqrt{\log T})\right]\sqrt{2T\log\frac{4}{\delta}},
\end{aligned}$$

in which γ_T is the maximum information gain about f obtained from any set of T observations.

Proof.

$$
\begin{aligned}
|Y_t - Y_{t-1}| = |X_t| &\leq |\overline{r}_t| + c_t \left(1 + \frac{10}{pp_1}\right) \sigma_{t-1}(\mathbf{x}_t) + \psi_t + \frac{2B}{t^2} \\
&\overset{(a)}{\leq} 2B + c_t \left(1 + \frac{10}{pp_1}\right) + \psi_t + \frac{2B}{t^2} \\
&\overset{(b)}{\leq} \frac{2Bc_t}{p} + c_t \left(1 + \frac{10}{pp_1}\right) + \psi_t + \frac{2Bc_t}{p} \\
&\leq c_t \left(1 + \frac{4B}{p} + \frac{10}{pp_1}\right) + \psi_t,
\end{aligned}
\tag{B.26}
$$

in which (a) follows since the posterior variance is upper-bounded by 1, (b) follows since $2B \leq 2Bc_t/p$ and $2B/t^2 \leq 2Bc_t/p$.

This allows us to apply the Azuma-Hoeffding Inequality (Lemma B.10) by using an error probability of $\delta/4$,

$$
\begin{aligned}
\sum_{t=1}^{T} \overline{r}_t &\leq \sum_{t=1}^{T} c_t \left(1 + \frac{10}{pp_1}\right) \sigma_{t-1}(x_t) + \sum_{t=1}^{T} \psi_t + \sum_{t=1}^{T} \frac{2B}{t^2} + \\
&\qquad \sqrt{2 \log \frac{4}{\delta} \sum_{t=1}^{T} \left[c_t \left(1 + \frac{4B}{p} + \frac{10}{pp_1}\right) + \psi_t \right]^2} \\
&\leq c_T \left(1 + \frac{10}{pp_1}\right) \sum_{t=1}^{T} \sigma_{t-1}(x_t) + \sum_{t=1}^{T} \psi_t + \frac{B\pi^2}{3} + \\
&\qquad \left[c_T \left(1 + \frac{4B}{p} + \frac{10}{pp_1}\right) + \psi_1 + \mathcal{O}(\sqrt{\log T}) \right] \sqrt{2T \log \frac{4}{\delta}} \\
&= c_T \left(1 + \frac{10}{pp_1}\right) \mathcal{O}(\sqrt{T\gamma_T}) + \sum_{t=1}^{T} \psi_t + \frac{B\pi^2}{3} + \\
&\qquad \left[c_T \left(1 + \frac{4B}{p} + \frac{10}{pp_1}\right) + \psi_1 + \mathcal{O}(\sqrt{\log T}) \right] \sqrt{2T \log \frac{4}{\delta}},
\end{aligned}
\tag{B.27}
$$

which holds with probability $\geq 1 - \delta/4$. The last inequality follows since c_t is increasing in t, $\sum_{t=1}^{T} 1/t^2 = \pi^2/6$, and $\psi_t \leq \psi_1 + \mathcal{O}(\sqrt{\log t}), \forall t \geq 1$ which is ensured by the way in which we choose the sequence p_t, i.e., such

that $(1-p_t)c_t \leq (1-p_1)c_1$ for all $t \geq 2$. Lastly, note that the event $E^f(t)$ holds with probability $\geq 1-\delta/4$, i.e., $\overline{r}_t = r_t$ with probability $\geq 1-\delta/4$. In the last equality, we made use of the fact that $\sum_{t=1}^{T} \sigma_{t-1}(\mathbf{x}_t) = \sqrt{C_1 T \gamma_T}$ ($C_1 \triangleq 8/(1+\sigma^{-2})$) which is proved by Lemmas 5.3 and 5.4 of (Srinivas et al., 2010). Taking into account the error probability of Lemma C.6 ($\delta/2$), which is required for $(Y_t : t = 0, \ldots, T)$ to form a super-martingale, completes the proof. □

Finally, we are ready to prove Theorem 5.1. Recall that

$$c_t = \mathcal{O}\left(\left(B + \sqrt{\gamma_t + \log(1/\delta)}\right)\sqrt{\log t}\right).$$

Therefore,

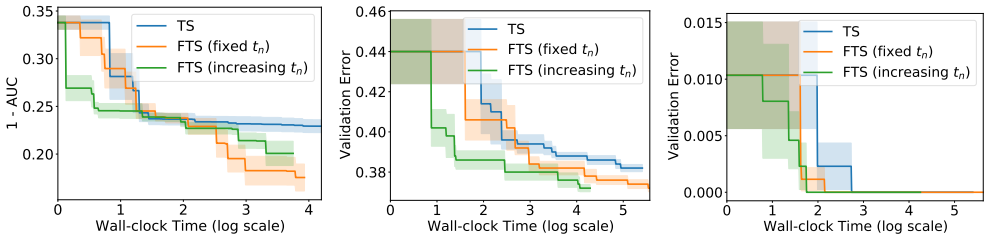
$$\begin{aligned} R_T &= \mathcal{O}\left(\frac{1}{p_1}\left(B + \sqrt{\gamma_T + \log\frac{1}{\delta}}\right)\sqrt{\log T}\sqrt{T\gamma_T} + \sum_{t=1}^{T}\psi_t + \right. \\ &\qquad \left.\left(B + \frac{1}{p_1}\right)\left(B + \sqrt{\gamma_T + \log\frac{1}{\delta}}\right)\sqrt{\log T}\sqrt{T\log\frac{1}{\delta}}\right) \\ &= \mathcal{O}\left(\left(B + \frac{1}{p_1}\right)\sqrt{T\log T\gamma_T\log\frac{1}{\delta}\left(\gamma_T + \log\frac{1}{\delta}\right)} + \sum_{t=1}^{T}\psi_t\right) \\ &= \tilde{\mathcal{O}}\left(\left(B + \frac{1}{p_1}\right)\gamma_T\sqrt{T} + \sum_{t=1}^{T}\psi_t\right). \end{aligned} \tag{B.28}$$

B.4 Further Experimental Details and Results

All experiments reported in this work are run on a computer with 48 cores of Xeon Silver 4116 (2.1Ghz) processors, RAM of 256GB, and 1 NVIDIA Tesla T4 GPU. For fair comparisons, in all experiments, the same random initializations are used by all methods.

B.4.1 Optimization of Synthetic Functions

In the synthetic experiments, we sample the objective functions from a GP with a length scale of 0.03. The functions are defined on a 1-dimensional discrete domain uniformly distributed in $[0, 1]$, with size $|\mathcal{X}| = 1,000$. The output values of all functions $f(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}$ are normalized into the range of $[0, 1]$. Whenever an input x is queried, the corresponding noisy observation is obtained by adding a zero-mean Gaussian noise $\mathcal{N}(0, \sigma^2)$ where $\sigma^2 = 0.01$ to the corresponding function value $f(\mathbf{x})$ (Section 5.2, first paragraph). For a sampled objective function for the target agent, we generate the objective functions of the other agents, as well as their observations, in the following way. For agent $\mathcal{A}_n$, we go through every input in the entire discrete domain, and for each input, we add either d_n or $-d_n$ to the corresponding output function value with probability 0.5 each, after which the resulting value is used as the objective function value of the agent $\mathcal{A}_n$. This step ensures the validity of the definition of d_n as the maximum difference between the objective function of the target agent $\mathcal{A}$ and that of agent $\mathcal{A}_n$: $d_n = \max_{\mathbf{x} \in \mathcal{X}} |f(\mathbf{x}) - g_n(\mathbf{x})|$ (Section 5.2, last paragraph). Next, we randomly sample t_n inputs from the entire discrete domain, and for each sampled input, we obtain a noisy output observation by adding a zero-mean Gaussian noise: $\mathcal{N}(0, \sigma^2)$ where $\sigma^2 = 0.01$, to the corresponding function value. Subsequently, following the procedures described in the first paragraph of Section 5.3.1, every agent $\mathcal{A}_n$ applies RFF approximation using its own t_n observations (input-output pairs) to derive the RFF approximation parameters $\boldsymbol{\nu}_n$ and $\boldsymbol{\Sigma}_n$ and hence to draw a sample of $\boldsymbol{\omega}_n$, which is the parameter to be passed to and used by the target agent $\mathcal{A}$. Finally, after receiving the parameters $\boldsymbol{\omega}_n$'s from all other agents, the target agent starts to run the FTS algorithm (Algorithm 5.1).



(a) (b) (c)

Figure B.1: Performances in the most general setting in which t_n is increasing (green curve) for the (a) landmine detection, (b) Google glasses and (c) mobile phone sensors experiments. The specific experimental setup is described in Appendix B.4.2.1. The results correspond to Fig. 5.3 in the main paper.

B.4.2 Real-world Experiments

B.4.2.1 Results in the Most General Setting with Increasing t_n

Here we perform additional experiments in the most general setting of our FTS algorithm (Section 5.3.1, last paragraph): (a) information can be received from every agent $\mathcal{A}_n$ before every iteration instead of only before the first iteration, and (b) every $\mathcal{A}_n$ may also be performing black-box optimization tasks (possibly also using FTS), such that $\mathcal{A}_n$ may collect more observations (i.e., t_n may increase) between different rounds of communication. We use the three real-world experiments (Section 5.5.2) to investigate the performances in this setting, and compare the performances with those in the simpler setting where communication is allowed only before the first iteration.

Here we describe the detailed experimental setup for the experiments in this section. Before the first iteration of the FTS algorithm, every agent $\mathcal{A}_n$ for $n = 1, \dots, N$, who has completed $t_n = 50$ iterations of its own BO task (we use standard TS here for simplicity, but it can be replaced by FTS in which $\mathcal{A}_n$ is the target agent), passes the first message to the target agent $\mathcal{A}$. Next, before every iteration $t > 1$ of the FTS algorithm (Algorithm 5.1), every agent $\mathcal{A}_n$ runs *one more iteration* of its own BO task, calculates the updated RFF approximation parameters $\boldsymbol{\nu}_{n,t}$ and $\boldsymbol{\Sigma}_{n,t}$ (5.2), samples a new $\boldsymbol{\omega}_{n,t}$ from its posterior distribution:

$\boldsymbol{\omega}_{n,t} \sim \mathcal{N}(\boldsymbol{\nu}_{n,t}, \sigma^2 \boldsymbol{\Sigma}_{n,t}^{-1})$, and finally passes the sampled $\boldsymbol{\omega}_{n,t}$ to the target agent $\mathcal{A}$. Then, $\mathcal{A}$ uses the received updated information to run iteration t of the FTS algorithm. After this, the information from every agent $\mathcal{A}_n$ is updated and sent to $\mathcal{A}$ again, and the FTS algorithm proceeds to the next iteration $t+1$. As a result, every $t_n, \forall n = 1, \ldots, N$ increases by 1 after every iteration of FTS.

The performances in all three experiments are shown in Fig. B.1, in which FTS outperforms standard TS in both settings for all experiments. The figure also shows that in the most general setting in which t_n is increasing, the performances for the two activity recognition experiments (Google glasses and mobile phone sensors experiments) are improved, whereas the performances for the landmine detection experiment are comparable in both settings. Note that the most general setting with increasing t_n may not necessarily lead to better performances: Although using more observations from those agents with similar objective functions to the target agent can give more useful information and hence potentially benefit the FTS algorithm, more observations from *heterogeneous agents* may turn out to hurt the performance of FTS since the information from these agents are actually harmful for the BO task of the target agent.

B.4.2.2 More Experimental Details

In all real-world experiments, we use length scale $= 0.01$ to generate the random features (Appendix B.1) and $\sigma^2 = 10^{-6}$ in the RFF approximation using equations (5.1) and (5.2).

Landmine Detection. This dataset[1] consists of the data from 29 landmine fields, with each field associated with a dataset for landmine detection. The dataset of each landmine field is made up of a number of input-output pairs, each corresponding to a location; for every location, the input includes 9 features extracted from radar images and the output is a binary label indicating whether

[1] http://www.ee.duke.edu/~lcarin/LandmineData.zip

the location contains landmines. The number of data points (input-output pairs) of every field ranges from 445 to 690, with a mean of 511; for every field, we use 50% of the data points as the training set, and the other 50% as the validation set. We use support vector machines (SVM) as the predictive model, and tune two SVM hyperparameters: RBF kernel parameter in the range of $[0.01, 10]$, and L2 regularization parameter in $[10^{-4}, 10]$. For every queried hyperparamter setting, the SVM model is trained on the training set using this particular set of hyperparameters, and evaluated using the validation set to produce the reported performances. (Figs. 5.2 and 5.3 in the main text). As mentioned in the main text, the dataset of the landmine fields are significantly imbalanced, i.e., there are considerably more locations without than with landmines. Specifically, the percentage of positive samples (i.e., locations with landmines) in different landmine fields ranges from 2.9% to 9.4%, with a mean of 6.2%. Therefore, for this dataset, validation error is inappropriate since an all-zero prediction would result in very low classification error. Hence, we use the Area Under the Receiver Operating Characteristic Curve (AUC) which is a more appropriate metric when evaluating the performance of ML models on imbalanced datasets.

Activity Recognition Using Google Glasses. This dataset[2] consists of two-hour long sensor data collected using Google glasses from 38 participants, while the participants are performing different activities such as eating. For every participant, we group the sensor data into different time windows; for each time window, we calculate the statistics (i.e., mean, variance and kurtosis) of different sensor measurements within this time window, and use them as the features (57 features in total are extracted from each time window); the label for each time window is a binary value indicating whether the participant is eating or conducting other activities during this time window. As a result, for every participant, each time window produces a data point, i.e., an input-output pair. The number of data

[2] http://www.skleinberg.org/data/GLEAM.tar.gz

points for every participant ranges from 242 to 3416 with an average of 1930. For every participant, we randomly select 100 data points as the validation set, and use the remaining data points as the training set. We use logistic regression (LR) as the activity prediction model for every participant, and tune 3 hyperparameters of LR: the batch size in the range of $[20, 60]$, the L2 regularization parameter in $[10^{-6}, 1]$, and the learning rate in $[0.01, 0.1]$. Following the common practice for using LR and neural network models, the inputs are pre-processed by removing the mean and dividing by the standard deviation.

Activity Recognition Using Mobile Phone Sensors. This dataset[3] contains measurements from mobile phone sensors (accelerometer and gyroscope) involving 30 subjects. 561 features were provided together with the dataset, with each set of features associated with a corresponding label indicating which one of the six activities the subject is performing. Therefore, the activity recognition problem for every subject corresponds to a 6-class classification problem. The number of data points (input-output pairs) possessed by the subjects ranges from 281 to 409 with a mean of 343. For every subject, we use 50% of the data points as the training set, and the remaining 50% as the validation set. We again use LR as the activity recognition model, and the tuned hyperparameters, as well as their ranges, are the same as those in the activity recognition experiment using Google glasses.

B.4.2.3 Additional Results for More Agents

In this section, we present additional experimental results for the three real-world experiments (Section 5.5.2). Note that as mentioned in Section 5.5.2 (last paragraph), the results presented in Fig. 5.3 in the main text correspond to using the first agent (of the 6 agents used to produce the results in Fig. 5.2 in the main text) as the target agent for every experiment. Meanwhile, the additional results

[3] https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

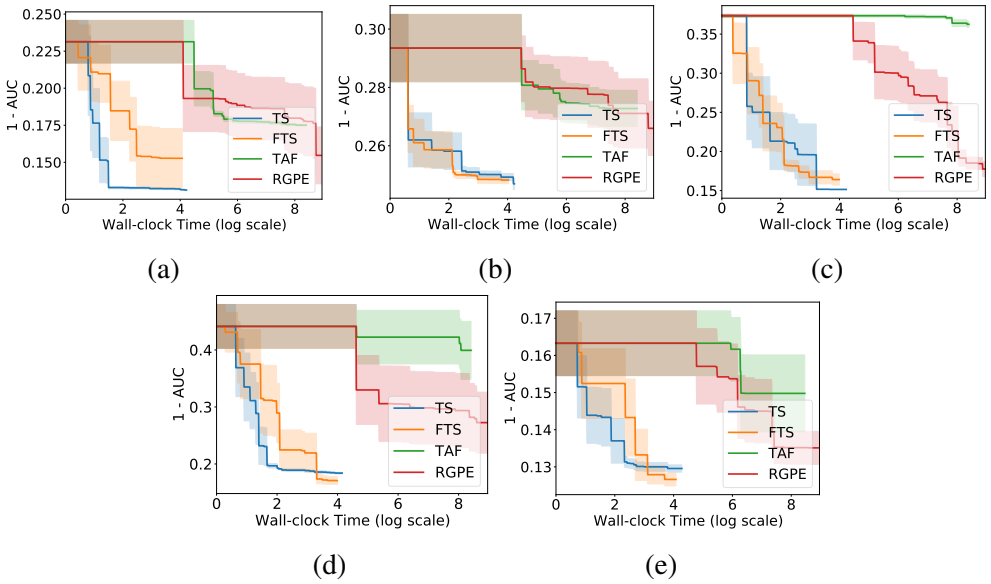


Figure B.2: Additional results for the other 5 target agents for the landmine detection experiment ($M = 100$).

shown in this section (Figs. B.2, B.3 and B.4) correspond to using each of the remaining 5 agents (agents 2 to 6) as the target agent. Note that since all three real-world datasets contain heterogeneous agents (Section 5.5.2, first paragraph), it is unreasonable to expect FTS to always outperform standard TS for all agents. Instead, as shown in the figures, FTS performs better than TS for some agents, and comparably with TS for other agents.

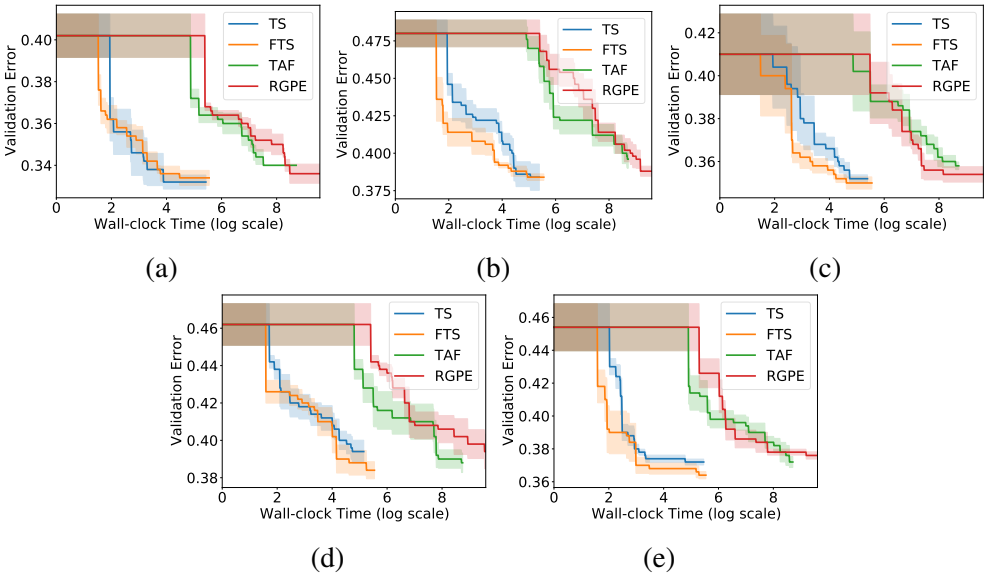


Figure B.3: Additional results for the other 5 target agents for the activity recognition experiment using Google glasses ($M = 100$).

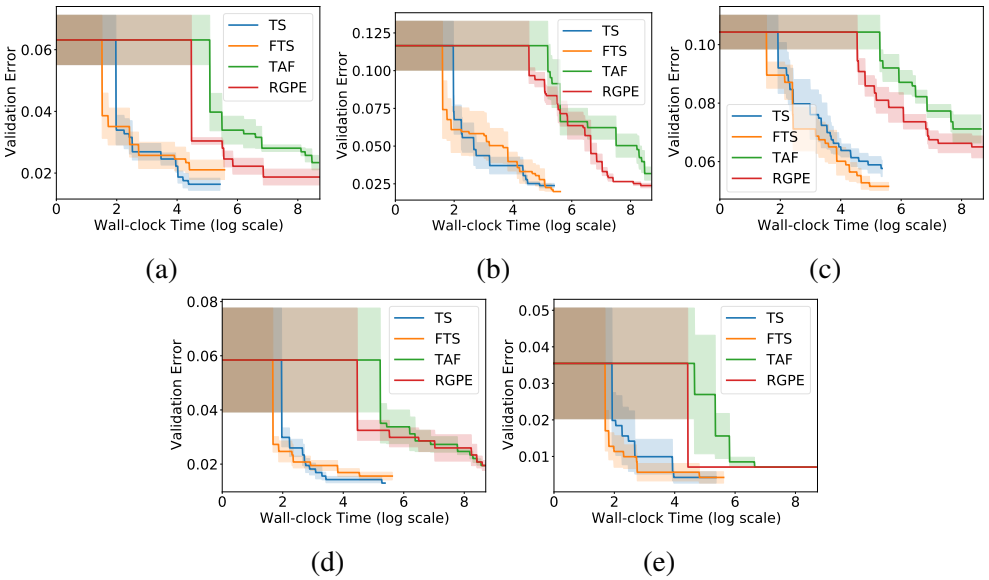


Figure B.4: Additional results for the other 5 target agents for the activity recognition experiment using mobile phone sensors ($M = 100$).

Appendix C

Appendix for Chapter 6

C.1 Proof of Theoretical Results

C.1.1 Proof of Proposition 6.1

Proposition 6.1 follows directly from the DP guarantee of the works of Abadi et al. (2016) and McMahan et al. (2018b) (e.g., Theorem 1 of Abadi et al. (2016)). Therefore, to prove the validity of Proposition 6.1, we only need to show that the joint subsampled Gaussian mechanism we apply in every iteration (Section 6.3.3) is the same as the one adopted by Abadi et al. (2016) and McMahan et al. (2018b). Therefore, we demonstrate here that the interpretation of our privacy-preserving transformations as a single subsampled Gaussian mechanism, which we have described in Section 6.3.3, is equivalent to the transformations adopted by the work of McMahan et al. (2018b).

Firstly, our subsampling step (step 6 of Algorithm 6.1) is the same as the one adopted by McMahan et al. (2018b) since we both use the same subsampling technique, i.e., select every agent with a fixed probability q. Secondly, we both clip the (joint) vector from every selected agent (step 9 of Algorithm 6.1) to ensure that its L_2 norm is upper-bounded: $\left\|\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}\right\|_2 \leq N\varphi_{\max}S, \forall n \in \mathcal{S}_t$. Thirdly, we have adopted one of the two weighted-average estimators proposed

by McMahan et al. (2018b), i.e., the unbiased estimator. Specifically, we set the weight (we follow McMahan et al. (2018b) and denote the weight of agent $\mathcal{A}_n$ by ω_n here) of every agent to be $\omega_n = 1, \forall n \in [N]$. As a result, the unbiased estimator leads to: $\boldsymbol{\omega}_t^{\text{joint}} = \frac{1}{q\sum_{n=1}^{N}\omega_n}\sum_{n\in\mathcal{S}_t}\omega_n\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}} = \frac{1}{qN}\sum_{n\in\mathcal{S}_t}\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}$. Lastly, we have calculated the sensitivity (which determines the variance of the Gaussian noise) in the same way as McMahan et al. (2018b), i.e., using Lemma 1 of McMahan et al. (2018b). In particular, note that our clipping step has ensured that $\left\|\omega_n\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}\right\|_2 \leq N\varphi_{\max}S, \forall n \in \mathcal{S}_t$; according to Lemma 1 of McMahan et al. (2018b), we have that the sensitivity can be upper-bounded by: $\mathbb{S} \leq \frac{N\varphi_{\max}S}{q\sum_{n=1}^{N}\omega_n} = \varphi_{\max}S/q$, which leads to the standard deviation of the Gaussian noise we have added (step 11 of Algorithm 6.1): $z\mathbb{S} = z\varphi_{\max}S/q$.

To conclude, the single joint subsampled Gaussian mechanism performed by our DP-FTS-DE algorithm in every iteration is the same as the one adopted by McMahan et al. (2018b). Therefore, the DP guarantee of McMahan et al. (2018b) and Abadi et al. (2016) is also valid for our DP-FTS-DE algorithm, hence justifying the validity of our Proposition 6.1.

C.1.2 Proof of Theorem 6.1

In this section, we prove Theorem 6.1, which gives an upper bound on the cumulative regret of agent $\mathcal{A}_1$ running our DP-FTS-DE algorithm. The proof of Theorem 6.1 makes use of the proof of the FTS algorithm (Chapter 5 and Appendix B.3) (Dai et al., 2020b), and the main technical challenge is how to take into account the impacts of (a) our modification to the original FTS algorithm by incorporating a central server and an aggregation through weighted averaging (first paragraph of Section 6.3.1), (b) the privacy-preserving transformations (lines 5-11 of Algorithm 6.1), and (c) distributed exploration (DE) (Section 6.3.2). Since we are mainly interested in the asymptotic regret upper bound, we ignore the impact of the initialization period. Considering initialization would only add a

constant term $2BN_{\text{init}}$ to the upper bound on the cumulative regret in Theorem 6.1 (N_{init} is the number of initial inputs selected during initialization), and hence would not affect the asymptotic no-regret property of our algorithm.

Note that as we have mentioned in Sections 6.2 and 6.4.2, we prove here an upper bound on the cumulative regret of agent $\mathcal{A}_1$, i.e., $R_T^1 \triangleq \sum_{t=1}^{T}(f^1(\mathbf{x}^{1,*}) - f^1(\mathbf{x}_t^1))$. To simplify notations, we drop the superscript 1 in the subsequent analysis, i.e., we use f to denote f^1, f_t to denote f_t^1, $\mathbf{x}_t$ to denote $\mathbf{x}_t^1$, $\mathbf{x}^*$ to denote $\mathbf{x}^{1,*}$, etc. Similarly, we use μ_{t-1} and σ_{t-1} to represent the GP posterior mean and standard deviation of $\mathcal{A}_1$ at iteration t.

C.1.2.1 Definitions and Supporting Lemmas

We firstly define some notations we use to represent the privacy-preserving transformations, which are consistent with those in the main text. In iteration t, we use $\boldsymbol{\omega}_{n,t}$ to denote the vector the central server receives from agent $\mathcal{A}_n$. For a given set of agents $\mathcal{C} \in \{1, \ldots, N\}$, define $\widetilde{\varphi}_{\mathcal{C}}^{(i)} \triangleq \sum_{n\in\mathcal{C}} \varphi_n^{(i)}$, i.e., the total weight of those agents in the set $\mathcal{C}$ for the sub-region $\mathcal{X}_i$. Next, we define N indicator (Bernoulli) random variables $\mathbb{I}_n, \forall n = 1, \ldots, N$, where $\mathbb{P}(\mathbb{I}_n = 1) = q, \forall n = 1, \ldots, N$. These indicator random variables will be used to account for the subsampling step (i.e., step 6 of Algorithm 6.1). Denote by $\widehat{\boldsymbol{\omega}}_{n,t}$ the resulting vector after $\boldsymbol{\omega}_{n,t}$ is clipped to have a maximum L_2 norm of $S/\sqrt{P}$ (i.e., step 9 of Algorithm 6.1):

$$\widehat{\boldsymbol{\omega}}_{n,t} \triangleq \frac{\boldsymbol{\omega}_{n,t}}{\max(1, \frac{\|\boldsymbol{\omega}_{n,t}\|_2}{S/\sqrt{P}})}.$$

An immediate consequence is that $\forall \mathbf{x} \in \mathcal{X}$:

$$\begin{aligned} |\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}| &= |\boldsymbol{\phi}(\mathbf{x})^\top \frac{\boldsymbol{\omega}_{n,t}}{\max(1, \frac{\|\boldsymbol{\omega}_{n,t}\|_2}{S/\sqrt{P}})}| \\ &= |\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}| \frac{1}{\max(1, \frac{\|\boldsymbol{\omega}_{n,t}\|_2}{S/\sqrt{P}})} \leq |\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}|. \end{aligned} \tag{C.1}$$

Denote by $\boldsymbol{\eta}$ the added Gaussian noise vector (i.e., step 11 of Algorithm 6.1): $\boldsymbol{\eta} \sim \mathcal{N}\left(\mathbf{0}, (z\varphi_{\max} S/q)^2 \mathbf{I}\right)$. Next, define

$$\boldsymbol{\omega}_t^{(i)} \triangleq \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q} + \boldsymbol{\eta}.$$

As a result, for agent $\mathcal{A}_1$ at iteration $t > 1$, with probability $1 - p_t$, the query $\mathbf{x}_t^1$ is selected using the $\boldsymbol{\omega}_t^{(i)}$'s: $\mathbf{x}_t^1 = \arg\max_{\mathbf{x} \in \mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i^{[\mathbf{x}]})}$, where $i^{[\mathbf{x}]}$ represents the sub-region $\mathbf{x}$ is assigned to. This corresponds to line 7 of Algorithm 6.2.

Similar to the proof of the FTS algorithm (Appendix B.3), we let $\delta \in (0, 1)$, and define $\beta_t \triangleq B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)}$ and $c_t \triangleq \beta_t(1 + \sqrt{2\log(|\mathcal{X}|t^2)})$ for all $t \in \mathbb{Z}^+$. Denote by A_t the event that agent $\mathcal{A}_1$ chooses $\mathbf{x}_t^1$ by maximizing a sampled function from its own GP posterior belief (i.e., $\mathbf{x}_t^1 = \arg\max_{\mathbf{x} \in \mathcal{X}} f_t^1(\mathbf{x})$, as in line 5 of Algorithm 6.2), which happens with probability p_t; denote by B_t the event that $\mathcal{A}_1$ chooses $\mathbf{x}_t^1$ using the information received from the central server: $\mathbf{x}_t^1 = \arg\max_{\mathbf{x} \in \mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i^{[\mathbf{x}]})}$ (line 7 of Algorithm 6.2), which happens with probability $(1 - p_t)$.

Next, we denote as $\mathcal{F}_t$ the filtration which includes the history of selected inputs and observed outputs of agent $\mathcal{A}_1$ until (including) iteration t. Same as the proof of FTS (Appendix B.3), we need to make use of the following two $\mathcal{F}_{t-1}$-measurable events, which we state here again for completeness:

Lemma C.1 (Lemma 1 of Dai et al. (2020b))**.** *Let $\delta \in (0, 1)$. Define $E^f(t)$ as the event that $|\mu_{t-1}(\mathbf{x}) - f(\mathbf{x})| \leq \beta_t \sigma_{t-1}(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{X}$. We have that $\mathbb{P}\left[E^f(t)\right] \geq 1 - \delta/4$ for all $t \geq 1$.*

Lemma C.2 (Lemma 2 of Dai et al. (2020b)). *Define* $E^{f_t}(t)$ *as the event that* $|f_t(\mathbf{x}) - \mu_{t-1}(\mathbf{x})| \leq \beta_t \sqrt{2\log(|\mathcal{X}|t^2)}\sigma_{t-1}(\mathbf{x})$. *We have that* $\mathbb{P}\left[E^{f_t}(t)|\mathcal{F}_{t-1}\right] \geq 1 - 1/t^2$ *for any possible filtration* $\mathcal{F}_{t-1}$.

Note that conditioned on both events $E^f(t)$ and $E^{f_t}(t)$, we have that for all $x \in \mathcal{X}$ and all $t \geq 1$:

$$\begin{aligned} |f(\mathbf{x}) - f_t(\mathbf{x})| &\leq |f(\mathbf{x}) - \mu_{t-1}(\mathbf{x})| + |\mu_{t-1}(\mathbf{x}) - f_t(\mathbf{x})| \\ &= \beta_t \sigma_{t-1}(\mathbf{x}) + \beta_t \sqrt{2\log(|\mathcal{X}|t^2)}\sigma_{t-1}(\mathbf{x}) = c_t \sigma_{t-1}(\mathbf{x}). \end{aligned} \tag{C.2}$$

Next, again following the proof in Appendix B.3, at every iteration t, we define a set of *saturated points*, i.e., the set of "bad" inputs at iteration t. Intuitively, these inputs are considered as "bad" because their corresponding function values have relatively large differences from the value of the global maximum of f.

Definition C.1. *At iteration* t, *define the set of saturated points as*

$$S_t = \{\mathbf{x} \in \mathcal{X} : \Delta(\mathbf{x}) > c_t \sigma_{t-1}(\mathbf{x})\}$$

in which $\Delta(\mathbf{x}) \triangleq f(\mathbf{x}^*) - f(\mathbf{x})$ *and* $\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.

Note that $\Delta(\mathbf{x}^*) \triangleq f(\mathbf{x}^*) - f(\mathbf{x}^*) = 0 < c_t \sigma_{t-1}(\mathbf{x}^*)$ for all $t \geq 1$. Therefore, $\mathbf{x}^*$ is always unsaturated for all $t \geq 1$. S_t is $\mathcal{F}_{t-1}$-measurable.

Consistent with the main text in Chapter 6, we define $\widetilde{f}_t^n(\mathbf{x}) \triangleq \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}, \forall \mathbf{x} \in \mathcal{X}$, i.e., $\widetilde{f}_t^n$ is the sampled function from agent $\mathcal{A}_n$'s GP posterior with RFF approximation at iteration t. Note that compared with Chapter 5 and Appendix B.3, we have slightly modified the notations here for better clarity.

Lemma C.3 (Lemma 4 of Dai et al. (2020b)). *Given any* $\delta \in (0, 1)$. *We have that for all agents* $\mathcal{A}_n, \forall n = 1, \ldots, N$, *all* $\mathbf{x} \in \mathcal{X}$ *and all* $t \geq 1$, *with probability of at least* $1 - \delta/2$,

$$|\widetilde{f}_t^n(\mathbf{x}) - f^n(\mathbf{x})| \leq \tilde{\Delta}_{n,t},$$

where $\beta'_t = B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(8N/\delta)}$, *and*

$$\tilde{\Delta}_{n,t} \triangleq \varepsilon\frac{(t+1)^2}{\sigma^2}\left(B + \sqrt{2\log\left(\frac{4\pi^2t^2N}{3\delta}\right)}\right) + \beta'_{t+1} + \sqrt{2\log\frac{2\pi^2t^2N}{3\delta} + M}.$$

Note that a difference between our Lemma C.3 above and Lemma 4 in the work of Dai et al. (2020b) (i.e., Lemma B.4. in Appendix B.3) is that in their proof, they assumed that the number of observations from agent $\mathcal{A}_n$ is a constant t_n; in contrast, we have made use of the fact that in the setting of our DP-FTS-DE algorithm, the number of observations from the other agents are growing with t because all agents are running DP-FTS-DE concurrently. Furthermore, we define

$$\tilde{\Delta}_t^{(i)} \triangleq \sum_{n=1}^{N} \varphi_n^{(i)} \tilde{\Delta}_{n,t}. \tag{C.3}$$

The next lemma gives a uniform upper bound on the difference between the sampled function f_t from agent $\mathcal{A}_1$ and a weighted combination of the sampled function from all agents, which holds throughout all sub-regions $\mathcal{X}_i, \forall i = 1, \ldots, P$.

Lemma C.4. *Denote by* ε *an upper bound on the approximation error of RFF approximation (Section 6.2):* $\sup_{\mathbf{x},\mathbf{x}'\in\mathcal{X}} |k(\mathbf{x},\mathbf{x}') - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')| \leq \varepsilon$. *At iteration* t, *conditioned on the events* $E^f(t)$ *and* $E^{f_t}(t)$, *we have that for all* $\mathbf{x} \in \mathcal{X}$ *and all* $i \in [P]$, *with probability* $\geq 1 - \delta/2$,

$$|\sum_{n=1}^{N} \varphi_n^{(i)} \widetilde{f}_t^n(\mathbf{x}) - f_t(\mathbf{x})| \leq \Delta_t^{(i)},$$

in which $\Delta_t^{(i)} \triangleq \sum_{n=1}^{N} \varphi_n^{(i)} \Delta_{n,t}$, *and*

$$\begin{aligned}
\Delta_{n,t} &\triangleq \tilde{\Delta}_{n,t} + d_n + c_t \\
&= \varepsilon \frac{(t+1)^2}{\sigma^2} \left(B + \sqrt{2 \log \left(\frac{4\pi^2 t^2 N}{3\delta} \right)} \right) + \beta'_{t+1} \\
&\quad + \sqrt{2 \log \frac{2\pi^2 t^2 N}{3\delta} + M} + d_n + c_t \\
&= \tilde{\mathcal{O}}(M^{-1/2} B t^2 + B + \sqrt{M} + d_n + \sqrt{\gamma_t}).
\end{aligned} \tag{C.4}$$

Proof.

$$\begin{aligned}
|\sum_{n=1}^{N} \varphi_n^{(i)} \widetilde{f}_t^n(\mathbf{x}) - f_t(\mathbf{x})| &= |\sum_{n=1}^{N} \varphi_n^{(i)} \widetilde{f}_t^n(\mathbf{x}) - \sum_{n=1}^{N} \varphi_n^{(i)} f_t(\mathbf{x})| \\
&\leq \sum_{n=1}^{N} \varphi_n^{(i)} |\widetilde{f}_t^n(\mathbf{x}) - f_t(\mathbf{x})| \leq \sum_{n=1}^{N} \varphi_n^{(i)} \Delta_{n,t},
\end{aligned} \tag{C.5}$$

where the last inequality results from Lemma 5 of Dai et al. (2020b) (i.e., Lemma B.5. in Appendix B.3). □

Note that Lemma C.4 above takes into account our modifications to the original FTS algorithm (Dai et al., 2020b) (Chapter 5) by including a central server and using an aggregation (i.e., weighted average) of the vectors from all agents (first paragraph of Section 6.3.1). Next, define $\Delta_t \triangleq \sum_{n=1}^{N} \Delta_{n,t}$. Note that $\tilde{\Delta}_t^{(i)} \leq \Delta_t^{(i)} \leq \Delta_t, \forall i = 1, \ldots, P$, and that

$$\sum_{n=1}^{N} \tilde{\Delta}_{n,t} \leq \sum_{n=1}^{N} \Delta_{n,t} = \Delta_t, \tag{C.6}$$

which will be useful in subsequent proofs.

C.1.2.2 Main Proof

The following lemma lower-bounds the probability that the selected input $\mathbf{x}_t$ is unsaturated.

Lemma C.5 (Lemma 7 of Dai et al. (2020b))**.** *For any filtration $\mathcal{F}_{t-1}$, conditioned on the event $E^f(t)$, we have that with probability $\geq 1-\delta/2$,*

$$\mathbb{P}\left(x_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}\right) \geq P_t,$$

in which $P_t \triangleq p_t(p - 1/t^2)$ and $p = \frac{1}{4e\sqrt{\pi}}$.

Proof. Firstly, we have that

$$\begin{aligned}\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}\right) &\geq \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) \mathbb{P}(A_t) \\ &= \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) p_t.\end{aligned} \tag{C.7}$$

Next, we can lower-bound the probability $\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right)$ following Lemma 7 of Dai et al. (2020b) (i.e., Lemma B.7. in Appendix B.3), which leads to $\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) \geq (p - 1/t^2)$ and completes the proof. □

Next, we derive an upper bound on the expected instantaneous regret of our DP-FTS-DE algorithm.

Lemma C.6. *For any filtration $\mathcal{F}_{t-1}$, conditioned on the event $E^f(t)$, we have that with probability of $\geq 1 - 5\delta/8$*

$$\mathbb{E}[r_t | \mathcal{F}_{t-1}] \leq c_t \left(1 + \frac{10}{pp_1}\right) \mathbb{E}\left[\sigma_{t-1}(x_t) | \mathcal{F}_{t-1}\right] + 4B\mathbb{E}\left[\vartheta_t | \mathcal{F}_{t-1}\right] + \psi_t + \frac{2B}{t^2},$$

in which r_t is the instantaneous regret: $r_t \triangleq f(\mathbf{x}^*) - f(\mathbf{x}_t)$, $\vartheta_t \triangleq (1 - p_t)\sum_{i=1}^{P} \widetilde{\varphi}_{\mathcal{C}_t}^{(i)}$, and

$$\psi_t \triangleq (1-p_t)P\left[\left(\frac{\varphi_{\max}+2}{q}+6\right)\Delta_t + B\left(\frac{2}{q}+\frac{N\varphi_{\max}}{q}\right) + \frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}}\right].$$

Proof. Firstly, we define $\overline{\mathbf{x}}_t$ as the unsaturated input at iteration t with the smallest

posterior standard deviation according to agent $\mathcal{A}_1$'s own GP posterior:

$$\overline{\mathbf{x}}_t \triangleq \arg\min_{\mathbf{x}\in\mathcal{X}\setminus S_t} \sigma_{t-1}(\mathbf{x}). \tag{C.8}$$

Following this definition, for any $\mathcal{F}_{t-1}$ such that $E^f(t)$ is true, we have that

$$\begin{aligned}\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] &\geq \mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}, \mathbf{x}_t \in \mathcal{X}\setminus S_t\right]\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X}\setminus S_t|\mathcal{F}_{t-1}\right)\\ &\geq \sigma_{t-1}(\overline{\mathbf{x}}_t)P_t,\end{aligned} \tag{C.9}$$

in which the last inequality follows from the definition of $\overline{\mathbf{x}}_t$ and Lemma C.5.

Next, conditioned on both events $E^f(t)$ and $E^{f_t}(t)$, we have that

$$\begin{aligned}r_t = \Delta(\mathbf{x}_t) &= f(\mathbf{x}^*) - f(\overline{\mathbf{x}}_t) + f(\overline{\mathbf{x}}_t) - f(\mathbf{x}_t)\\ &\overset{(a)}{\leq} \Delta(\overline{\mathbf{x}}_t) + f_t(\overline{\mathbf{x}}_t) + c_t\sigma_{t-1}(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t) + c_t\sigma_{t-1}(\mathbf{x}_t)\\ &\overset{(b)}{\leq} c_t\sigma_{t-1}(\overline{\mathbf{x}}_t) + c_t\sigma_{t-1}(\overline{\mathbf{x}}_t) + c_t\sigma_{t-1}(\mathbf{x}_t) + f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)\\ &= c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t) + \sigma_{t-1}(\mathbf{x}_t)) + \underline{f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)},\end{aligned} \tag{C.10}$$

in which (a) follows from the definition of $\Delta(\mathbf{x})$ and equation (C.2), and (b) results from the fact that $\overline{\mathbf{x}}_t$ is unsaturated. Denote by $\bar{i}$ the sub-region to which $\overline{\mathbf{x}}_t$ belongs given $\mathcal{F}_{t-1}$. Next, we analyze the expected value of the underlined

term given $\mathcal{F}_{t-1}$:

$$
\begin{aligned}
&\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] \\
&\overset{(a)}{=} \mathbb{P}\left(A_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, A_t\right] + \mathbb{P}\left(B_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t\right] \\
&\overset{(b)}{\leq} \mathbb{P}\left(B_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t\right] \\
&\overset{(c)}{\leq} \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{P}\left[\mathbf{x}_t \in \mathcal{X}_i\right]\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right] \\
&\leq \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right] \\
&\overset{(d)}{\leq} \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{E}\left[\sum_{n=1}^{N}\varphi_n^{(i)}\widetilde{f}_t^n(\overline{\mathbf{x}}_t) + \Delta_t^{(i)} - \sum_{n=1}^{N}\varphi_n^{(i)}\widetilde{f}_t^n(\mathbf{x}_t) + \Delta_t^{(i)}\middle|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right] \\
&\overset{(e)}{\leq} \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{E}\left[\left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right]\underline{\sum_{n=1}^{N}\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}} + 2\Delta_t^{(i)}\middle|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right],
\end{aligned}
\tag{C.11}
$$

in which (a) and (c) result from the tower rule of expectation; (b) follows since conditioned on the event A_t, i.e., $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t(\mathbf{x})$, we have that $f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t) \leq 0$; (d) results from Lemma C.4 and hence holds with probability $\geq 1-\delta/2$; (e) is a consequence of the definition of $\widetilde{f}_t^n$: $\widetilde{f}_t^n(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top\boldsymbol{\omega}_{n,t}, \forall \mathbf{x} \in \mathcal{X}$. Next, we further decompose the underlined term $\underline{\sum_{n=1}^{N}\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}$ by:

$$
\begin{aligned}
\sum_{n=1}^{N}\varphi_n^{(i)}\boldsymbol{\omega}_{n,t} &= \frac{\sum_{n=1}^{N} q\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} = \frac{\sum_{n=1}^{N}\mathbb{E}_{\mathbb{I}_n}[\mathbb{I}_n]\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} \\
&= \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q}\right] \\
&= \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} - \boldsymbol{\eta} + \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} + \boldsymbol{\eta}\right] \\
&= \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} - \boldsymbol{\eta} + \boldsymbol{\omega}_t^{(i)}\right].
\end{aligned}
\tag{C.12}
$$

Next, we plug (C.12) back into (C.11):

$$
\begin{aligned}
&\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{P}\left(B_t\right) \sum_{i=1}^{P} \mathbb{E}\Bigg[\left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N} \mathbb{I}_n \varphi_n^{(i)} \boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^{N} \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q} - \boldsymbol{\eta}\right] + \\
&\qquad \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \mathbb{E}_{\mathbb{I}_{1:n}}[\boldsymbol{\omega}_t^{(i)}] - \boldsymbol{\phi}(\mathbf{x}_t)^\top \mathbb{E}_{\mathbb{I}_{1:n}}[\boldsymbol{\omega}_t^{(i)}]\right] + 2\Delta_t^{(i)} \Big| \mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i \Bigg] \\
&\leq \mathbb{P}\left(B_t\right) \sum_{i=1}^{P} \mathbb{E}\Bigg[\underbrace{\left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N} \mathbb{I}_n \varphi_n^{(i)} \boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^{N} \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right]}_{A_1} + \\
&\qquad \Big[\underbrace{\phi(\overline{\mathbf{x}}_t)^\top \mathbb{E}_{\mathbb{I}_{1:n}}[\boldsymbol{\omega}_t^{(i)}] - \phi(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_t^{(i)}}_{A_2} + \underbrace{\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_t^{(i)} - \boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_t^{(\bar{i})}}_{A_3} + \\
&\qquad \underbrace{\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_t^{(\bar{i})} - \boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_t^{(i)}}_{A_4} + \underbrace{\boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_t^{(i)} - \boldsymbol{\phi}(\mathbf{x}_t)^\top \mathbb{E}_{\mathbb{I}_{1:n}}[\boldsymbol{\omega}_t^{(i)}]}_{A_5} \Big] \\
&\qquad \underbrace{- \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \boldsymbol{\eta}}_{A_6} + 2\Delta_t^{(i)} \Big| \mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i \Bigg]
\end{aligned}
\tag{C.13}
$$

Next, we separately upper-bound the terms A_1 to A_6. Firstly, we bound the term A_1. Define $\mathcal{C}_t \triangleq \{n = 1, \ldots, N \Big| \left\|\boldsymbol{\omega}_{n,t}\right\|_2 > S/\sqrt{P}\}$. That is, $\mathcal{C}_t$ contains the indices of those agents whose vector of $\boldsymbol{\omega}_{n,t}$ has a larger L_2 norm than $S/\sqrt{P}$ in

iteration t. A_1 can thus be analyzed as:

$$
\begin{aligned}
&\left| \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right]\right| \\
&\overset{(a)}{=} \left| \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \sum_{n=1}^N \varphi_n^{(i)} (\boldsymbol{\omega}_{n,t} - \widehat{\boldsymbol{\omega}}_{n,t}) \right| \\
&\overset{(b)}{=} \left| \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} (\boldsymbol{\omega}_{n,t} - \widehat{\boldsymbol{\omega}}_{n,t}) \right| \\
&= \left| \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \left[\boldsymbol{\omega}_{n,t} - \widehat{\boldsymbol{\omega}}_{n,t}\right] \right| \\
&= \left| \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_{n,t} + \boldsymbol{\phi}(\mathbf{x}_t)^\top \widehat{\boldsymbol{\omega}}_{n,t} - \boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \widehat{\boldsymbol{\omega}}_{n,t} - \boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_{n,t}\right] \right| \\
&\leq \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} \left[\left|\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_{n,t}\right| + \left|\boldsymbol{\phi}(\mathbf{x}_t)^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| + \left|\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| + \left|\boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_{n,t}\right|\right] \\
&\leq \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} \left[\left|\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_{n,t}\right| + \left|\boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_{n,t}\right| + \left|\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_{n,t}\right| + \left|\boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_{n,t}\right|\right] \\
&\leq 2 \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} \left[\left|\widetilde{f}_t^n(\overline{\mathbf{x}}_t)\right| + \left|\widetilde{f}_t^n(\mathbf{x})\right|\right] \\
&\overset{(c)}{\leq} 2 \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} (\tilde{\Delta}_{n,t} + B + \tilde{\Delta}_{n,t} + B) \\
&= 4 \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} \tilde{\Delta}_{n,t} + 4 \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)} B \\
&\overset{(d)}{\leq} 4 \sum_{n=1}^N \varphi_n^{(i)} \tilde{\Delta}_{n,t} + 4B \widetilde{\varphi}_{\mathcal{C}_t}^{(i)} \\
&\overset{(e)}{=} 4 \left(\tilde{\Delta}_t^{(i)} + B \widetilde{\varphi}_{\mathcal{C}_t}^{(i)}\right),
\end{aligned}
\tag{C.14}
$$

in which (a) follows since $\mathbb{E}_n[\mathbb{I}_n] = q, \forall n = 1, \ldots, N$; (b) follows since for those agents $n \notin \mathcal{C}_t$, $\boldsymbol{\omega}_{n,t} - \widehat{\boldsymbol{\omega}}_{n,t} = \mathbf{0}$; (c) results from Lemma C.3 and that $|f^n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}, n = 1, \ldots, N$ (this is because of our assumption that $\|f^n\|_k \leq B, \forall n = 1, \ldots, N$, Section 6.2); (d) follows from the definition of $\widetilde{\varphi}_{\mathcal{C}_t}^{(i)} \triangleq \sum_{n\in\mathcal{C}_t} \varphi_n^{(i)}$; (e) results from the definition of $\tilde{\Delta}_t^{(i)}$ (C.3).

Subsequently, we upper-bound the terms A_2 and A_5. For any $\mathbf{x} \in \mathcal{X}$, we have

that

$$
\begin{aligned}
&\left|\boldsymbol{\phi}(\mathbf{x})^{\top}\mathbb{E}_{\mathbb{I}_{1:n}}[\boldsymbol{\omega}_t^{(i)}] - \boldsymbol{\phi}(\mathbf{x})^{\top}\boldsymbol{\omega}_t^{(i)}\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^{\top}\left(\mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q}\right] - \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q}\right)\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^{\top}\left(\frac{\sum_{n=1}^{N}q\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} - \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q}\right)\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^{\top}\frac{1}{q}\sum_{n=1}^{N}(q-\mathbb{I}_n)\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}\right| \leq \frac{1}{q}\sum_{n=1}^{N}\left|(q-\mathbb{I}_n)\varphi_n^{(i)}\boldsymbol{\phi}(\mathbf{x})^{\top}\widehat{\boldsymbol{\omega}}_{n,t}\right| \\
&\overset{(a)}{\leq} \frac{1}{q}\sum_{n=1}^{N}\varphi_n^{(i)}\left|\boldsymbol{\phi}(\mathbf{x})^{\top}\widehat{\boldsymbol{\omega}}_{n,t}\right| \overset{(b)}{\leq} \frac{1}{q}\sum_{n=1}^{N}\varphi_n^{(i)}\left|\boldsymbol{\phi}(\mathbf{x})^{\top}\boldsymbol{\omega}_{n,t}\right| \\
&= \frac{1}{q}\sum_{n=1}^{N}\varphi_n^{(i)}\left|\widetilde{f}_t^n(\mathbf{x})\right| = \frac{1}{q}\sum_{n=1}^{N}\varphi_n^{(i)}\left|\widetilde{f}_t^n(\mathbf{x}) - f^n(\mathbf{x}) + f^n(\mathbf{x})\right| \\
&\leq \frac{1}{q}\sum_{n=1}^{N}\varphi_n^{(i)}\left(|\widetilde{f}_t^n(\mathbf{x}) - f^n(\mathbf{x})| + |f^n(\mathbf{x})|\right) \\
&\overset{(c)}{\leq} \frac{1}{q}\sum_{n=1}^{N}\varphi_n^{(i)}\left(\tilde{\Delta}_{n,t} + B\right) \\
&\overset{(d)}{=} \frac{1}{q}\left(\tilde{\Delta}_t^{(i)} + B\right),
\end{aligned}
\tag{C.15}
$$

in which (a) follows since $|q - \mathbb{I}_n| \leq 1$; (b) results from (C.1); (c) results from Lemma C.3 and that $|f^n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}, n = 1, \ldots, N$; (d) results from the definition of $\tilde{\Delta}_t^{(i)}$ (C.3).

Next, we upper-bound the term A_3, which arises because the sub-regions i

and $\bar{i}$ may be different. We have that for any $\mathbf{x} \in \mathcal{X}$,

$$\begin{aligned}
\left|\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i)} - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(\bar{i})}\right| &= \left|\boldsymbol{\phi}(\mathbf{x})^\top \left(\boldsymbol{\omega}_t^{(i)} - \boldsymbol{\omega}_t^{(\bar{i})}\right)\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^\top \left(\frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q} - \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(\bar{i})} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right)\right| \\
&\leq \frac{1}{q}\sum_{n=1}^N \mathbb{I}_n \left|\varphi_n^{(i)} - \varphi_n^{(\bar{i})}\right| \left|\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| \\
&\leq \frac{1}{q}\sum_{n=1}^N \varphi_{\max} \left|\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| \\
&\overset{(a)}{\leq} \frac{1}{q}\sum_{n=1}^N \varphi_{\max} \left|\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}\right| \\
&\overset{(b)}{\leq} \frac{1}{q}\sum_{n=1}^N \varphi_{\max} \left(\tilde{\Delta}_{n,t} + B\right) \\
&\overset{(c)}{\leq} \frac{\varphi_{\max}}{q}\left(\Delta_t + NB\right),
\end{aligned} \tag{C.16}$$

(a) follows because of (C.1); (b) results from Lemma C.3 and that $|f^n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}, n = 1, \ldots, N$; (c) follows from (C.6).

Next, regarding A_4, note that conditioned on the event B_t, $\mathbf{x}_t$ is selected by: $\mathbf{x}_t = \arg\max_{\mathbf{x}\in\mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i^{[\mathbf{x}]})}$ in which $i^{[\mathbf{x}]}$ represents the sub-region $\mathbf{x}$ belongs to. Therefore, because $\overline{\mathbf{x}}_t \in \mathcal{X}_{\bar{i}}$ and $\mathbf{x}_t \in \mathcal{X}_i$ (since we are conditioning on this event), we have that $\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top \boldsymbol{\omega}_t^{(\bar{i})} - \boldsymbol{\phi}(\mathbf{x}_t)^\top \boldsymbol{\omega}_t^{(i)} \leq 0$. In other words, $A_4 \leq 0$.

Finally, The term A_6 can be upper-bounded using standard Gaussian concentration inequality:

$$\begin{aligned}
\left|\left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right]\boldsymbol{\eta}\right| &\leq \left\|\boldsymbol{\phi}(\overline{\mathbf{x}}_t) - \boldsymbol{\phi}(\mathbf{x}_t)\right\|_2 \|\boldsymbol{\eta}\|_2 \\
&\leq \left(\left\|\boldsymbol{\phi}(\overline{\mathbf{x}}_t)\right\|_2 + \left\|\boldsymbol{\phi}(\mathbf{x}_t)\right\|_2\right)\|\boldsymbol{\eta}\|_2 \overset{(a)}{\leq} 2\|\boldsymbol{\eta}\|_2 \\
&\overset{(b)}{\leq} \frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}},
\end{aligned} \tag{C.17}$$

where (a) follows since the random features have been constructed such that $\left\|\boldsymbol{\phi}(\mathbf{x})\right\|_2^2 = \sigma_0^2 \leq 1$ (Dai et al., 2020b) (Appendix B.1), and (b) follows from

standard Gaussian concentration inequality and hence holds with probability $> 1 - \delta/8$.

Now we can exploit the upper bounds on the terms A_1 to A_6 we have derived above (equations (C.14), (C.15), (C.16), (C.17)), and continue to upper-bound $\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]$ following (C.13):

$$
\begin{aligned}
\mathbb{E}[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}] &\leq \mathbb{P}(B_t) \sum_{i=1}^{P} \mathbb{E}\Bigg[\underbrace{4\left(\tilde{\Delta}_t^{(i)} + B\widetilde{\varphi}_{\mathcal{C}_t}^{(i)}\right)}_{A_1} + \underbrace{\frac{2}{q}\left(\tilde{\Delta}_t^{(i)} + B\right)}_{A_2+A_5} + \\
&\quad \underbrace{\frac{\varphi_{\max}}{q}\left(\Delta_t + NB\right)}_{A_3} + \underbrace{\frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}}}_{A_6} + 2\Delta_t^{(i)} \Bigg| \mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i \Bigg] \\
&= (1-p_t)\sum_{i=1}^{P}\Bigg[4\left(\tilde{\Delta}_t^{(i)} + B\mathbb{E}\left[\widetilde{\varphi}_{\mathcal{C}_t}^{(i)}|\mathcal{F}_{t-1}\right]\right) + \frac{2}{q}\left(\tilde{\Delta}_t^{(i)} + B\right) + \\
&\quad \frac{\varphi_{\max}}{q}\left(\Delta_t + NB\right) + \frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}} + 2\Delta_t^{(i)}\Bigg] \\
&= (1-p_t)\sum_{i=1}^{P}\Bigg[4B\mathbb{E}\left[\widetilde{\varphi}_{\mathcal{C}_t}^{(i)}|\mathcal{F}_{t-1}\right] + \left(\frac{2}{q}+4\right)\tilde{\Delta}_t^{(i)} + 2\Delta_t^{(i)} + \frac{\varphi_{\max}}{q}\Delta_t + \\
&\quad B\left(\frac{2}{q} + \frac{N\varphi_{\max}}{q}\right) + \frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}}\Bigg] \\
&\overset{(a)}{\leq} (1-p_t)\sum_{i=1}^{P}\Bigg[4B\mathbb{E}\left[\widetilde{\varphi}_{\mathcal{C}_t}^{(i)}|\mathcal{F}_{t-1}\right] + \left(\frac{2}{q}+6+\frac{\varphi_{\max}}{q}\right)\Delta_t + \\
&\quad B\left(\frac{2}{q} + \frac{N\varphi_{\max}}{q}\right) + \frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}}\Bigg] \\
&= 4B\mathbb{E}\left[(1-p_t)\sum_{i=1}^{P}\widetilde{\varphi}_{\mathcal{C}_t}^{(i)}|\mathcal{F}_{t-1}\right] + (1-p_t)\Bigg[P\left(\frac{2}{q}+6+\frac{\varphi_{\max}}{q}\right)\Delta_t + \\
&\quad PB\left(\frac{2}{q} + \frac{N\varphi_{\max}}{q}\right) + P\frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}}\Bigg] \\
&= 4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right] + \psi_t,
\end{aligned}
\tag{C.18}
$$

where (a) follows because $\tilde{\Delta}_t^{(i)} \leq \Delta_t^{(i)} \leq \Delta_t, \forall i \in [P]$. In the last equality,

we have made use of the definitions of ϑ_t and ψ_t. Note that since we have made use of Lemma C.4 (C.11) which holds with probability $\geq 1-\delta/2$, and Gaussian concentration inequality (C.17) which holds with probability $\geq 1-\delta/8$, equation (C.18) holds with probability $\geq 1-\delta/2-\delta/8 = 1-5\delta/8$.

Finally, we plug (C.18) back into (C.10):

$$
\begin{aligned}
&\mathbb{E}\left[r_t|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))+f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+2B\mathbb{P}\left[\overline{E^{f_t}(t)}|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))|\mathcal{F}_{t-1}\right]+\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] \\
&\qquad +2B\mathbb{P}\left[\overline{E^{f_t}(t)}|\mathcal{F}_{t-1}\right] \\
&\overset{(a)}{\leq} \frac{2c_t}{P_t}\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+c_t\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2} \\
&\leq c_t\left(1+\frac{2}{P_t}\right)\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2} \\
&\overset{(b)}{\leq} c_t\left(1+\frac{10}{pp_1}\right)\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2},
\end{aligned}
\tag{C.19}
$$

in which (a) follows from (C.9) and (C.18), and (b) follows since:

$$
\frac{2}{P_t}=\frac{2}{p_t(p-\frac{1}{t^2})}\leq\frac{10}{pp_t}\leq\frac{10}{pp_1},
\tag{C.20}
$$

which was valid because $1/(p-1/t^2)\leq 5/p$ and $p_t\geq p_1$ for all $t\geq 1$.

Note that since the proof of (C.19) makes use of (C.18), therefore, (C.19), as well as Lemma C.6, also holds with probability of $\geq 1-5\delta/8$.

□

Lemma C.7. *Given $\delta \in (0,1)$, then with probability of at least $1-\delta$,*

$$R_T \leq c_T \left(1 + \frac{10}{pp_1}\right) \mathcal{O}(\sqrt{T\gamma_T}) + \sum_{t=1}^{T} \psi_t + \frac{B\pi^2}{3} + 4B \sum_{t=1}^{T} \vartheta_t + \left[c_T \left(1 + \frac{4B}{p} + \frac{10}{pp_1}\right) + \psi_1 + 4B\right] \sqrt{2T \log \frac{8}{\delta}},$$

in which γ_T is the maximum information gain about f obtained from any set of T observations.

Proof. The proof resembles the that of Lemma 11 of Dai et al. (2020b) (Lemma B.11. of Appendix B), and is hence omitted. Note that an error probability of $\delta/8$ has been used here for the Azuma-Hoeffding Inequality in the proof. □

Finally, we are ready to prove Theorem 6.1. Recall we have that $c_t = \mathcal{O}\left(\left(B + \sqrt{\gamma_t + \log(1/\delta)}\right)\sqrt{\log t}\right)$. Therefore,

$$\begin{aligned} R_T &= \mathcal{O}\left(\frac{1}{p_1}\left(B + \sqrt{\gamma_T + \log\frac{1}{\delta}}\right)\sqrt{\log T}\sqrt{T\gamma_T} + \sum_{t=1}^{T}\psi_t + B\sum_{t=1}^{T}\vartheta_t + \left(B + \frac{1}{p_1}\right)\left(B + \sqrt{\gamma_T + \log\frac{1}{\delta}}\right)\sqrt{\log T}\sqrt{T\log\frac{1}{\delta}}\right) \\ &= \mathcal{O}\left(\left(B + \frac{1}{p_1}\right)\sqrt{T\log T\gamma_T \log\frac{1}{\delta}\left(\gamma_T + \log\frac{1}{\delta}\right)} + \sum_{t=1}^{T}\psi_t + B\sum_{t=1}^{T}\vartheta_t\right) \\ &= \tilde{\mathcal{O}}\left(\left(B + \frac{1}{p_1}\right)\gamma_T\sqrt{T} + \sum_{t=1}^{T}\psi_t + B\sum_{t=1}^{T}\vartheta_t\right), \end{aligned} \tag{C.21}$$

which finally completes the proof.

C.2 Experiments

As we have mentioned in the main text (Section 6.5), to design the sets of weights for different sub-regions, for every sub-region $\mathcal{X}_i$, we choose the corresponding

set of weights such that $\varphi_n^{(i)} \propto \exp(a)$ for those agents exploring the sub-region $\mathcal{X}_i$, and $\varphi_n^{(i)} \propto \exp(b)$ for the other agents, with $a \geq b > 0$. For the synthetic experiments, we design the adaptive weights by fixing $b = 1$, setting $a = 16$ for the first 5 iterations, and decaying the value of a linearly to $b = 1$ in the next 5 iterations (i.e., $a = 16$ for $t = 1, \ldots, 5$, and $a = 16, 12.25, 8.5, 4.75, 1$ for $t = 6, \ldots, 10$ respectively). After the first 10 iterations, the weights for all sub-regions become uniform among all agents (i.e., $a = b = 1, \forall t > 10$). In this way, for every sub-region $\mathcal{X}_i$, in the early stage, we give most weights to those agents exploring $\mathcal{X}_i$, whereas as t becomes large, we gradually make the weights become uniform among all agents. Similarly, for all real-world experiments, we fix $b = 1$ and $a = 16$ for the first 10 iterations, and then decay a linearly to $b = 1$ in the next 30 iterations, such that the weights for every sub-region become uniform among all agents after the first 40 iterations. All our experiments are performed on a computing cluster where each device has one NVIDIA Tesla T4 GPU and 48 cores of Xeon Silver 4116 (2.1Ghz) processors.

C.2.1 Synthetic Experiments

C.2.1.1 Detailed Experimental Setting

Our synthetic experiments involve $N = 200$ agents. We define the domain of the synthetic functions to be 1-dimensional and discrete, i.e., an equally spaced grid on the 1-dimensional interval $[0, 1]$ with a domain size of $|\mathcal{X}| = 1000$. To generate the objective functions for the $N = 200$ different agents, we firstly sample a function f from a GP with the SE kernel and a length scale of 0.03, and normalize the function values into the range $[0, 1]$. Next, for every agent $\mathcal{A}_n, \forall n = 1, \ldots, N$, we go through all $|\mathcal{X}| = 1000$ inputs in the entire domain, and for each input $\mathbf{x}$, we derive the function value for agent $\mathcal{A}_n$ as $f^n(\mathbf{x}) = f(\mathbf{x}) + d$, in which $d = 0.02$ or $= -0.02$ with equal probability (i.e., a probability of 0.5 each). In this way,

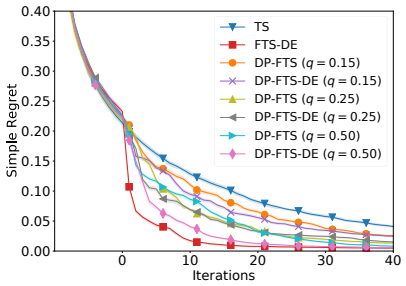


Figure C.1: Comparisons of the performances of DP-FTS (without DE) and DP-FTS-DE. For a fair comparison, we have used $S = 8$ and $S = 11$ for DP-FTS and DP-FTS-DE respective, such that a similar small percentage vectors are clipped for both algorithms (0.31% for DP-FTS and 0.80% for DP-FTS-DE). We have used $z = 1.0$ for both algorithms.

the objective functions of all agents are related to each other. When observing a function value, we add a Gaussian noise $\zeta \sim \mathcal{N}(0, \sigma^2)$ with a variance of $\sigma^2 = 0.01$ (Section 6.2).

To construct the P sub-regions to be used for distributed exploration (DE), we simply need to divide the interval $[0, 1]$ into P disjoint hyper-rectangles with equal volumes. For example, when $P = 2$, the two sub-regions contain the inputs in the sub-regions $[0, 0.5)$ and $[0.5, 1]$ respectively; when $P = 3$, the three sub-regions include the inputs in the sub-regions $[0, 1/3)$, $[1/3, 2/3)$ and $[2/3, 1]$ respectively.

C.2.1.2 More Experimental Results

Comparison between DP-FTS-DE and DP-FTS. We have shown in the main text (Fig. 6.3a) that FTS-DE significantly outperforms FTS without DE. Here, we demonstrate in Fig. C.1 that after DP is integrated, DP-FTS-DE still yields a significantly better utility than DP-FTS for the same level of privacy guarantee (loss). These results justify the practical benefit of the technique of DE (Section 6.3.2). Note that for a fair comparison, we have used a smaller value of S for DP-FTS without DE such that a similar percentage of vectors are clipped for both DP-FTS-DE and DP-FTS.

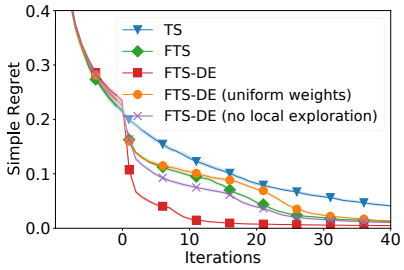


Figure C.2: Investigating the importance of both major components of the technique of distributed exploration (DE). The orange curve is obtained by giving equal weights to all agent for every sub-region, and the purple curve is derived by letting every agent explore the entire domain at initialization instead of a local sub-region.

Investigation of DE. We also investigate the importance of both of the major components of the DE technique (Section 6.3.2): (a) assigning every agent to explore only a local sub-region instead of the entire domain, and (b) giving more weights to those agents exploring the particular sub-region. In Fig. C.2, the orange curve is obtained by removing component (b) (i.e., in every iteration and for each sub-region, we give equal weights to all agents), the purple curve is derived by removing component (a) (i.e., letting every agent explore the entire domain at initialization instead of a smaller local sub-region). As the figure shows, the performances of both the orange and purple curves are significantly worse than our FTS-DE algorithm (red curve), which verifies that both of these components are critical for the practical performance of FTS-DE.

Trade-off Induced by P. As we have discussed at the end of Section 6.4.2, the value of P (i.e., the number of sub-regions) induces a trade-off about the practical performance of our DP-FTS-DE algorithm. Here we empirically verify this trade-off in Fig. C.3. As shown in the figure, for the same values of q, z and S, a smaller value of P (i.e., larger local sub-regions) may deteriorate the performance (orange curve) since larger sub-regions are harder for the GP surrogate to model, however, a larger value of P may also result in a worse performance (yellow curve) since it causes the vectors from more agents to be

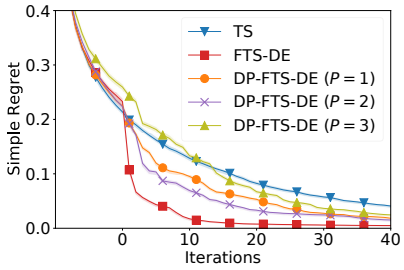


Figure C.3: Trade-off induced by P regarding the practical performance of our DP-FTS-DE algorithm. Note that a larger P reduces the size of every local sub-region and hence leads to a better modeling by the GP surrogates, yet also negatively impacts the performance by causing more vectors to be clipped. Here we have used $q = 0.25, z = 1.0, S = 11.0$ for all values of P.

clipped (Section 6.4.2). These observations verify our discussions in the last paragraph of Section 6.4.2.

C.2.2 Real-world Experiments

C.2.2.1 More Experimental Details

In all real-world experiments, when generating the random features for the RFF approximation, we use the SE kernel with a length scale of 0.01 and a variance of $\sigma^2 = 10^{-6}$ for the observation noise. Refer to Appendix B.1 for more details on how the random features are generated and how they are shared among all agents.

As we have mentioned in the main text, we use $P = 4$ sub-regions in all three real-world experiments, and divide the entire domain into $P = 4$ hyper-rectangles (i.e., sub-regions) with equal volumes. Following the common practice in BO, we assume that the domain $\mathcal{X} \in \mathbb{R}^D$ is a D-dimensional hyper-rectangle, and w.l.o.g., assume that every dimension of the domain is normalized into the range $[0, 1]$. That is, the domain can be represented as $[0, 1]^D = \{[0, 1], [0, 1], \ldots, [0, 1]\}$. Note that every domain which is a hyper-rectangle can be normalized into this form. As a result, when the input dimension is $D = 2$ (i.e., the landmine detection experiment), we construct the $P = 4$ hyper-

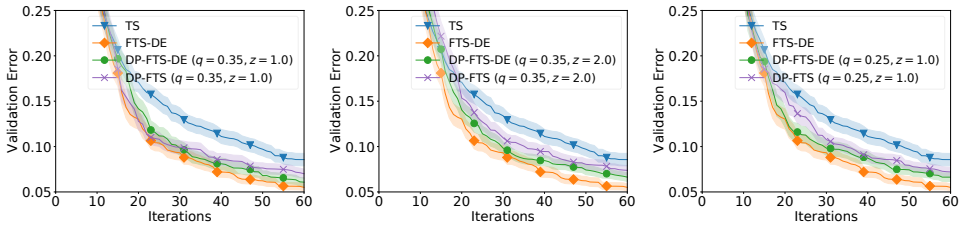
rectangles such that $\mathcal{X}_1 = \{[0, 0.5), [0, 0.5)\}$, $\mathcal{X}_2 = \{[0, 0.5), [0.5, 1.0]\}$, $\mathcal{X}_3 = \{[0.5, 1.0], [0, 0.5)\}$ and $\mathcal{X}_4 = \{[0.5, 1.0], [0.5, 1.0]\}$. Similarly, when the input dimension $D = 3$ (i.e., the human activity recognition and EMNIST experiments), we construct the $P = 4$ hyper-rectangles such that $\mathcal{X}_1 = \{[0, 0.5), [0, 0.5), [0, 1]\}$, $\mathcal{X}_2 = \{[0, 0.5), [0.5, 1.0], [0, 1]\}$, $\mathcal{X}_3 = \{[0.5, 1.0], [0, 0.5), [0, 1]\}$ and $\mathcal{X}_4 = \{[0.5, 1.0], [0.5, 1.0], [0, 1]\}$.

The datasets and detailed settings for the landmine detection and human activity recognition experiments have been introduced in Appendix B.4.2.2. The EMNIST dataset[1] used in the third experiment is a dataset of images of handwritten characters from different persons, and is a widely used benchmark in FL Kairouz et al. (2019). Here we use the images from the first $N = 50$ subjects (agents) which can be accessed from the TensorFlow Federated library[2]. Every subject (agent) uses a convolutional neural network (CNN) to learn to classify an image into one of the ten classes corresponding to the digits $0 - 9$. Here the task for every agent is to tune three CNN hyperparameters: learning rate, learning rate decay and L2 regularization parameter, all in the range of $[10^{-7}, 0.02]$. We follow the standard training/validation split offered by the TensorFlow Federated library for every agent, and again use the validation error as the performance metric. All images are pre-processed by normalizing all pixel values into the range of $[0, 1]$, and no data is excluded. Refer to Cohen et al. (2017) for more details on this dataset.

C.2.2.2 Comparison between DP-FTS-DE and DP-FTS

We have shown in the main text (Figs. 6.5a,c,e) that FTS-DE significantly outperforms FTS without DE. Here we further verify in Fig. C.4 the importance of the technique of DE after DP is integrated, using the human activity recognition experiment. Specifically, the figures show that after the incorporation of DP,

[1] https://www.nist.gov/itl/products-and-services/emnist-dataset.
[2] https://www.tensorflow.org/federated.



(a) (b) (c)

Figure C.4: Comparison between DP-FTS and DP-FTS-DE for the same level of privacy guarantee (the human activity recognition experiment). We have used $S = 22$ and $S = 11$ for DP-FTS-DE and DP-FTS (without DE) respectively, such that a similar percentage of vectors are clipped in both cases: 1.02% for DP-FTS-DE and 1.09% for DP-FTS.

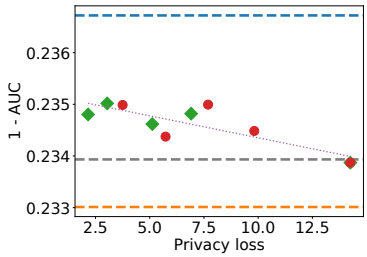


Figure C.5: Results for the landmine detection experiment using Rényi DP Wang et al. (2019).

DP-FTS-DE (green curves in all three figures) still achieves a better utility than DP-FTS (purple curves) for the same level of privacy guarantee (loss). Note that same as Fig. C.1, to facilitate a fair comparison, we have used a smaller value of S for DP-FTS without DE such that a similar percentage of vectors are clipped for both DP-FTS-DE and DP-FTS.

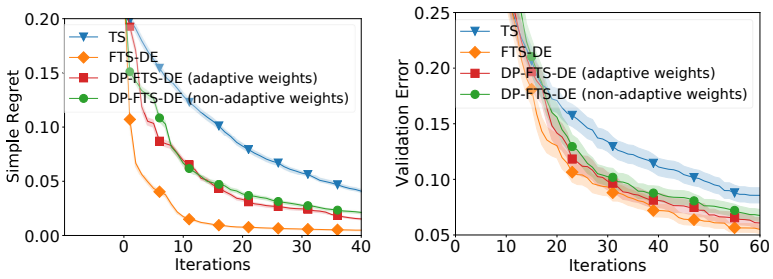
C.2.2.3 Rényi DP

Fig. C.5 shows the privacy-utility trade-off in the landmine detection experiment using Rényi DP Wang et al. (2019). The results demonstrate that Rényi DP, despite requiring modifications to our theoretical analysis (i.e., proof of Theorem 6.1), leads to slightly better privacy losses (compared with Fig. 6.5a) with comparable utilities.

C.2.2.4 Adaptive Weights vs. Non-adaptive Weights

As we have discussed in the last paragraph of Section 6.3.2, we have designed the set of weights for every sub-region to be adaptive such that they gradually become uniform among all agents as t becomes large. Here we explore the performance of our algorithm if the weights are *non-adaptive*, i.e., for every sub-region $\mathcal{X}_i$, we fix the set of weights $\{\varphi_n^{(i)}, \forall n = 1, \ldots, N\}$ for all $t = 1, \ldots, T$. Similar to our setting of the adaptive weights as we have introduced in the first paragraph of Appendix C.2, for a sub-region $\mathcal{X}_i$, we set $\varphi_n^{(i)} \propto \exp(a)$ for those agents exploring this particular sub-region $\mathcal{X}_i$, and $\varphi_n^{(i)} \propto \exp(b)$ for the other agents. However, for every sub-region $\mathcal{X}_i$, we now use the same set of weights throughout all iterations by fixing $a = 16, b = 1$ for all $t = 1, \ldots, T$.

Fig. C.6 shows the comparisons between adaptive and non-adaptive weights using (a) the synthetic experiment and (b) human activity recognition experiment. Both figures show that although in the initial stage, DP-FTS-DE with non-adaptive weights performs similarly to DP-FTS-DE with adaptive weights, however, as t becomes large, adaptive weights (red curves) lead to better performances than non-adaptive weights (green curves). This can be attributed to the fact that as t becomes large, every agent is likely to have explored (and become informative about) more sub-regions in addition to the sub-region that it is assigned to explore at initialization. Therefore, if the weights are non-adaptive, i.e., for a sub-region $\mathcal{X}_i$, if after t has become large, most weights are still given to those agents that are assigned to explore $\mathcal{X}_i$ at initialization, then the information from the other agents who are likely to have become informative about $\mathcal{X}_i$ (i.e., have collected some observations in $\mathcal{X}_i$) is not utilized. This under-utilization of information might explain the performance deficit caused by the use of non-adaptive weights. However, note that despite being outperformed by DP-FTS-DE with adaptive weights, DP-FTS-DE with non-adaptive weights (green curve) is still able to consistently outperform standard TS (blue curves).



(a) (b)

Figure C.6: Comparison between DP-FTS-DE with adaptive weights and non-adaptive weights, using (a) the synthetic experiment and (b) human activity recognition experiment.

C.2.2.5 Computational Cost

When maximizing the acquisition function to select the next query (lines 5 and 7 of Algorithm 6.2), firstly, we uniformly randomly sample a large number (i.e., 1000) of points from the entire domain; next, we use the L-BFGS-B method with 20 random restarts to refine the optimization.

For the central server, the integration of DP and DE (in expectation) incurs an additional computational cost of $\mathcal{O}(PNq)$. However, these additional computations are negligible since they only involve simple vector additions/multiplications (lines 5-11 of Algorithm 6.1). For agents, the incorporation of DP brings no additional computational cost to them. Meanwhile, the addition of DE, which affects line 7 of Algorithm 6.2, only incurs minimal additional computations. For example, in the landmine detection experiment, line 7 takes on average 14.47s and 17.96s to compute for $P = 1$ and $P = 4$, respectively.

Appendix D

Appendix for Chapter 7

D.1 Proof of Theoretical Results

To facilitate the theoretical analysis, we introduce the following auxiliary term:

$$\widetilde{\zeta}_t(\mathbf{x}) = \nu_t \left[\sum_{i=1}^{M} \omega_i \left[\widetilde{\mu}_i(\mathbf{x}) + \sqrt{\tau} \widetilde{\sigma}_i(\mathbf{x}) \right] \right] + (1 - \nu_t) \left[\mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t} \sigma_{t-1}(\mathbf{x}) \right] \tag{D.1}$$

in which $\widetilde{\mu}_i(\mathbf{x})$ and $\widetilde{\sigma}_i(\mathbf{x})$ are obtained by replacing each noisy output of the meta-observations $y_{i,j}$ in the calculation of $\overline{\mu}_i(\mathbf{x})$ and $\overline{\sigma}_i(\mathbf{x})$ (7.1) by the (hypothetically available) noisy target function output observation at the corresponding input $\mathbf{x}_{i,j}$. Eq. (D.1) will serve as the bridge to connect the acquisition function (7.1) with the target function f in the subsequent theoretical analysis, which will be demonstrated in Appendix D.1.2. The next lemma shows that the difference between $\overline{\zeta}_t(\mathbf{x})$ (7.1) and $\widetilde{\zeta}_t(\mathbf{x})$ (D.1) is bounded $\forall \mathbf{x} \in \mathcal{D}$, whose proof is given in Appendix D.1.1.

Lemma D.1. *Let $\delta \in (0, 1)$. Suppose the RM-GP-UCB algorithm is run with parameters $\nu_t \in [0, 1]\, \forall t \geq 1$, and $\omega_i \geq 0$ for $i = 1, \ldots, M$ and $\sum_{i=1,\ldots,M} \omega_i = 1$.*

Then with probability $\geq 1 - \delta/3$,

$$\left|\overline{\zeta}_t(\mathbf{x}) - \widetilde{\zeta}_t(\mathbf{x})\right| \leq \nu_t \alpha \qquad \forall \mathbf{x} \in \mathcal{D}$$

in which

$$\alpha \triangleq \sum_{i=1}^{M} \omega_i \alpha_i,$$

$$\alpha_i \triangleq \frac{N_i}{\sigma^2}\left(2\sqrt{N_i}\sqrt{2\sigma^2 \log\frac{6N_i}{\delta}} + d_i\sqrt{N_i}\right).$$

Next, we need the following lemma that shows upper and lower bounds on the target function values, which follows from Gaussian concentration inequality and will be used extensively in the subsequent proofs.

Lemma D.2. *Let* $\delta \in (0,1)$ *and* $\beta_t = 2\log(|\mathcal{D}|t^2\pi^2/2\delta)$, *then*

$$|f(\mathbf{x}) - \mu_{t-1}(\mathbf{x})| \leq \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathcal{D},\ t \geq 1$$

which holds with probability $\geq 1 - \delta/3$. *Furthermore, let* $\tau = 2\log(3|\mathcal{D}|M/\delta)$, *we get*

$$|f(\mathbf{x}) - \widetilde{\mu}_i(\mathbf{x})| \leq \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathcal{D},\ i = 1, ..., M$$

which also holds with probability $\geq 1 - \delta/3$.

Note that $\widetilde{\mu}_i(\mathbf{x})$ and $\widetilde{\sigma}_i(\mathbf{x})$ are defined in (D.1), and Lemma D.2 can be proven with a slight modification to Lemma 5.1 of Srinivas et al. (2010).

D.1.1 Proof of Lemma D.1

Let $\mathbf{K}_i = [k(\mathbf{x}_{i,j}, \mathbf{x}_{i,j'})]_{j,j'=1,...,N_i}$ represent the Gram matrix corresponding to the inputs of the meta-observations from meta-task i, and $\mathbf{k}_i = [k(\mathbf{x}_{i,j}, \mathbf{x})]_{j=1,...,N_i}^{\top}$. Denote by $\lambda_j[\mathbf{A}]$ the j-th eigenvalue of matrix $\mathbf{A}$.

Firstly, we need the following lemma proving an upper bound on Frobenius norm:

Lemma D.3.

$$\left\|\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\right\|_F \leq \frac{\sqrt{N_i}}{\sigma^2}.$$

Proof.

$$\begin{aligned}
\left\|\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\right\|_F &\overset{\text{(a)}}{=} \sqrt{\text{Tr}\left(\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\left(\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\right)^T\right)} \\
&\overset{\text{(b)}}{=} \sqrt{\text{Tr}\left(\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\right)} \\
&\overset{\text{(c)}}{=} \sqrt{\sum_{j=1}^{N_i} \lambda_j\left[\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\right]} \\
&= \sqrt{\sum_{j=1}^{N_i}\left(\lambda_j\left[\left(\mathbf{K}_i + \sigma^2 I\right)^{-1}\right]\right)^2} \\
&= \sqrt{\sum_{j=1}^{N_i}\left(\frac{1}{\lambda_j\left[\mathbf{K}_i + \sigma^2 I\right]}\right)^2} \\
&= \sqrt{\sum_{j=1}^{N_i}\left(\frac{1}{\lambda_j\left[\mathbf{K}_i\right] + \sigma^2}\right)^2} \\
&\overset{\text{(d)}}{\leq} \sqrt{\sum_{j=1}^{N_i}\frac{1}{(\sigma^2)^2}} = \frac{\sqrt{N_i}}{\sigma^2}
\end{aligned}$$

in which (a) results from the definition of matrix Frobenius norm, (b) follows since $\mathbf{K}_i + \sigma^2 I$ (hence its inverse) is symmetric, (c) holds since the trace of a matrix is equal to the sum of its eigenvalues, the ensuing equalities make use of several identities of matrix eigenvalues. (d) follows because all eigenvalues of $\mathbf{K}_i$ are non-negative since $\mathbf{K}_i$ is positive semi-definite (because the kernel k is positive semi-definite). □

Next, define $\overline{\mathbf{f}}_i = [f_i(\mathbf{x}_{i,j})]_{j=1,\ldots,N_i}$ (in which $f_i(\mathbf{x}_{i,j})$ represents the value of meta-function i at input $\mathbf{x}_{i,j}$), and $\widetilde{\mathbf{f}}_i = [f(\mathbf{x}_{i,j})]_{j=1,\ldots,N_i}$ (in which $f(\mathbf{x}_{i,j})$

represents the value of target function at input $\mathbf{x}_{i,j}$). Similarly, define $\overline{\mathbf{y}}_i = [y_{i,j}]_{j=1,\ldots,N_i}$ (in which $y_{i,j}$ represents the noisy output observation of meta-task i at input $\mathbf{x}_{i,j}$), and $\widetilde{\mathbf{y}}_i = [y(\mathbf{x}_{i,j})]_{j=1,\ldots,N_i}$ (in which $y(\mathbf{x}_{i,j})$ represents the hypothetically observed noisy output observation of the target function at input $\mathbf{x}_{i,j}$). With these definitions, the next lemma shows upper bounds on the distance between $\overline{\mathbf{y}}_i$ and $\overline{\mathbf{f}}_i$, as well as that distance between $\widetilde{\mathbf{y}}_i$ and $\widetilde{\mathbf{f}}_i$.

Lemma D.4. *With probability* $\geq 1 - \delta/3$,

$$\left\|\overline{\mathbf{y}}_i - \overline{\mathbf{f}}_i\right\|_2 \leq \sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}},$$
$$\left\|\widetilde{\mathbf{y}}_i - \widetilde{\mathbf{f}}_i\right\|_2 \leq \sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}}.$$

Proof. Following the same analysis as Lemma 5.1 of Srinivas et al. (2010), we have that for the standard Gaussian random variable $z \sim \mathcal{N}(0,1)$,

$$\mathbb{P}(|z| > c) \leq e^{-\frac{c^2}{2}}. \tag{D.2}$$

Since for each $j = 1, \ldots, N_i$, we have that $y_{i,j} - f_i(\mathbf{x}_{i,j}) \sim \mathcal{N}(0, \sigma^2)$ and that $y(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,j}) \sim \mathcal{N}(0, \sigma^2)$, which leads to the following,

$$\begin{aligned}
\mathbb{P}\left(\left|\frac{y_{i,j} - f_i(\mathbf{x}_{i,j})}{\sigma}\right| > \sqrt{2\log\frac{6N_i}{\delta}}\right) &= \mathbb{P}\left(\left|y_{i,j} - f_i(\mathbf{x}_{i,j})\right| > \sqrt{2\sigma^2\log\frac{6N_i}{\delta}}\right) \\
&\leq \frac{\delta}{6N_i}, \\
\mathbb{P}\left(\left|\frac{y(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,j})}{\sigma}\right| > \sqrt{2\log\frac{6N_i}{\delta}}\right) &= \mathbb{P}\left(\left|y(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,j})\right| > \sqrt{2\sigma^2\log\frac{6N_i}{\delta}}\right) \\
&\leq \frac{\delta}{6N_i}.
\end{aligned}$$

Taking a union bound over $j = 1, \ldots, N_i$ for each of the two equations above,

we have that for *all* $j = 1, \ldots, N_i$,

$$\begin{aligned}\left|y_{i,j} - f_i(\mathbf{x}_{i,j})\right| &\leq \sqrt{2\sigma^2 \log \frac{6N_i}{\delta}}, \\ \left|y(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,j})\right| &\leq \sqrt{2\sigma^2 \log \frac{6N_i}{\delta}},\end{aligned}$$

both of which hold with probability $\geq 1 - \delta/6$. Therefore, with probability $\geq 1 - \delta/6$,

$$\begin{aligned}\left\|\overline{\mathbf{y}}_i - \overline{\mathbf{f}}_i\right\|_2 &= \sqrt{\sum_{j=1}^{N_i} \left|y_{i,j} - f_i(\mathbf{x}_{i,j})\right|^2} \leq \sqrt{\sum_{j=1}^{N_i} 2\sigma^2 \log \frac{6N_i}{\delta}} \\ &\leq \sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}}.\end{aligned} \tag{D.3}$$

Repeating the procedure above leads to

$$\left\|\widetilde{\mathbf{y}}_i - \widetilde{\mathbf{f}}_i\right\|_2 \leq \sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}} \tag{D.4}$$

which also holds with probability $\geq 1 - \delta/6$. Taking a union bound over equations (D.3) and (D.4) completes the proof. □

With these supporting lemmas, Lemma D.1 can be proven as follows:

$$\begin{aligned}&\|\overline{\zeta}_t(\mathbf{x}) - \widetilde{\zeta}_t(\mathbf{x})\| \\ &= \left|\nu_t \left[\sum_{i=1}^{M} \omega_i[\overline{\mu}_i(\mathbf{x}) + \sqrt{\tau}\overline{\sigma}_i(\mathbf{x})]\right] - \nu_t \left[\sum_{i=1}^{M} \omega_i[\widetilde{\mu}_i(\mathbf{x}) + \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x})]\right]\right| \\ &\overset{(a)}{=} \left|\nu_t \sum_{i=1}^{M} \omega_i[\overline{\mu}_i(\mathbf{x}) - \widetilde{\mu}_i(\mathbf{x})]\right| \\ &\leq \nu_t \sum_{i=1}^{M} \omega_i \left|\overline{\mu}_i(\mathbf{x}) - \widetilde{\mu}_i(\mathbf{x})\right| \\ &\leq \nu_t \sum_{i=1}^{M} \omega_i \left|\mathbf{k}_i(\mathbf{x})^\top (\mathbf{K}_i + \sigma^2 I)^{-1}(\overline{\mathbf{y}}_i - \widetilde{\mathbf{y}}_i)\right|\end{aligned} \tag{D.5}$$

$$
\begin{aligned}
&\overset{\text{(b)}}{\leq} \nu_t \sum_{i=1}^{M} \omega_i \left\|\mathbf{k}_i(\mathbf{x})\right\|_2 \left\|(\mathbf{K}_i + \sigma^2 I)^{-1}\right\|_F \|\overline{\mathbf{y}}_i - \widetilde{\mathbf{y}}_i\|_2 \\
&\overset{\text{(c)}}{\leq} \nu_t \sum_{i=1}^{M} \omega_i \left\|\mathbf{k}_i(\mathbf{x})\right\|_2 \frac{\sqrt{N_i}}{\sigma^2} \|\overline{\mathbf{y}}_i - \widetilde{\mathbf{y}}_i\|_2 \\
&\overset{\text{(d)}}{\leq} \nu_t \sum_{i=1}^{M} \omega_i \sqrt{N_i} \frac{\sqrt{N_i}}{\sigma^2} \|\overline{\mathbf{y}}_i - \widetilde{\mathbf{y}}_i\|_2 \\
&\leq \nu_t \sum_{i=1}^{M} \omega_i \frac{N_i}{\sigma^2} \left\|\overline{\mathbf{y}}_i - \overline{\mathbf{f}}_i + \overline{\mathbf{f}}_i - \widetilde{\mathbf{f}}_i + \widetilde{\mathbf{f}}_i - \widetilde{\mathbf{y}}_i\right\|_2 \\
&\leq \nu_t \sum_{i=1}^{M} \omega_i \frac{N_i}{\sigma^2} \left[\left\|\overline{\mathbf{y}}_i - \overline{\mathbf{f}}_i\right\|_2 + \left\|\overline{\mathbf{f}}_i - \widetilde{\mathbf{f}}_i\right\|_2 + \left\|\widetilde{\mathbf{f}}_i - \widetilde{\mathbf{y}}_i\right\|_2\right] \\
&\overset{\text{(e)}}{\leq} \nu_t \sum_{i=1}^{M} \omega_i \frac{N_i}{\sigma^2} \left(2\sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}} + \left\|\overline{\mathbf{f}}_i - \widetilde{\mathbf{f}}_i\right\|_2\right) \\
&= \nu_t \sum_{i=1}^{M} \omega_i \frac{N_i}{\sigma^2} \left(2\sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}} + \sqrt{\sum_{j=1}^{N_i} \left(f_i(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,j})\right)^2}\right) \\
&\overset{\text{(f)}}{\leq} \nu_t \sum_{i=1}^{M} \omega_i \frac{N_i}{\sigma^2} \left(2\sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}} + d_i\sqrt{N_i}\right) \\
&\triangleq \nu_t \alpha
\end{aligned}
\tag{D.6}
$$

which holds with probability $\geq 1 - \delta/3$. (a) holds because $\overline{\sigma}_i(\mathbf{x}) = \widetilde{\sigma}_i(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{D}$, since we have assumed that f and all f_i's are sampled from the same GP with kernel k and the posterior standard deviation only depends on the input locations, and is independent of the corresponding output responses; (b) follows from Cauchy-Schwarz inequality, (c) follows from Lemma D.3, (d) results from the assumption that $k(\mathbf{x}, \mathbf{x}') \leq 1$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$, (e) follows from Lemma D.4, (f) is obtained from the definition of the function gap: $d_i \triangleq \max_{j=1,\ldots,N_i} \left|f(\mathbf{x}_{i,j}) - f_i(\mathbf{x}_{i,j})\right|$ for $i = 1, \ldots, M$. This completes the proof of Lemma D.1.

D.1.2 Proof of Theorem 7.1

To begin with, we need the following lemma showing a high-probability upper bound on the global maximum of the target function.

Lemma D.5. *Given $\delta \in (0,1)$. Let $\mathbf{x}^*$ denote a global maximizer of the target function f, and α be as defined in Lemma D.1. Suppose the RM-GP-UCB algorithm is run with the parameter $\nu_t \in [0,1]$ for all $t \geq 1$. Then, with probability $\geq 1-\delta$,*

$$f(\mathbf{x}^*) \leq \overline{\zeta}_t(\mathbf{x}_t) + \nu_t \alpha \qquad \forall t \geq 1.$$

Proof. Firstly, as a result of Lemma D.2, at any iteration $t \geq 1$ and for all $\mathbf{x} \in \mathcal{D}$, we have that with probability $\geq 1-\delta/3-\delta/3$, $\widetilde{\zeta}_t(\mathbf{x})$ is an upper bound on $f(\mathbf{x})$:

$$\begin{aligned} \widetilde{\zeta}_t(\mathbf{x}) - f(\mathbf{x}) &= \widetilde{\zeta}_t(\mathbf{x}) - \left[\nu_t \sum_{i=1}^{M} \omega_i f(\mathbf{x}) + (1-\eta_t) f(\mathbf{x})\right] \\ &= \nu_t \sum_{i=1}^{M} \omega_i \left[\widetilde{\mu}_i(\mathbf{x}) + \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x}) - f(\mathbf{x})\right] + \\ &\quad (1-\nu_t)\left[\mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}) - f(\mathbf{x})\right] \geq 0. \end{aligned} \tag{D.7}$$

Therefore, with probability $\geq 1-\delta/3-\delta/3-\delta/3$,

$$f(\mathbf{x}^*) \overset{\text{(a)}}{\leq} \widetilde{\zeta}_t(\mathbf{x}^*) \overset{\text{(b)}}{\leq} \overline{\zeta}_t(\mathbf{x}^*) + \nu_t\alpha \overset{\text{(c)}}{\leq} \overline{\zeta}_t(\mathbf{x}_t) + \nu_t\alpha \tag{D.8}$$

in which (a) results from (D.7), (b) is obtained via Lemma D.1, and (c) follows from the policy for selecting $\mathbf{x}_t$, i.e., by maximizing (7.1). This completes the proof. □

Subsequently, we can show a high-probability upper bound on the instantaneous regret with the following lemma .

Lemma D.6. *Given $\delta \in (0,1)$. Let α be as defined in Lemma D.1. Suppose the RM-GP-UCB algorithm is run with the parameters β_t, τ and ν_t. Then, with probability $\geq 1-\delta$,*

$$r_t \leq 2\nu_t(\alpha + \sqrt{\tau}) + 2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t).$$

Proof. The instantaneous regret can be upper-bounded by

$$\begin{aligned}
r_t &= f(\mathbf{x}^*) - f(\mathbf{x}_t) \overset{\text{(a)}}{\leq} \overline{\zeta}_t(\mathbf{x}_t) + \nu_t\alpha - f(\mathbf{x}_t) \\
&\leq \underline{\overline{\zeta}_t(\mathbf{x}_t) - \widetilde{\zeta}_t(\mathbf{x}_t)} + \widetilde{\zeta}_t(\mathbf{x}_t) - f(\mathbf{x}_t) + \nu_t\alpha \\
&\overset{\text{(b)}}{\leq} \nu_t\alpha + \nu_t \sum_{i=1}^{M} \omega_i \left[\widetilde{u}_i(\mathbf{x}_t) + \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x}_t)\right] + (1-\nu_t)\left[u_{t-1}(\mathbf{x}_t) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)\right] \\
&\qquad - f(\mathbf{x}_t) + \nu_t\alpha \\
&= \nu_t\alpha + \nu_t \sum_{i=1}^{M} \omega_i \left[\widetilde{u}_i(\mathbf{x}_t) + \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x}_t)\right] + (1-\nu_t)\left[u_{t-1}(\mathbf{x}_t) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)\right] \\
&\qquad - \left[\nu_t \sum_{i=1}^{M} \omega_i f(\mathbf{x}_t) + (1-\nu_t) f(\mathbf{x}_t)\right] + \nu_t\alpha \\
&\leq \nu_t\alpha + \nu_t \sum_{i=1}^{M} \omega_i \left[\widetilde{u}_i(\mathbf{x}_t) - f(\mathbf{x}_t)\right] + \nu_t \sum_{i=1}^{M} \omega_i \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x}_t) \\
&\qquad + (1-\nu_t)\left[u_{t-1}(\mathbf{x}_t) - f(\mathbf{x}_t)\right] + (1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t) + \nu_t\alpha \\
&\overset{\text{(c)}}{\leq} 2\nu_t\alpha + 2\nu_t \sum_{i=1}^{M} \omega_i \sqrt{\tau}\widetilde{\sigma}_i(\mathbf{x}_t) + 2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t) \\
&\overset{\text{(d)}}{\leq} 2\nu_t\alpha + 2\nu_t\sqrt{\tau} + 2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t) \\
&= 2\nu_t(\alpha + \sqrt{\tau}) + 2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)
\end{aligned} \tag{D.9}$$

which holds with probability $\geq 1-\delta$. (a) follows from Lemma D.5, (b) results from Lemma D.1 as well as the definition of $\widetilde{\zeta}_t(\mathbf{x}_t)$ (D.1), (c) is a result of Lemma D.2, and (d) follows because $\widetilde{\sigma}_i(\mathbf{x}_t) \leq 1$ for all $\mathbf{x}_t \in \mathcal{D}$, which can be easily verified using the formula of the GP posterior variance (2.1) and the

assumption that $k(\mathbf{x}, \mathbf{x}') \leq 1$ for all $\mathbf{x}, \mathbf{x}' \in \mathcal{D}$. □

Next, we need to connect the second term from Lemma D.6 with the information gain. The following lemma, which is Lemma 5.3 of Srinivas et al. (2010), defines the information gain on the target function from any set of observations.

Lemma D.7. *Let $\mathbf{f}_T$ and $\mathbf{y}_T$ denote the set of function values and noisy observations of the target function respectively after T iterations. Then, the information gain about f from the first T observations can be expressed as*

$$I(\mathbf{y}_T; \mathbf{f}_T) = \frac{1}{2} \sum_{t=1}^{T} \log \left[1 + \sigma^{-2} \sigma_{t-1}^2(\mathbf{x}_t)\right].$$

Subsequently, we can upper bound the second term from Lemma D.6 (summed from iterations 1 to T) by the maximum information gain via the following lemma.

Lemma D.8. *Suppose the RM-GP-UCB algorithm is run with the parameters β_t $\forall t \geq 1$ and a non-increasing sequence $\nu_t \in [0, 1]$ $\forall t \geq 1$. Define the maximum information gain as $\gamma_T = \max_{A \in \mathcal{D}, |A| = T} I(\mathbf{y}_A; \mathbf{f}_A)$ in which $\mathbf{f}_A$ and $\mathbf{y}_A$ represent the function values and noisy observations from a set A of inputs of size T. Then,*

$$\sum_{t=1}^{T} \left[2(1 - \nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)\right]^2 \leq (1 - \nu_T)^2 C_1 \beta_T \gamma_T$$

in which $C_1 \triangleq \frac{8}{\log(1+\sigma^{-2})}$.

Proof. Each term inside the summation can be upper-bounded by

$$\begin{aligned}
4(1 - \nu_t)^2 \beta_t \sigma_{t-1}^2(\mathbf{x}_t) &\overset{(a)}{\leq} 4(1 - \nu_T)^2 \beta_T \sigma^2 \left(\sigma^{-2} \sigma_{t-1}^2(\mathbf{x}_t)\right) \\
&\overset{(b)}{\leq} 4(1 - \nu_T)^2 \beta_T \sigma^2 \left(\frac{\sigma^{-2}}{\log(1 + \sigma^{-2})} \log\left(1 + \sigma^{-2} \sigma_{t-1}^2(\mathbf{x}_t)\right)\right) \\
&= (1 - \nu_T)^2 \beta_T \frac{8}{\log(1 + \sigma^{-2})} \left[\frac{1}{2} \log\left(1 + \sigma^{-2} \sigma_{t-1}^2(\mathbf{x}_t)\right)\right]
\end{aligned} \tag{D.10}$$

in which (a) follows since β_t is non-decreasing in t and ν_t is non-increasing in t, (b) follows since $\sigma^{-2}x \leq \frac{\sigma^{-2}}{\log(1+\sigma^{-2})}\log(1+\sigma^{-2}x)$ for all $x \in (0,1]$ and $\sigma^2_{t-1}(\mathbf{x}_t) \in (0,1]$.

As a result, the summation can be decomposed as

$$\begin{aligned}
&\sum_{t=1}^{T}\left[2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)\right]^2 \\
&\quad\overset{\text{(a)}}{\leq}(1-\nu_T)^2\beta_T\frac{8}{\log(1+\sigma^{-2})}\sum_{t=1}^{T}\left[\frac{1}{2}\log\left(1+\sigma^{-2}\sigma^2_{t-1}(\mathbf{x}_t)\right)\right] \\
&\quad\overset{\text{(b)}}{=}(1-\nu_T)^2\beta_T\frac{8}{\log(1+\sigma^{-2})}I(\mathbf{y}_T;\mathbf{f}_T) \\
&\quad\overset{\text{(c)}}{\leq}(1-\nu_T)^2C_1\beta_T\gamma_T
\end{aligned}$$

in which (a) results from (D.10), (b) follows from Lemma D.7, and (c) is obtained by making use of the definition of C_1 and γ_T.

□

Finally, an upper bound on the cumulative regret follows from combining these supporting lemmas:

$$\begin{aligned}
R_T &= \sum_{t=1}^{T} r_t \overset{\text{(a)}}{\leq} \sum_{t=1}^{T}\left[2\nu_t(\alpha+\sqrt{\tau})+2\left(1-\nu_t\right)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)\right] \\
&= 2(\alpha+\sqrt{\tau})\sum_{t=1}^{T}\nu_t+\sum_{t=1}^{T}2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t) \\
&\overset{\text{(b)}}{\leq} 2(\alpha+\sqrt{\tau})\sum_{t=1}^{T}\nu_t+\sqrt{T}\sqrt{\sum_{t=1}^{T}\left[2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)\right]^2} \\
&\overset{\text{(c)}}{\leq} 2(\alpha+\sqrt{\tau})\sum_{t=1}^{T}\nu_t+\sqrt{C_1T(1-\nu_T)^2\beta_T\gamma_T}
\end{aligned} \tag{D.11}$$

which holds with probability $\geq 1-\delta$. (a) is a result of Lemma D.6, (b) follows from Cauchy-Schwarz inequality, and (c) is obtained using Lemma D.8. This completes the proof.

D.1.3 Meta-tasks Can Improve the Convergence by Accelerating Exploration

Here, we utilize the analysis in Appendix D.1.2 to illustrate how the meta-tasks (if similar to the target task) can help RM-GP-UCB obtain a better regret bound than standard GP-UCB in the early stage of the algorithm. For simplicity, we focus on the most favorable scenario where all meta-functions have equal values to the target function at their corresponding input locations, i.e., all function gaps are 0: $d_i \triangleq \max_{j=1,\ldots,N_i} \left|f(\mathbf{x}_{i,j}) - f_i(\mathbf{x}_{i,j})\right| = 0, \forall i = 1, \ldots, M$. Although not realistic, this scenario is useful for illustrating how the meta-tasks help our RM-GP-UCB algorithm achieve a better convergence at the initial stage.

In this case, according to the definition of $\widetilde{\zeta}_t$ (D.1) and $\overline{\zeta}_t$ (7.1), we have that $\widetilde{\zeta}_t(\mathbf{x}) = \overline{\zeta}_t(\mathbf{x}), \forall \mathbf{x} \in \mathcal{D}, t \geq 1$. As a result, the analysis of (D.8) in the proof of Lemma D.5 can be similarly applied, yielding:

$$f(\mathbf{x}^*) \leq \widetilde{\zeta}_t(\mathbf{x}^*) = \overline{\zeta}_t(\mathbf{x}^*) \leq \overline{\zeta}_t(\mathbf{x}_t). \tag{D.12}$$

Next, we can re-analyze the instantaneous regret following similar steps to (D.9):

$$\begin{aligned} r_t &= f(\mathbf{x}^*) - f(\mathbf{x}_t) \leq \overline{\zeta}_t(\mathbf{x}_t) - f(\mathbf{x}_t) \\ &\leq 2\nu_t \sum_{i=1}^{M} \omega_i \sqrt{\tau}\overline{\sigma}_i(\mathbf{x}_t) + 2(1-\nu_t)\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t) \\ &= \underbrace{2\nu_t \left(\sum_{i=1}^{M} \omega_i \sqrt{\tau}\overline{\sigma}_i(\mathbf{x}_t) - \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t) \right)}_{A1} + \underbrace{2\sqrt{\beta_t}\sigma_{t-1}(\mathbf{x}_t)}_{A2}, \end{aligned} \tag{D.13}$$

in which some intermediate steps that are identical to those used in (D.9) have been omitted for simplicity. Note that term A_2 in (D.13) is identical to the upper bound on the instantaneous regret for standard GP-UCB (Srinivas et al., 2010).

Therefore, the meta-tasks affect the upper bound on the instantaneous regret through the term A_1.

Recall Corollary 7.1 has told us that we should choose $\nu_t \to 0$ as $t \to \infty$. In the initial stage of the algorithm when ν_t is large, the impact of A_1 on the regret of the algorithm is large. In this case, the meta-tasks improve the upper bound on the instantaneous regret (compared with standard GP-UCB) if $A_1 < 0$, that is:

$$\sum_{i=1}^{M} \omega_i \overline{\sigma}_i(\mathbf{x}_t) < \sqrt{\frac{\beta_t}{\tau}} \sigma_{t-1}(\mathbf{x}_t). \tag{D.14}$$

In other words, RM-GP-UCB converges faster than standard GP-UCB in the initial stage if the (weighted combination of) meta-tasks have smaller uncertainty (i.e., posterior standard deviation) at $\mathbf{x}_t$ compared with the target task (scaled by $\sqrt{\beta_t}/\tau$). Fortunately, in the early stage of the algorithm, this condition is highly likely to be satisfied: When the number of observations of the target task is small, the posterior standard deviation of the target GP posterior (i.e., RHS of Equation (D.14)) is usually large; therefore, Equation (D.14) is highly likely to be satisfied. This insight turns out to have an intuitive and elegant interpretation as well. In the initial stage of the standard GP-UCB algorithm, due to the lack of observations, the algorithm *has large uncertainty* regarding the objective function and hence tends to *explore*; however, the meta-tasks (assuming that they are similar to the target task) provides additional information for the algorithm, which *reduces the uncertainty* about the objective function and hence *decreases the requirement for initial exploration*. To summarize, in the initial stage, the meta-tasks, if similar to the target task, help RM-GP-UCB achieve smaller regret upper bound (hence converge faster) than GP-UCB by reducing the degree of exploration. In less favorable scenarios where the function gaps are nonzero (i.e., the meta-functions are not exactly equal to the target function), some amount of errors will be introduced to the upper bound on the instantaneous regret (D.13).

As a results, a positive error term will be added to the LHS of (D.14), making the theoretical condition for a faster convergence (D.14) harder to satisfy. At later stages where ν_t is already small and close to 0, the impact of the term A_1 is significantly diminished, thus allowing our RM-GP-UCB algorithm to converge to no regret at a similar rate to standard GP-UCB.

D.1.4 Proof of Lemma 7.1

From the definitions of $U_{t,i,j}$ and $L_{t,i,j}$ (7.2), and the fact that $L_{t,i,j} \leq f(\mathbf{x}_{i,j}) \leq U_{t,i,j}, \forall t, i, j$ with probability $\geq 1-\delta$ (Section 7.5.1), we have that

$$\begin{aligned} d_i &= \max_{j=1,\ldots,N_i} \left| f_i(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,j}) \right| \\ &\leq \max_{j=1,\ldots,N_i} \left[\max\{ \left| f_i(\mathbf{x}_{i,j}) - U_{t,i,j} \right|, \left| f_i(\mathbf{x}_{i,j}) - L_{t,i,j} \right| \} \right] \end{aligned} \tag{D.15}$$

which holds with probability $\geq 1-\delta, \forall\, i = 1, \ldots, M, \forall t \geq 1$. Next, we derive upper bounds on $\left| f_i(\mathbf{x}_{i,j}) - U_{t,i,j} \right|$ and $\left| f_i(\mathbf{x}_{i,j}) - L_{t,i,j} \right|$ that only consist of known or computable terms, such that the upper bounds on d_i can be efficiently calculated in practice.

Lemma D.9. *Let $\delta' \in (0,1)$. With probability $\geq 1-\delta'$, $\forall\, t \geq 1$, $\forall i, j$,*

$$\begin{aligned} \left| f_i(\mathbf{x}_{i,j}) - U_{t,i,j} \right| &\leq \sqrt{2\sigma^2 \log \frac{2\sum_{i=1}^M N_i}{\delta'}} + \left| y_{i,j} - U_{t,i,j} \right|, \\ \left| f_i(\mathbf{x}_{i,j}) - L_{t,i,j} \right| &\leq \sqrt{2\sigma^2 \log \frac{2\sum_{i=1}^M N_i}{\delta'}} + \left| y_{i,j} - L_{t,i,j} \right|. \end{aligned}$$

Proof. To begin with, note that $f_i(\mathbf{x}_{i,j}) - y_{i,j} \sim \mathcal{N}(0, \sigma^2)$. Therefore, (D.2) suggests that

$$\mathbb{P}\left(\left| f_i(\mathbf{x}_{i,j}) - y_{i,j} \right| > \sigma \sqrt{2 \log \frac{2\sum_{i=1}^M N_i}{\delta'}} \right) \leq \frac{\delta'}{2\sum_{i=1}^M N_i} \tag{D.16}$$

which naturally leads to a high-probability upper bound on $\left|f_i(\mathbf{x}_{i,j}) - U_{t,i,j}\right|$:

$$\begin{aligned}\left|f_i(\mathbf{x}_{i,j}) - U_{t,i,j}\right| &= \left|f_i(\mathbf{x}_{i,j}) - y_{i,j} + y_{i,j} - U_{t,i,j}\right| \\ &\leq \left|f_i(\mathbf{x}_{i,j}) - y_{i,j}\right| + \left|y_{i,j} - U_{t,i,j}\right| \\ &\leq \sqrt{2\sigma^2 \log \frac{2\sum_{i=1}^{M} N_i}{\delta'}} + \left|y_{i,j} - U_{t,i,j}\right|\end{aligned} \tag{D.17}$$

which holds with probability $\geq 1 - \frac{\delta'}{2\sum_{i=1}^{M} N_i}$. Applying the same reasoning to $\left|f_i(\mathbf{x}_{i,j}) - L_{t,i,j}\right|$ results in a similar high-probability upper bound:

$$\left|f_i(\mathbf{x}_{i,j}) - L_{t,i,j}\right| \leq \sqrt{2\sigma^2 \log \frac{2\sum_{i=1}^{M} N_i}{\delta'}} + \left|y_{i,j} - L_{t,i,j}\right|. \tag{D.18}$$

Next, the proof is completed by taking a union bound over both $U_{t,i,j}$ and $L_{t,i,j}$, as well as all $\sum_{i=1}^{M} N_i$ observations of the meta-tasks. □

Finally, Lemma 7.1 follows by combining (D.15) and Lemma D.9.

D.1.5 Proof of Proposition 7.1

In iteration t, define $\overline{\alpha}_t$ by replacing d_i in α with $\overline{d}_{i,t}$:

$$\overline{\alpha}_t \triangleq \sum_{i=1}^{M} \omega_i \frac{N_i}{\sigma^2} \left(2\sqrt{N_i}\sqrt{2\sigma^2 \log \frac{6N_i}{\delta}} + \overline{d}_{i,t}\sqrt{N_i}\right). \tag{D.19}$$

Since according to Lemma 7.1, $d_i \leq \overline{d}_{i,t}\ \forall i = 1, \ldots, M, t \geq 1$ with probability $\geq 1 - \delta - \delta'$, we have that $\alpha \leq \overline{\alpha}_t\ \forall t \geq 1$, which also holds with probability $\geq 1 - \delta - \delta'$.

Therefore, Theorem 7.1 implies that, with probability $\geq 1 - \delta - \delta'$,

$$R_T \leq 2\sum_{t=1}^{T} \overline{\alpha}_t \nu_t + 2\sqrt{\tau}\sum_{t=1}^{T} \nu_t + \sqrt{C_1 T \left(1 - \nu_T\right)^2 \beta_T \gamma_T}. \tag{D.20}$$

In (D.20), only the underlined term depends on the the ω_i's. Define two column vectors $\overline{\boldsymbol{\alpha}} = [\overline{\alpha}_t]_{t=1,\dots,T}^\top$ and $\boldsymbol{\nu} = [\nu_t]_{t=1,\dots,T}^\top$. Then, the underlined term in (D.20) can be further decomposed as

$$2\sum_{t=1}^{T}\overline{\alpha}_t\nu_t \triangleq 2\overline{\boldsymbol{\alpha}}^\top\boldsymbol{\nu} \overset{\text{(a)}}{\leq} 2\|\overline{\boldsymbol{\alpha}}\|_2\|\boldsymbol{\nu}\|_2 \overset{\text{(b)}}{\leq} 2\|\overline{\boldsymbol{\alpha}}\|_1\|\boldsymbol{\nu}\|_1 \overset{\text{(c)}}{=} 2\underline{\sum_{t=1}^{T}\overline{\alpha}_t}\sum_{t=1}^{T}\nu_t \tag{D.21}$$

in which (a) results from Cauchy-Schwarz inequality, (b) follows because the L2 norm is upper-bounded by the L1 norm, and (c) is obtained because $\overline{\alpha}_t > 0, \nu_t \geq 0, \forall t \geq 1$.

In (D.21), the dependence on the ω_i's appears in the underlined term, which can be further decomposed as

$$\begin{aligned}\sum_{t=1}^{T}\overline{\alpha}_t &= \sum_{t=1}^{T}\left[\sum_{i=1}^{M}\omega_i\frac{N_i}{\sigma^2}\left(2\sqrt{N_i}\sqrt{2\sigma^2\log\frac{6N_i}{\delta}} + \overline{d}_{i,t}\sqrt{N_i}\right)\right] \\ &\triangleq \frac{1}{\sigma^2}\sum_{t=1}^{T}\left[\sum_{i=1}^{M}\omega_i l_{i,t}\right] \\ &\triangleq \frac{1}{\sigma^2}\sum_{t=1}^{T}\boldsymbol{\omega}^\top\boldsymbol{l}_t\end{aligned} \tag{D.22}$$

in which we have defined $\boldsymbol{\omega} \triangleq [\omega_i]_{i=1,\dots,M}$, $\boldsymbol{l}_t \triangleq [l_{i,t}]_{i=1,\dots,M}$, with

$$l_{i,t} \triangleq 2N_i^{\frac{3}{2}}\sqrt{2\sigma^2\log\frac{6N_i}{\delta}} + \overline{d}_{i,t}N_i^{\frac{3}{2}}. \tag{D.23}$$

Plugging (D.21) and (D.22) in to (D.20) completes the proof.

D.1.6 Derivation of Equation (7.4)

Recall that our objective is to minimize

$$\sum_{s=1}^{t-1}\boldsymbol{\omega}'^\top\boldsymbol{l}_s + \frac{1}{\eta}\sum_{i=1}^{M}\omega'_i\log\omega'_i$$

subject to the constraint that $\boldsymbol{\omega}'$ forms a probability simplex: $\sum_{i=1}^{M} \omega_i' = 1.0$ and $\omega_i' \geq 0$ for all $i = 1, \ldots, M$. Define the Lagrangian as

$$L(\boldsymbol{\omega}, \lambda) = \sum_{s=1}^{t-1} \boldsymbol{\omega}'^{\top} \boldsymbol{l}_s + \frac{1}{\eta} \sum_{i=1}^{M} \omega_i' \log \omega_i' + \lambda \left(1 - \sum_{i=1}^{M} \omega_i' \right). \tag{D.24}$$

Taking the derivative of $L(\boldsymbol{\omega}, \lambda)$ with respect to ω_i', we get

$$\frac{\partial L(\boldsymbol{\omega}, \lambda)}{\partial \omega_i'} = \sum_{s=1}^{t-1} l_{i,s} + \frac{1}{\eta} \left(\log \omega_i' + 1 \right) - \lambda. \tag{D.25}$$

Setting (D.25) to 0 gives us

$$\omega_i' = e^{\eta\lambda - 1} e^{-\eta \sum_{s=1}^{t-1} l_{i,s}} \propto e^{-\eta \sum_{s=1}^{t-1} l_{i,s}}. \tag{D.26}$$

Normalizing the ω_i''s for all $i = 1 \ldots, M$ to form a probability simplex leads to (7.4).

D.2 More Experimental Details and Results

In every experiment, the same set of random initializations are used for all methods to ensure fair comparisons. The kernel bandwidth parameter ρ in TAF is set to $\rho = 0.5$ in all experiments, but we have observed that other values of ρ (such as 0.1 and 0.9) lead to similar performances. $S = 500$ posterior samples are used to compute the ensemble weights in RGPE. All experiments are run on a server with 16 cores of Intel Xeon processor, 256G of RAM and 5 NVIDIA GTX1080 Ti GPUs.

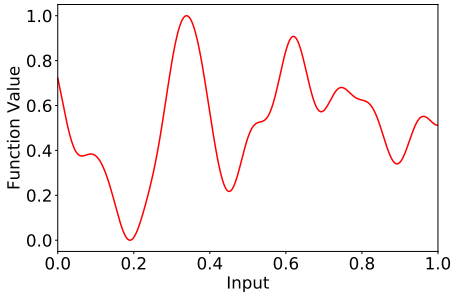


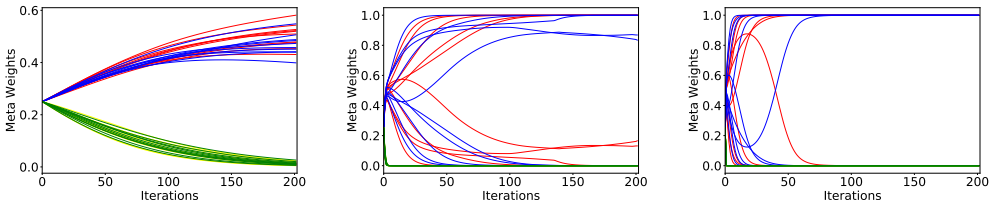
Figure D.1: An example synthetic function sampled from a GP.

D.2.1 Optimization of Synthetic Functions

D.2.1.1 Synthetic Functions Sampled from GPs

The objective functions are drawn from GP's with the Squared Exponential kernel (with the length scale of 0.05) from the domain $\mathcal{D} = [0, 1]$. Fig. D.1 shows an example of such synthetic functions. The meta-functions and meta-tasks are generated in the following way. To begin with, we fix the number of meta-tasks $M = 4$, the number of observations (input-output pairs) for each meta-task $N = N_i = 20$ for $i = 1 \dots M$, and the function gaps: $d_1 = d_2 = 0.05$, $d_3 = d_4 = 4.0$. For the i-th meta-task, firstly, N_i inputs are randomly drawn from the entire domain $\mathcal{D} = [0, 1]$. Then for each of the N_i inputs $\mathbf{x}_{i,j}$, a number is randomly drawn from $[-d_i, d_i]$, which is added to the value of the target function $f(\mathbf{x}_{i,j})$ to produce the corresponding function value of the meta-function $f_i(\mathbf{x}_{i,j})$. Subsequently, a zero-mean Gaussian noise (with a noise variance of 0.01) is added to $f_i(\mathbf{x}_{i,j})$, resulting in the corresponding output of the meta-observation $y_i(\mathbf{x}_{i,j})$. The above-mentioned procedure is repeated for each of the $M = 4$ meta-tasks. Note that according to the specified function gaps, meta-tasks 1 and 2 are relatively more similar to the target task, whereas meta-tasks 3 and 4 are dissimilar to the target task due to the larger function gaps.

Fig. D.2 plots the evolution of the meta-weights for each of the 4 meta-tasks in the experiments exploring the impact of η, i.e., corresponding to Fig. 7.1c in Section 7.6.1. These figures are used to demonstrate the observations that



(a) $\eta N^{3/2} = 0.01$. (b) $\eta N^{3/2} = 1.0$. (c) $\eta N^{3/2} = 5.0$.

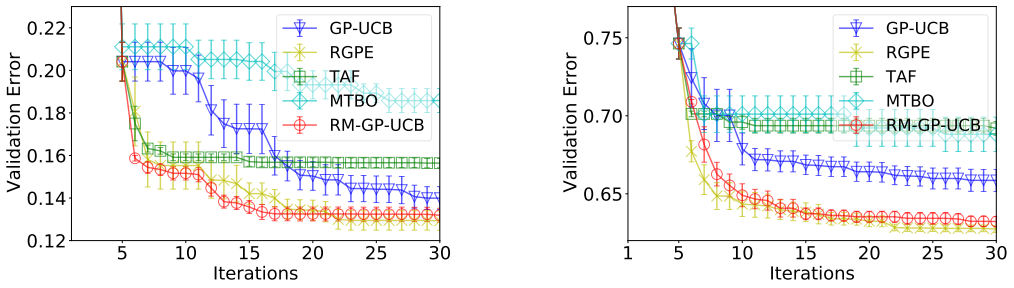
Figure D.2: Evolution of the meta-weights with different learning rate, η, for online meta-weight optimization in the synthetic experiments. In each figure, the red and blue curves represent the meta-weights of the two meta-tasks that are more similar to the target task (i.e., the first two meta-tasks), whereas the green and yellow curves correspond to the meta-weights of the other two dissimilar meta-tasks. Every color has 10 curves in each figure, which correspond to 10 independent runs of the algorithm with different random initializations.

overly large and excessively small values of η can both degrade the performance of RM-GP-UCB.

D.2.2 Real-world Experiments

Hyperparameter Tuning for Convolutional Neural Networks (CNNs). The MNIST, CIFAR-10 and CIFAR-100 datasets can all be directly downloaded using the Keras Python package[1], and the SVHN dataset can be downloaded from http://ufldl.stanford.edu/housenumbers/. The image pixel values are all normalized into the range $[0, 1]$. The CNN hyperparameters being optimized in this set of experiments are the learning rate, learning rate decay, and the L2 regularization parameter, all of which have the search space from 10^{-7} to 10^{-2}. Other than these hyperparameters, a common CNN architecture is used for all datasets, i.e., a CNN containing two convolutional layers (both with 32 filters and each filter has a size of 3×3) each of which is followed by a Max pooling layer (with a pooling size of 3×3), followed by two fully connected layers (both with 64 hidden units); all non-linear activations are ReLU. The size of the training set and validation set for the four datasets are: 60,000/10,000 for MNIST,

[1] https://keras.io/



(a) SVHN. (b) CIFAR-100.

Figure D.3: Best validation error of CNN (both averaged over 10 random initializations).

73,257/26,032 for SVHN, 50,000/10,000 for both CIFAR-10 and CIFAR-100. For the evaluation of a set of selected hyperparameters, the CNN model is trained using the RMSprop algorithm for 20 epochs, and the final validation error is used as the corresponding output observation. Fig. D.3 presents the results when the SVHN and CIFAR-100 datasets are used to produce the target functions.

Comparing Figs. 7.2a, 7.2b and Fig. D.3 shows that our RM-GP-UCB performs similarly to RGPE for the CIFAR-10, CIFAR-100 and SVHN datasets, and outperforms RGPE for MNIST. After inspection, we have found that this is because for the first three datasets (Fig. 7.2b and Fig. D.3), both RM-GP-UCB and RGPE assign most meta-weights to the same meta-task. On the other hand, for MNIST (Fig. 7.2a), RM-GP-UCB is able to assign most weights to SVHN which is indeed more similar to MNIST since they both contain images of digits. In contrast, RGPE mistakenly assigns more meta-weights to CIFAR-10. The reason is that RGPE chooses the weights based on how accurately each meta-task's GP surrogate predicts the pairwise ranking of the target observations (Section 3.4, second paragraph). However, for MNIST, most target observations have very similar values since the overall accuracy is very high due to the simplicity of the MNIST dataset. Therefore, the predicted pairwise rankings become unreliable, thus rendering the weights learned by RGPE inaccurate and deteriorating the performance.

Hyperparameter Tuning for CNNs Using the Omniglot Dataset. The Omniglot dataset can be downloaded from https://github.com/brendenlake/omniglot. The dataset consists of 50 alphabets, 30 from the background set and 20 from the evaluation set. Each alphabet includes a number of characters, and all alphabets combine to have 1623 characters. Every character only consists of 20 example images, each drawn by a different person. To perform one-shot classification, we use a Siamese neural network Koch et al. (2015), which takes two images as inputs and outputs a score indicating whether the pair of input images are predicted to be the same character. The evaluation metric we use in the experiment is 2-way validation error. That is, we compare a test image in the validation set with two other images, only one of which is the same character as the test image, and evaluate whether the Siamese network is able to output a higher predictive score for the correct image which is the same character; we do this using every test image, and use the percentage of errors as the 2-way validation error. In our setting, each task represents tuning 3 hyperparameters of the Siamese network (the same hyperparameters and ranges as the CNN experiments above) using one alphabet. For each task, we use 75% of the characters in the alphabet to produce the training set, and the remaining 25% to generate the validation set. We use 10 alphabets from the background set as 10 meta-tasks. For each meta-task, we generate 30 meta-observations by running BO (using GP-UCB) for 30 iterations. This in total produces $10 \times 30 = 300$ meta-observations. We use one of the alphabets from the evaluation set as the target task.

Hyperparameter Tuning for Support Vector Machines (SVMs). This benchmark dataset, originally introduced by Wistuba et al. (2015a) and can be downloaded from https://github.com/wistuba/TST, is created by performing hyperparameter tuning of SVM using 50 diverse datasets. 6 hyperparameters are tuned: 3 binary parameters indicating whether a linear, polynomial or radial basis function (RBF) kernel is used, the penalty parameter, the degree of the

polynomial kernel, and the bandwidth parameter for the RBF kernel. A fixed grid of hyperparameters of size 288 is created. For each dataset, every hyperparameter configuration on the grid is evaluated and the corresponding validation accuracy is recorded as the observed output of the objective function. In our experiments, each dataset corresponds to a task. We treat one of the 50 tasks as the target task, and the remaining tasks as 49 meta-tasks. For each meta-task, the meta-observations are produced by randomly sampling 50 points (hyperparameter configurations) from the grid. The results reported in the main text (Fig. 7.2d) are averaged over 25 trials, each trial treating a different task as the target task; for each trial/target task, we again average the results over 5 random initializations.

Human Activity Recognition (HAR). The dataset used in this experiment, which has also been used in the experiments in Chapters 5 and 6, can be downloaded from https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones. In this experiment of human activity prediction, each data instance (input-output pair) is characterized by a feature vector of length 561 and a label corresponding to one of the 6 activities. The SVM hyperparameters being optimized are the penalty parameter C (from 0.01 to 10) and the radial basis function (RBF) kernel coefficient γ (from 0.01 to 1). There are in total 7,352 data instances for the 21 subjects that are used to generate the meta-tasks, and 2,947 instances for the 9 subjects used for performance validation. For each subject, half of the instances are used as the training set, with the other half being used for validation.

Fig. D.4 plots the performances of each of the 9 subjects used for performance validation, in which RM-GP-UCB performs most consistently among all algorithms under comparison (summarized in Fig. 7.3a in the main text). Specifically, RGPE fails to outperform standard GP-UCB in Figs. D.4c, d, e, g and h, and TAF fails to perform better than standard GP-UCB in Figs. D.4e and h, whereas RM-GP-UCB fails to outperform GP-UCB only in Fig. D.4h.

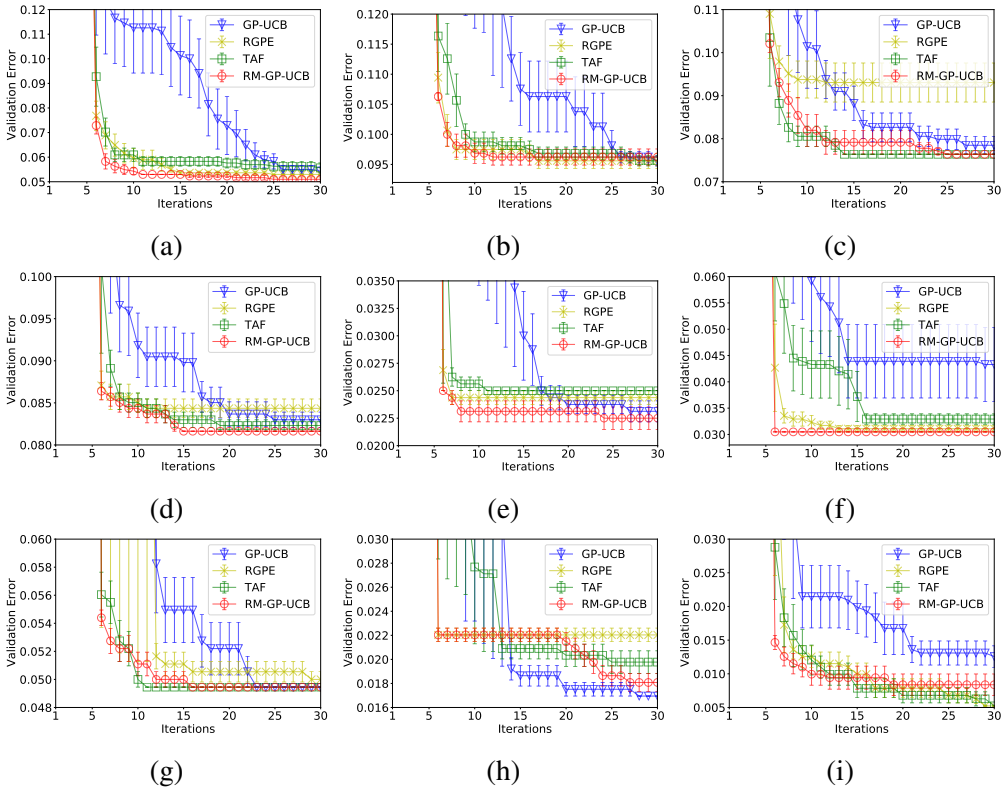
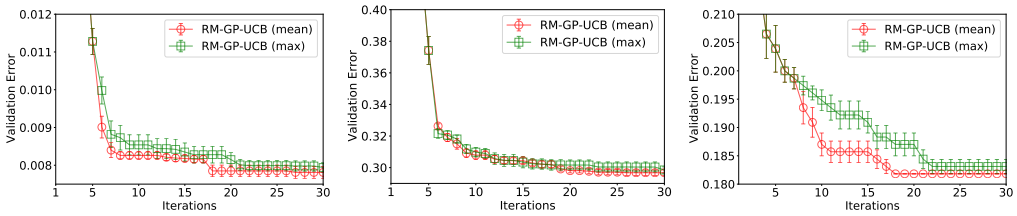


Figure D.4: Best validation error of SVM for human activity recognition for the 9 individual subjects (each averaged over 10 random initializations).

Non-stationary Bayesian Optimization. The clinical diagnosis dataset used in this experiment can be found at https://www.kaggle.com/uciml/pima-indians-diabetes-database. The hyperparameters of the logistic regression (LR) model being optimized are the batch size (20 to 60), the L2 regularization parameter (10^{-6} to 0.01) and the learning rate (0.01 to 0.1). The dataset represents a binary classification problem (whether a patient has diabetes or not), with each input instance consisting of 8 diagnostic features: number of pregnancies, plasma glucose concentration, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function, and age. The entire dataset consists of 768 data instances, among which 77 instances are set aside to measure the validation accuracy. The sizes of the 5 progressively growing training datasets (i.e., corresponding to the 4 meta-tasks and the target task respectively) are 138, 276, 414, 552, and 691.



(a) (b) (c)

Figure D.5: Impacts of using max vs empirical mean in estimating the upper bound on the function gaps, using the (a) MNIST, (b) CIFAR-10 and (c) clinical diagnosis experiments.

D.2.3 Impacts of Max vs Mean in Function Gap Estimation

Here we explore the impact of the choice between using $\max$ (the outer $\max$ operator over $j = 1, ..., N_i$) or the empirical mean in the estimated upper bound on the function gap (Lemma 7.1), as mentioned in the first paragraph of Section 7.6. Fig. D.5 plots the different performances using these two choices in the MNIST, CIFAR-10 and clinical diagnosis (non-stationary BO) experiments. The results show that the performance deficit resulting from the use of the $\max$ operator is marginal in some experiments (Fig. D.5a and b), whereas the difference can be larger in some other experiments (Fig. D.5c). Therefore, it is recommended to use the empirical mean when estimating the upper bound on the function gap in practice.

D.2.4 Scalability of Our RM-GP-UCB Algorithm

Here we demonstrate the scalability of our RM-GP-UCB algorithm. Firstly, we plot the runtime of different algorithms in the non-stationary BO (diabetes diagnosis) experiment. We have chosen to use this experiment since its scale is not excessively large such that it is still computationally feasible for the MTBO algorithm. As shown in Fig. D.6, our RM-GP-UCB algorithm, as well as RGPE and TAF, runs much faster than the MTBO algorithm. Next, we demonstrate that our algorithm can be applied to experiments with a very large scale, and

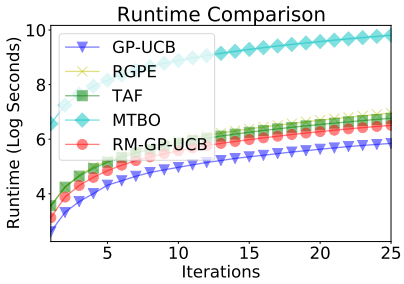
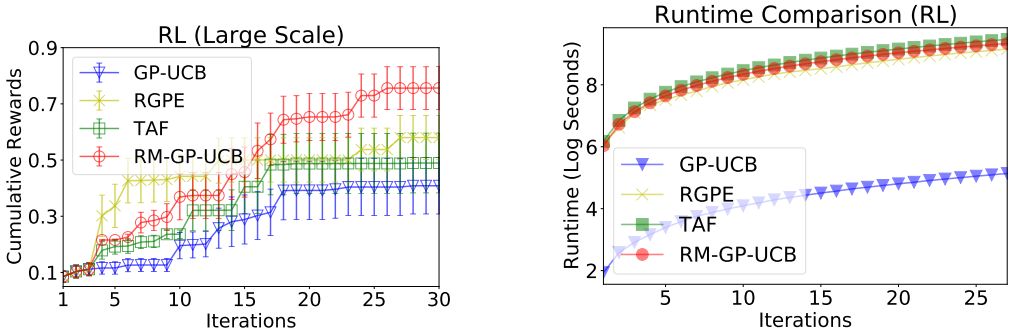


Figure D.6: Runtime of different algorithms in the non-stationary BO (clinical diagnosis) experiment.



(a) Cumulative rewards.

(b) Runtime.

Figure D.7: Results demonstrating that our algorithm can be applied to experiments of a very large scale, using a larger version of the RL experiment (with $60 \times 130 = 7800$ meta-observations).

still performs competitively. Specifically, we construct a much larger version of the experiment on policy search for RL, with 60 meta-tasks each containing 130 meta-observations. Fig. D.7a and b plot the performance and runtime in this large-scale experiment. Consistent with Fig. 7.3c in the main text, our RM-GP-UCB algorithm still performs the best among all algorithms.

Appendix E

Appendix for Chapter 8

E.1 Background on the GP-MW Algorithm

When $\mathcal{A}$ (the attacker) adopts the GP-MW algorithm as the level-0 strategy, after iteration t of the repeated game, $\mathcal{A}$ calculates the updated value of the GP-UCB acquisition function at every input in its entire domain $\mathcal{X}_1$ (while fixing the defender's input $\mathbf{x}_2$ at the value selected in iteration t: $\mathbf{x}_{2,t}$), plugs in the (negative) GP-UCB values as the loss vector (with the length of the vector being equal to the size of its domain: $|\mathcal{X}_1|$) in the widely used multiplicative-weight online learning algorithm to update the randomized/mixed strategy $\mathcal{P}^0_{1,t+1}$. Subsequently, the resulting updated distribution will be used to sample $\mathcal{A}$'s action in the next iteration $t+1$, i.e., $\mathbf{x}_{1,t+1} \sim \mathcal{P}^0_{1,t+1}$. Note that the proof of Theorem 8.1 results from a slight modification to the proof of GP-MW (Sessa et al., 2019), i.e., the work of (Sessa et al., 2019) has assumed that the payoff function has bounded norm in a reproducing kernel Hilbert space, whereas we assume that the payoff function is sampled from a GP. Both assumptions are commonly used in the analysis of BO algorithms. Refer to the work of (Sessa et al., 2019) for more details about the GP-MW algorithm.

E.2 Extension to Games Involving More than Two Agents

The R2-B2, as well as R2-B2-Lite, algorithm can be extended to repeated games involving more than two ($M > 2$) agents. A motivating scenario for this type of games with $M > 2$ agents is MARL, in which every individual agent attempts to maximize its own return (payoff). Here, we use $\mathcal{A}_1, \ldots, \mathcal{A}_M$ to represent the M agents.

Level-$k = 0$ Strategy. The extension of level-0 reasoning is trivial since level-0 strategies are agnostic with respect to the other agent's action selection strategies, and can thus treat all other agents as a single collective agent. As a result, if GP-MW is adopted as the level-0 strategy, the theoretical guarantee of Theorem 8.1 still holds.

Level-$k = 1$ Strategy. If the agent $\mathcal{A}_1$ thinks that all other agents $(\mathcal{A}_2, \ldots, \mathcal{A}_M)$ reason at level 0 and knows the level-0 strategies of all other agents, $\mathcal{A}_1$ can reason at level 1 by:

$$\mathbf{x}_{1,t}^1 = \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \mathbb{E}_{\mathbf{x}_{2,t}^0, \ldots, \mathbf{x}_{M,t}^0} \left[\alpha_{1,t}(\mathbf{x}_1, \mathbf{x}_{2,t}^0, \ldots, \mathbf{x}_{M,t}^0) \right], \tag{E.1}$$

in which the expectation is taken over the level-0 strategies of all other agents $\mathcal{A}_2, \ldots, \mathcal{A}_M$. R2-B2-Lite can also be applied:

$$\mathbf{x}_{1,t}^1 = \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \alpha_{1,t}(\mathbf{x}_1, \widetilde{\mathbf{x}}_{2,t}^0, \ldots, \widetilde{\mathbf{x}}_{M,t}^0), \tag{E.2}$$

in which $\widetilde{\mathbf{x}}_{2,t}^0, \ldots, \widetilde{\mathbf{x}}_{M,t}^0$ are sampled from the corresponding level-0 strategies of agents $\mathcal{A}_2, \ldots, \mathcal{A}_M$.

For level-1 reasoning, the actions of all other agents can be viewed as the joint action of a single collective agent, whose level-0 strategy (action distribution) factorizes across different agents. As a result, the theoretical guarantees of

Theorems 8.2 and 8.4 are still valid.

Level-$k \geq 2$ Strategy. Level-$k \geq 2$ reasoning with $M > 2$ agents is significantly more complicated than the two-agent setting, mainly due to the fact that the other agents may not reason at the same level. For simplicity, we consider the scenario in which the agent $\mathcal{A}_1$ reasons at level 2, and thus all other agents reason at either level 1 or 0. This is a common scenario since as discussed in Section 8.3.1.3 and will be explained at the end of this section, the agents have a strong tendency to reason at lower levels in the setting with $M > 2$ agents. Without loss of generality, we assume that agents 2 to M_0 reason at level 0, and agents $M_0 + 1$ to M reason at level 1 (by following the strategy of (E.1)). In this case, the level-2 action of agent $\mathcal{A}_1$ is selected by best-responding to the corresponding strategy of each of the other agents:

$$\mathbf{x}_{1,t}^2 = \arg\max_{\mathbf{x}_1 \in \mathcal{X}_1} \mathbb{E}_{\mathbf{x}_{2,t}^0, \ldots, \mathbf{x}_{M_0,t}^0} \left[\alpha_{1,t}(\mathbf{x}_1, \mathbf{x}_{2,t}^0, \ldots, \mathbf{x}_{M_0,t}^0, \mathbf{x}_{M_0+1,t}^1, \ldots, \mathbf{x}_{M,t}^1) \right]. \tag{E.3}$$

Specifically, the level-1 actions of those agents reasoning at level 1 ($\mathbf{x}_{M_0+1,t}^1, \ldots, \mathbf{x}_{M,t}^1$) can be calculated using (E.1), and the expectation in (E.3) is taken with respect to the level-0 strategies of those agents reasoning at level 0 ($\mathbf{x}_{2,t}^0, \ldots, \mathbf{x}_{M_0,t}^0$). Interestingly, the level-2 reasoning strategy of (E.3) enjoys the same regret upper bound as shown in Theorem 8.2 or Theorem 8.3, depending on whether there exists level-0 agents (see the detailed explanation and the proof in Appendix E.5). Unfortunately, the complexity of reasoning at levels $k \geq 3$ grows excessively. Firstly, every other agent reasoning at a lower level $k \geq 2$ may best-respond to the other agents in multiple ways. For example, if there are $M = 3$ agents in the environment and agent $\mathcal{A}_1$ reasons at level 2, $\mathcal{A}_1$ might choose its level-2 action in three different ways, with the corresponding reasoning levels of the 3 agents being $[2, 1, 1]$, $[2, 1, 0]$ or $[2, 0, 1]$. As a result, if Agent $\mathcal{A}_2$ chooses to reason at

level 3, in addition to obtaining the information that agent $\mathcal{A}_1$ reasons at level 2, $\mathcal{A}_2$ also needs to additionally know in which of the three ways will the level-2 reasoning of $\mathcal{A}_1$ be performed. Therefore, when $M > 2$ agents are present, as the reasoning level increases, the reasoning complexity, as well as computational cost, grows significantly. As a consequence, compared with the agents in 2-agent games, the agents in games with $M > 2$ agents are expected to display a stronger preference to reasoning at low levels.

E.3 Proof of Theorems 8.2 and 8.3

Before proving the main theorems, we need the following lemma showing a high-probability uniform upper bound on the value of the payoff function.

Lemma E.1. *Let $\delta \in (0, 1)$ and $\beta_t = 2\log(|\mathcal{X}_1|t^2\pi^2/3\delta)$, then with probability $\geq 1 - \delta$,*

$$|f_1(\mathbf{x}_1, \mathbf{x}_2) - \mu_{t-1}(\mathbf{x}_1, \mathbf{x}_2)| \leq \beta_t^{1/2}\sigma_{t-1}(\mathbf{x}_1, \mathbf{x}_2)$$

for all $\mathbf{x}_1 \in \mathcal{X}_1$, $\mathbf{x}_2 \in \mathcal{X}_2$, and $t \geq 1$.

The proof of Lemma E.1 makes use of the Gaussian concentration inequality and the union bound, and the proof can be found in Lemma 5.1 of (Srinivas et al., 2010). Note that a tighter confidence bound (i.e., a smaller value of $\beta_t = 2\log(|\mathcal{X}_1|t^2\pi^2/6\delta)$) is possible, however, the value of β_t in Lemma E.1 is selected for convenience to match the requirement of GP-MW (Theorem 8.1).

E.3.1 Theorem 8.2

Denote the history of game plays for $\mathcal{D}$ (the defender) up to iteration $t-1$ as $\mathcal{H}_{t-1}$, which includes $\mathcal{D}$'s selected actions (inputs) and observed payoffs (outputs) in every iteration from 1 to $t-1$: $\mathcal{H}_{t-1} = [\mathbf{x}_{2,1}, y_{2,1}, \mathbf{x}_{2,2}, y_{2,2}, \ldots, \mathbf{x}_{2,t-1}, y_{2,t-1}]$.

Again, we use superscripts to denote the reasoning level such that if $\mathcal{D}$ reasons at level 0, $\mathcal{H}_{t-1} = [\mathbf{x}^0_{2,1}, y^0_{2,1}, \mathbf{x}^0_{2,2}, y^0_{2,2}, \ldots, \mathbf{x}^0_{2,t-1}, y^0_{2,t-1}]$.

Here, we analyze the regret of the level-1 strategy, i.e., when $\mathcal{A}$ (the attacker) reasons at level $k = 1$ and $\mathcal{D}$ (the defender) reasons at level $k' = 0$. Note that in iteration t, the level-0 strategy of $\mathcal{D}$ (i.e., the distribution of $\mathbf{x}_{2,t}$) may depend on the history of input-output pairs of $\mathcal{D}$, i.e., $\mathcal{H}_{t-1}$, which is true for both the GP-MW and EXP3 strategies. Therefore, when analyzing $\mathcal{A}$'s expected regret in iteration t (with the expectation taken over the level-0 strategy of $\mathcal{D}$ in iteration t), we need to condition on $\mathcal{H}_{t-1}$. We denote the regret of $\mathcal{A}$ in iteration t as $r_{1,t}$, i.e., $R_{1,T} = \sum_{t=1}^{T} r_{1,t}$ in which $R_{1,T}$ represents external regret defined in (8.1). As a result, with probability of at least $1 - \delta$, the expected regret of $\mathcal{A}$ (the attacker) in iteration t, given $\mathcal{H}_{t-1}$, can be analyzed as

$$
\begin{aligned}
\mathbb{E}_{\mathbf{x}^0_{2,t}}[r_{1,t}|\mathcal{H}_{t-1}] &= \mathbb{E}_{\mathbf{x}^0_{2,t}}\left[f_1\left(\mathbf{x}^*_1, \mathbf{x}^0_{2,t}\right) - f_1\left(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}\right)|\mathcal{H}_{t-1}\right] \\
&\overset{(a)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\alpha_{1,t}\left(\mathbf{x}^*_1, \mathbf{x}^0_{2,t}\right) - f_1\left(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}\right)|\mathcal{H}_{t-1}\right] \\
&\overset{(b)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\alpha_{1,t}\left(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}\right) - f_1\left(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}\right)|\mathcal{H}_{t-1}\right] \\
&\overset{(c)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\mu_{t-1}(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}) + \beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}) - f_1\left(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t}\right)|\mathcal{H}_{t-1}\right] \\
&\overset{(d)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}, \mathbf{x}^0_{2,t})|\mathcal{H}_{t-1}\right]
\end{aligned}
\tag{E.4}
$$

in which (a) results from Lemma E.1 and the definition of the GP-UCB acquisition function (α) in Section 8.2, (b) follows from the definition of the level-1 strategy (8.3) as well as the linearity of the expectation operator, (c) results from the definition of the GP-UCB acquisition function, and (d) is again a consequence of Lemma E.1.

Next, the expected external regret of $\mathcal{A}$ reasoning at level 1 can be upper-

bounded:

$$
\begin{aligned}
\mathbb{E}[R_{1,T}] &= \mathbb{E}_{\mathbf{x}^0_{2,1}, y^0_{2,1}, \ldots, \mathbf{x}^0_{2,T-1}, y^0_{2,T-1}, \mathbf{x}^0_{2,T}}[R_{1,T}] \\
&= \mathbb{E}_{\mathbf{x}^0_{2,1}, y^0_{2,1}, \ldots, \mathbf{x}^0_{2,T-1}, y^0_{2,T-1}, \mathbf{x}^0_{2,T}} \left[\sum_{t=1}^{T} r_{1,t} \right] \\
&\overset{(a)}{=} \mathbb{E}_{\mathbf{x}^0_{2,1}} \left[r_{1,1} \right] + \mathbb{E}_{\mathbf{x}^0_{2,1}, y^0_{2,1}, \mathbf{x}^0_{2,2}} \left[r_{1,2} \right] + \ldots + \\
&\quad \mathbb{E}_{\mathbf{x}^0_{2,1}, y^0_{2,1}, \ldots, \mathbf{x}^0_{2,T-1}, y^0_{2,T-1}, \mathbf{x}^0_{2,T}} \left[r_{1,T} \right] \\
&\overset{(b)}{=} \mathbb{E}_{\mathbf{x}^0_{2,1}} \left[r_{1,1} \right] + \mathbb{E}_{\mathbf{x}^0_{2,1}, y^0_{2,1}} \left[\mathbb{E}_{\mathbf{x}^0_{2,2}} \left[r_{1,2} | \mathbf{x}^0_{2,1}, y^0_{2,1} \right] \right] + \ldots + \\
&\quad \mathbb{E}_{\mathbf{x}^0_{2,1}, y^0_{2,1}, \ldots, \mathbf{x}^0_{2,T-1}, y^0_{2,T-1}} \left[\mathbb{E}_{\mathbf{x}^0_{2,T}} \left[r_{1,T} | \mathbf{x}^0_{2,1}, y^0_{2,1}, \ldots, \mathbf{x}^0_{2,T-1}, y^0_{2,T-1} \right] \right] \\
&= \mathbb{E}_{\mathbf{x}^0_{2,1}} \left[r_{1,1} \right] + \mathbb{E}_{\mathcal{H}_1} \left[\mathbb{E}_{\mathbf{x}^0_{2,2}} \left[r_{1,2} | \mathcal{H}_1 \right] \right] + \ldots + \\
&\quad \mathbb{E}_{\mathcal{H}_{T-1}} \left[\mathbb{E}_{\mathbf{x}^0_{2,T}} \left[r_{1,T} | \mathcal{H}_{T-1} \right] \right] \\
&\overset{(c)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,1}} \left[2\beta_1^{1/2} \sigma_0(\mathbf{x}_{1,1}, \mathbf{x}_{2,1}) \right] + \\
&\quad \mathbb{E}_{\mathcal{H}_1} \left[\mathbb{E}_{\mathbf{x}^0_{2,2}} \left[2\beta_2^{1/2} \sigma_1(\mathbf{x}_{1,2}, \mathbf{x}_{2,2}) | \mathcal{H}_1 \right] \right] + \ldots + \\
&\quad \mathbb{E}_{\mathcal{H}_{T-1}} \left[\mathbb{E}_{\mathbf{x}^0_{2,T}} \left[2\beta_T^{1/2} \sigma_{T-1}(\mathbf{x}_{1,T}, \mathbf{x}_{2,T}) | \mathcal{H}_{T-1} \right] \right] \\
&\overset{(d)}{=} \mathbb{E}_{\mathbf{x}^0_{2,1}} \left[2\beta_1^{1/2} \sigma_0(\mathbf{x}_{1,1}, \mathbf{x}_{2,1}) \right] + \mathbb{E}_{\mathcal{H}_1, \mathbf{x}^0_{2,2}} \left[2\beta_2^{1/2} \sigma_1(\mathbf{x}_{1,2}, \mathbf{x}_{2,2}) \right] + \ldots + \\
&\quad \mathbb{E}_{\mathcal{H}_{T-1}, \mathbf{x}^0_{2,T}} \left[2\beta_T^{1/2} \sigma_{T-1}(\mathbf{x}_{1,T}, \mathbf{x}_{2,T}) \right] \\
&\overset{(e)}{=} \mathbb{E}_{\mathcal{H}_{T-1}, \mathbf{x}^0_{2,T}} \left[\sum_{t=1}^{T} 2\beta_t^{1/2} \sigma_{t-1}(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}) \right] \\
&\overset{(f)}{\leq} \mathbb{E}_{\mathcal{H}_{T-1}, \mathbf{x}^0_{2,T}} \left[\sqrt{C_1 T \beta_T \gamma_T} \right] \\
&\overset{(g)}{=} \sqrt{C_1 T \beta_T \gamma_T}
\end{aligned}
\tag{E.5}
$$

in which $C_1 = 8/\log(1 + \sigma_1^{-2})$, β_T is defined in Lemma E.1, and γ_T is the maximum information gain about the function f_1 obtained from any set of observations of size T. Steps (a) and (e) both result from the fact that $r_{1,t}$ only depends on the level-0 strategy of iteration t and the history up to iteration $t-1$

(through the level-0 strategy of iteration t), and is thus independent of those input actions and output observations in future iterations $t+1, \ldots, T$. (b) and (d) both follow from the law of total expectation, (c) results from (E.4), (f) follows from Lemmas 5.3 and 5.4 of (Srinivas et al., 2010), (g) follows since all terms inside the expectation are independent of the history of input-output pairs. Note that the expectation in (E.5) is taken over the history of selected actions and observed payoffs of $\mathcal{D}$. Note that an upper bound on the regret can be easily derived using the upper bound on the expected regret (E.5) through Markov's inequality, which suggests that level-1 reasoning achieves no regret asymptotically.

Of note, in the scenario in which more than two ($M > 2$) agents are present (Appendix E.2), with the modified level-1 policy given by (E.1), the proofs of (E.4) and (E.5) still go through by simply replacing $\mathbf{x}_{2,t}^0$ with the concatenated vector of $[\mathbf{x}_{2,t}^0, \ldots, \mathbf{x}_{M,t}^0]$ (i.e., the concatenation of the level-0 actions of all other agents) in every step of the proof. Similarly, the expectation of the regret would be taken over the history of input-output pairs of all other agents $2, \ldots, M$.

E.3.2 Theorem 8.3

For level-$k \geq 2$ reasoning, i.e., when $\mathcal{A}$ reasons at level k (for $k \geq 2$) and $\mathcal{D}$ reasons at level $k' = k-1 \geq 1$, the regret of $\mathcal{A}$ in iteration t can be analyzed as:

$$\begin{aligned} r_{1,t} &= f_1\left(\mathbf{x}_1^*, \mathbf{x}_{2,t}\right) - f_1\left(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}\right) \\ &= f_1\left(\mathbf{x}_1^*, \mathbf{x}_{2,t}^{k-1}\right) - f_1\left(\mathbf{x}_{1,t}^k, \mathbf{x}_{2,t}^{k-1}\right) \\ &\overset{(a)}{\leq} \alpha_{1,t}\left(\mathbf{x}_1^*, \mathbf{x}_{2,t}^{k-1}\right) - f_1\left(\mathbf{x}_{1,t}^k, \mathbf{x}_{2,t}^{k-1}\right) \\ &\overset{(b)}{\leq} \alpha_{1,t}\left(\mathbf{x}_{1,t}^k, \mathbf{x}_{2,t}^{k-1}\right) - f_1\left(\mathbf{x}_{1,t}^k, \mathbf{x}_{2,t}^{k-1}\right) \\ &\leq 2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}) \end{aligned} \tag{E.6}$$

in which (a) follows from Lemma E.1, (b) results from the fact that $\mathbf{x}_{1,t}^k$ is selected by maximizing the GP-UCB acquisition function α with respect to $\mathbf{x}_{2,t}^{k-1}$ according

to (8.6). (E.6) also holds with probability of at least $1-\delta$.

Next, the external regret can be upper bounded in a similar way as (E.5):

$$R_{1,T} = \sum_{t=1}^{T} r_{1,t} \overset{\text{(a)}}{\leq} \sum_{t=1}^{T} 2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}_{1,t}, \mathbf{x}_{2,t}) \overset{\text{(b)}}{\leq} \sqrt{C_1 T \beta_T \gamma_T} \tag{E.7}$$

in which (a) results from (E.6), and (b) again follows from Lemmas 5.3 and 5.4 of (Srinivas et al., 2010).

E.4 Proof of Theorem 8.4

Note that the level-1 action selected by $\mathcal{A}$ (the attacker) following R2-B2-Lite (8.8) is stochastic, instead of being deterministic as in R2-B2 (8.3). In the following, we denote the level-1 action of $\mathcal{A}$ following R2-B2-Lite as $\mathbf{x}_{1,t}^1(\widetilde{\mathbf{x}}_{2,t}^0)$ since, conditioned on all the game history up to iteration $t-1$, the selected level-1 action is a deterministic function of $\mathcal{A}$'s simulated action of $\mathcal{D}$ (the defender) at level 0 ($\widetilde{\mathbf{x}}_{2,t}^0$). Note that, in contrast to the corresponding definition in Appendix E.3.1, the history of game plays $\mathcal{H}'_{t-1}$ we define here additionally includes $\mathcal{A}$'s simulated action of $\mathcal{D}$ in every iteration: $\mathcal{H}'_{t-1} = [\mathbf{x}_{2,1}^0, \widetilde{\mathbf{x}}_{2,1}^0, y_{2,1}^0, \mathbf{x}_{2,2}^0, \widetilde{\mathbf{x}}_{2,2}^0, y_{2,2}^0, \ldots, \mathbf{x}_{2,t-1}^0, \widetilde{\mathbf{x}}_{2,t-1}^0, y_{2,t-1}^0]$. We use $\Sigma_{2,t}$ to denote the covariance matrix of the level-0 mixed strategy of $\mathcal{D}$ in iteration t ($\mathcal{P}_{2,t}$), and use $\mathrm{Tr}(\Sigma_{2,t})$ to represent its trace. As a result, the expected regret of $\mathcal{A}$ in iteration t can be analyzed as:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_{2,t}^0, \widetilde{\mathbf{x}}_{2,t}^0}[r_{1,t}|\mathcal{H}'_{t-1}] = \mathbb{E}_{\mathbf{x}_{2,t}^0, \widetilde{\mathbf{x}}_{2,t}^0}\left[f_1\left(\mathbf{x}_1^*, \mathbf{x}_{2,t}^0\right) - f_1\left(\mathbf{x}_{1,t}^1(\widetilde{\mathbf{x}}_{2,t}^0), \mathbf{x}_{2,t}^0\right)|\mathcal{H}'_{t-1}\right] \\
&\overset{\text{(a)}}{\leq} \mathbb{E}_{\mathbf{x}_{2,t}^0, \widetilde{\mathbf{x}}_{2,t}^0}\left[\alpha_{1,t}\left(\mathbf{x}_1^*, \mathbf{x}_{2,t}^0\right) - f_1\left(\mathbf{x}_{1,t}^1(\widetilde{\mathbf{x}}_{2,t}^0), \mathbf{x}_{2,t}^0\right)|\mathcal{H}'_{t-1}\right] \\
&\overset{\text{(b)}}{=} \mathbb{E}_{\mathbf{x}_{2,t}^0, \widetilde{\mathbf{x}}_{2,t}^0}\left[\alpha_{1,t}\left(\mathbf{x}_1^*, \widetilde{\mathbf{x}}_{2,t}^0\right) - f_1\left(\mathbf{x}_{1,t}^1(\widetilde{\mathbf{x}}_{2,t}^0), \mathbf{x}_{2,t}^0\right)|\mathcal{H}'_{t-1}\right] \\
&\overset{\text{(c)}}{\leq} \mathbb{E}_{\mathbf{x}_{2,t}^0, \widetilde{\mathbf{x}}_{2,t}^0}\left[\alpha_{1,t}\left(\mathbf{x}_{1,t}^1(\widetilde{\mathbf{x}}_{2,t}^0), \widetilde{\mathbf{x}}_{2,t}^0\right) - f_1\left(\mathbf{x}_{1,t}^1(\widetilde{\mathbf{x}}_{2,t}^0), \mathbf{x}_{2,t}^0\right)|\mathcal{H}'_{t-1}\right]
\end{aligned}$$

$$
\begin{aligned}
&\overset{(d)}{=} \mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\Bigg[\underline{\alpha_{1,t}\left(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^{(0,1)}_{2,t}),\widetilde{\mathbf{x}}^0_{2,t}\right)-\alpha_{1,t}\left(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t}\right)} \\
&\quad+\underline{\alpha_{1,t}\left(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t}\right)-f_1\left(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t}\right)}|\mathcal{H}'_{t-1}\Bigg] \\
&\overset{(e)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[L_{\alpha_1}\left\|\widetilde{\mathbf{x}}^0_{2,t}-\mathbf{x}^0_{2,t}\right\|_2|\mathcal{H}'_{t-1}\right] \\
&\quad+\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[\alpha_{1,t}\left(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t}\right)-f_1\left(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t}\right)|\mathcal{H}'_{t-1}\right] \\
&\overset{(f)}{\leq} \mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[L_{\alpha_1}\sqrt{\left\|\widetilde{\mathbf{x}}^0_{2,t}-\mathbf{x}^0_{2,t}\right\|^2_2}|\mathcal{H}'_{t-1}\right]+ \qquad \text{(E.8)} \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&\overset{(g)}{\leq} L_{\alpha_1}\sqrt{\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[\left\|\widetilde{\mathbf{x}}^0_{2,t}-\mathbf{x}^0_{2,t}\right\|^2_2|\mathcal{H}'_{t-1}\right]}+ \qquad \text{(E.9)} \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&= L_{\alpha_1}\sqrt{\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[\left(\widetilde{\mathbf{x}}^0_{2,t}-\mathbf{x}^0_{2,t}\right)^\top\left(\widetilde{\mathbf{x}}^0_{2,t}-\mathbf{x}^0_{2,t}\right)|\mathcal{H}'_{t-1}\right]}+ \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&= L_{\alpha_1}\sqrt{\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[\left(\widetilde{\mathbf{x}}^0_{2,t}\right)^\top\left(\widetilde{\mathbf{x}}^0_{2,t}\right)+\left(\mathbf{x}^0_{2,t}\right)^\top\left(\mathbf{x}^0_{2,t}\right)-2\left(\widetilde{\mathbf{x}}^0_{2,t}\right)^\top\left(\mathbf{x}^0_{2,t}\right)|\mathcal{H}'_{t-1}\right]}+ \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&\overset{(h)}{=} L_{\alpha_1}\sqrt{\mathbb{E}_{\widetilde{\mathbf{x}}^0_{2,t}}\left[\left(\widetilde{\mathbf{x}}^0_{2,t}\right)^\top\left(\widetilde{\mathbf{x}}^0_{2,t}\right)\right]+\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\left(\mathbf{x}^0_{2,t}\right)^\top\left(\mathbf{x}^0_{2,t}\right)\right]-2\mathbb{E}_{\widetilde{\mathbf{x}}^0_{2,t}}\left[\widetilde{\mathbf{x}}^0_{2,t}\right]^\top\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\mathbf{x}^0_{2,t}\right]}+ \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&\overset{(i)}{=} L_{\alpha_1}\sqrt{\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\left(\mathbf{x}^0_{2,t}\right)^\top\left(\mathbf{x}^0_{2,t}\right)\right]+\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\left(\mathbf{x}^0_{2,t}\right)^\top\left(\mathbf{x}^0_{2,t}\right)\right]-2\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\mathbf{x}^0_{2,t}\right]^\top\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\mathbf{x}^0_{2,t}\right]}+ \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&= \sqrt{2}L_{\alpha_1}\sqrt{\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\left(\mathbf{x}^0_{2,t}\right)^\top\left(\mathbf{x}^0_{2,t}\right)\right]-\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\mathbf{x}^0_{2,t}\right]^\top\mathbb{E}_{\mathbf{x}^0_{2,t}}\left[\mathbf{x}^0_{2,t}\right]}+ \\
&\quad\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&\overset{(j)}{=} \sqrt{2}L_{\alpha_1}\sqrt{\mathrm{Tr}\left(\Sigma_{2,t}\right)}+\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \\
&\overset{(k)}{\leq} \sqrt{2}L_{\alpha_1}\sqrt{\omega_t}+\mathbb{E}_{\mathbf{x}^0_{2,t},\widetilde{\mathbf{x}}^0_{2,t}}\left[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^1_{1,t}(\widetilde{\mathbf{x}}^0_{2,t}),\mathbf{x}^0_{2,t})|\mathcal{H}'_{t-1}\right] \qquad \text{(E.10)}
\end{aligned}
$$

in which (a) results from Lemma E.1; (b) holds because, conditioned on $\mathcal{H}_{t-1}$, $\mathbf{x}^0_{2,t}$ and $\widetilde{\mathbf{x}}^{(0,1)}_{2,t}$ are sampled from the same distribution and thus identically distributed; (c) follows from the way in which $\mathbf{x}^1_{1,t}$ is selected using the R2-B2-Lite algorithm (8.8), i.e., by deterministically best-responding to $\widetilde{\mathbf{x}}^0_{2,t}$ in terms of the GP-UCB acquisition function; (d) simply subtracts and adds the same GP-UCB term; (e) follows from the Lipschitz continuity of the GP-UCB acquisition function, whose Lipschitz constant (denoted as L_{α_1}) has been shown to be finite in (Kim and Choi, 2019); (f) is a result of the definition of the GP-UCB acquisition function (Section 8.2) and Lemma E.1; (g) results from the concavity of the square root function; (h) follows from the linearity of expectation and the fact that $\widetilde{\mathbf{x}}^0_{2,t}$ and $\mathbf{x}^0_{2,t}$ are independent; (i) again results from the fact that $\widetilde{\mathbf{x}}^0_{2,t}$ and $\mathbf{x}^0_{2,t}$ are identically distributed; (j) follows from the definition of $\Sigma_{2,t}$, i.e., the covariance matrix of the level-0 mixed strategy of the defender in iteration t; (k) follows from our assumption in Theorem 8.4 that the trace of $\Sigma_{2,t}$ is upper-bounded by the sequence $\{\omega_t\}$ for all $t \geq 1$. Note that all expectations in (E.10) are conditioned on $\mathcal{D}'_{t-1}$, and some of the conditioning are omitted to shorten the expression.

Next, the expected external regret can be upper-bounded in a similar way as (E.5):

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}^0_{2,1},\widetilde{\mathbf{x}}^0_{2,1},y^0_{2,1},\ldots,\mathbf{x}^0_{2,T-1},\widetilde{\mathbf{x}}^0_{2,T-1},y^0_{2,T-1},\mathbf{x}^0_{2,T},\widetilde{\mathbf{x}}^0_{2,T}}[R_{1,T}] \\
&\quad= \mathbb{E}_{\mathbf{x}^0_{2,1},\widetilde{\mathbf{x}}^0_{2,1},y^0_{2,1},\ldots,\mathbf{x}^0_{2,T-1},\widetilde{\mathbf{x}}^0_{2,T-1},y^0_{2,T-1},\mathbf{x}^0_{2,T},\widetilde{\mathbf{x}}^0_{2,T}}\left[\sum_{t=1}^{T} r_{1,t}\right] \\
&\quad\leq \sqrt{2}L_{\alpha_1}\sum_{t=1}^{T}\sqrt{\omega_t} + \sqrt{C_1 T \beta_T \gamma_T}
\end{aligned} \tag{E.11}$$

Note that compared with Theorem 8.2, the expectation in Theorem 8.4 is additionally taken over $\mathcal{A}$'s simulated action of $\mathcal{D}$ in all iterations, i.e.,

$\widetilde{\mathbf{x}}^0_{2,1}, \ldots, \widetilde{\mathbf{x}}^0_{2,T}$. Finally, Theorem 8.4 follows:

$$\mathbb{E}[R_{1,T}] \leq \mathcal{O}\left(\sum_{t=1}^{T} \sqrt{\omega_t} + \sqrt{T\beta_T\gamma_T}\right) \tag{E.12}$$

Similar to the analysis of R2-B2, in the scenario where more than two ($M > 2$) agents are involved, with the modified level-1 R2-B2-Lite algorithm given by (E.2), the proofs given above still go through by simply replacing $\mathbf{x}^0_{2,t}$ with the concatenated vector of $[\mathbf{x}^0_{2,t}, \ldots, \mathbf{x}^0_{M,t}]$ (and replacing $\widetilde{\mathbf{x}}^0_{2,t}$ with the concatenated vector of $[\widetilde{\mathbf{x}}^0_{2,t}, \ldots, \widetilde{\mathbf{x}}^0_{M,t}]$) in every step of the proof. Again, the expectation of the regret of agent $\mathcal{A}_1$ is taken over the history of input-output pairs of all other agents, as well as $\mathcal{A}_1$'s simulated level-0 actions of all other agents in every iteration.

E.5 Proof of Theorems 8.2 and 8.3 for $M > 2$ Agents

We prove here that the regret upper bound in Theorems 8.2 and 8.3 also hold in games with $M > 2$ agents. We only give the proof for level-$k \geq 2$ strategy since the proofs for level-0 and level-1 strategies are straightforward as explained in Appendices E.2 and E.3. For simplicity, we only focus on the scenario in which agent $\mathcal{A}_1$ reasons at level 2, whereas all other agents reason at either level 0 or level 1. However, the proof can be generalized to the settings in which agent $\mathcal{A}_1$ reasons at a higher level $k > 2$. Following the notations of Appendix E.2, the

expected regret of $\mathcal{A}_1$ in iteration t can be upper bounded as:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t}}[r_{1,t}|\mathcal{H}_{t-1}] = \mathbb{E}_{\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t}}\Big[f_1\big(\mathbf{x}^*_1,\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}\big) - \\
&\quad f_1\left(\mathbf{x}_{1,t},\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}\right)|\mathcal{H}_{t-1}\Big] \\
&\quad \leq \mathbb{E}_{\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t}}\Big[\alpha_{1,t}\left(\mathbf{x}^*_1,\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}\right) - \\
&\qquad f_1\left(\mathbf{x}^1_{2,t},\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}\right)|\mathcal{H}_{t-1}\Big] \\
&\quad \leq \mathbb{E}_{\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t}}\Big[\alpha_{1,t}\left(\mathbf{x}^2_{1,t},\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}\right) - \\
&\qquad f_1\left(\mathbf{x}^1_{2,t},\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}\right)|\mathcal{H}_{t-1}\Big] \\
&\quad \leq \mathbb{E}_{\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t}}\Big[2\beta_t^{1/2}\sigma_{t-1}(\mathbf{x}^2_{1,t},\mathbf{x}^0_{2,t},\ldots,\mathbf{x}^0_{M_0,t},\mathbf{x}^1_{M_0+1,t},\ldots,\mathbf{x}^1_{M,t}|\mathcal{H}_{t-1}\Big]
\end{aligned} \tag{E.13}$$

The proof given in (E.13) is analogous to (E.4). The key difference from (E.4) is that in this case, the expectation here is taken over the level-0 strategies of those agents reasoning at level 0, i.e., $\mathcal{A}_2,\ldots,\mathcal{A}_{M_0}$. In contrast, in (E.4), the expectation is only taken over the level-0 strategy of the single opponent reasoning at level 0.

Note that if none of the other agents reason at level 0, the expectation operator in (E.13) can be dropped. As a result, (E.7) can be directly used to show that the resulting upper bound on the regret is the same as that given in Theorem 8.3. On the other hand, if there exists at least 1 level-0 agents, the expectation operator remains. Therefore, the subsequent proof follows from (E.5) and the resulting regret upper bound becomes the same as that shown in Theorem 8.2, except that the expectation of the regret is taken over the history of input-output pairs of all level-0 agents.

E.6 More Experimental Details and Results

All experiments are run on computers with 16 cores of Intel Xeon processor, 5 NVIDIA GTX1080 Ti GPUs, and a RAM of 256G.

E.6.1 Synthetic Games

E.6.1.1 2-Agent Synthetic Games

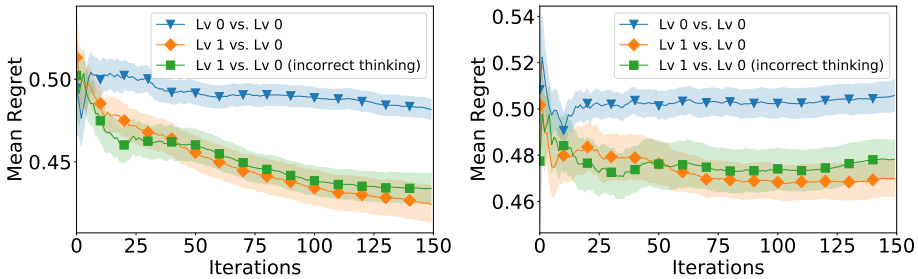
(a) Detailed Experimental Setting

The payoff functions used in the synthetic games are sampled from GPs with the Squared Exponential kernel with length scale 0.1. All payoff functions are defined on a 2-dimensional grid of equally spaced points in $[0, 1]^2$ with size $|\mathcal{X}_1| \times |\mathcal{X}_2| = 100 \times 100$. Therefore, the action spaces of agent 1 and agent 2 both consist of $|\mathcal{X}_1| = |\mathcal{X}_2| = 100$ points. For common-payoff games, we randomly sample a function f_1 from a GP on the domain $\mathcal{X}_1 \times \mathcal{X}_2$ and set $f_2(\mathbf{x}_1, \mathbf{x}_2) = f_1(\mathbf{x}_1, \mathbf{x}_2)$ for all $\mathbf{x}_1 \in \mathcal{X}_1$ and $\mathbf{x}_2 \in \mathcal{X}_2$; regarding general-sum games, we randomly and independently sample two functions, f_1 and f_2, from the same GP; as for constant-sum games, we draw a function f_1 from the GP, and set $f_2(\mathbf{x}_1, \mathbf{x}_2) = 1 - f_1(\mathbf{x}_1, \mathbf{x}_2)$ for all $\mathbf{x}_1 \in \mathcal{X}_1$ and $\mathbf{x}_2 \in \mathcal{X}_2$. All payoff functions are scaled into the range $[0, 1]$. Note that since the domain size is not excessively large, the level-1 action can be selected by solving (8.3) exactly instead of approximately. The true GP hyperparameters, with which the synthetic payoff functions are sampled, are used as the GP hyperparameters.

(b) More Results on the Impact of Incorrect Thinking about the Other Agent

We further investigate how the performance of an agent is affected by incorrect thinking about the other agent. Fig. E.1 plots the performance of agent 1 when agent 1 and agent 2 reason at levels 1 and 0 respectively, while agent 1's thinking about agent 2's level-0 strategy is incorrect. The figures demonstrate that in

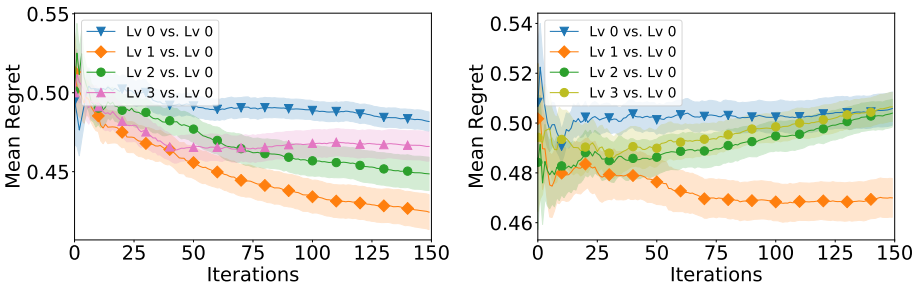
the presence of an incorrect thinking about the other agent's level-0 strategy, the performance of agent 1 only suffers from a marginal drop, although the theoretical guarantee offered by Theorem 8.2 no longer holds. Fig. E.2 illustrates the impacts of an incorrect thinking about the other agent's reasoning level. As shown in the figure, when agent 2's reasoning level is fixed at level 0, agent 1 obtains the best performance when reasoning at level 1, which agrees with our theoretical analysis since by reasoning at level 1, agent 1's performance is theoretically guaranteed (Theorem 8.2). Meanwhile, when agent 1 reasons at a higher level (e.g., level 2 or level 3), the performance becomes worse (compared with reasoning at level 1) yet is still better than reasoning at level 0 (the blue curve); this might be attributed to the fact that when agent 1 reasons at level 2 or 3, even though agent 1's GP-UCB value is highly likely to be maximized with respect to the wrong action in every iteration (8.6), this could still help agent 1 to eliminate some potentially "dominated actions", i.e., those actions which yield small GP-UCB values regardless of the action of agent 2. This ability to discard those dominated actions gives agent 1 a preference to avoid selecting actions with small GP-UCB values, and thus might help agent 1 obtain a better performance compared with reasoning at level 0.



(a) General-sum games. (b) Constant-sum games.

Figure E.1: Agent 1's performance of level-1 reasoning (agent 2 reasons at level 0) when agent 1's thinking about agent 2's level-0 strategy is incorrect. I.e., agent 2 uses GP-MW as the level-0 strategy, while agent 1 thinks that agent 2 uses the random search level-0 strategy.

(c) Results Using Other Level-0 Strategies

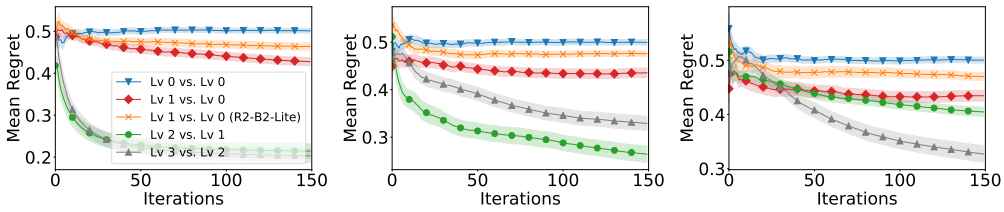


(a) General-sum games. (b) Constant-sum games.

Figure E.2: Agent 1's performance when its thinking about agent 2's reasoning level is incorrect. That is, agent 2 reasons at level 0, while agent 1 reasons at levels 1, 2 and 3, where the last two settings result from agent 1's incorrect thinking about agent 2's reasoning level.

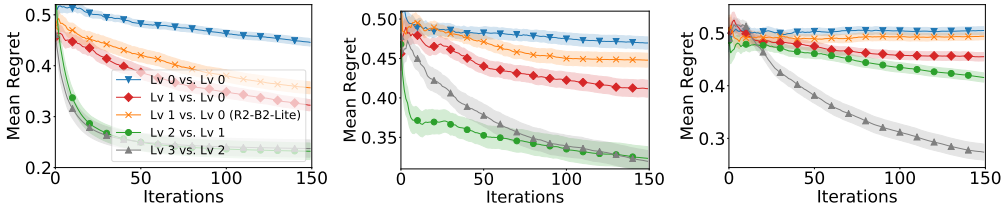
In addition to the results presented in the main text which use GP-MW as the level-0 strategy (Fig. 8.2a to c), the entire set of experiments are repeated for the random search and EXP3 level-0 strategies, whose corresponding results are presented in Figs. E.3 and E.4. These results yield the same observations and interpretations as Figs. 8.2a to c, and demonstrate the robustness of our R2-B2 algorithm with respect to the choice of the level-0 strategy. Another interesting observation regarding different level-0 strategies is that in common-payoff and general-sum games, when both agents reason at level 0, running a no-regret level-0 strategy (e.g., GP-MW or EXP3), instead of random search, leads to decreasing mean regret. Specifically, when both agents reason at level 0, the mean regret in common-payoff and general-sum games is decreasing if either GP-MW (Fig. 8.2a and b) or EXP3 (Fig. E.4a and b) is used as the level-0 strategy (with the decreasing trend more discernible in common-payoff games), while the random search level-0 strategy results in a non-decreasing mean regret (Fig. E.3a and b). This observation demonstrates the benefit of adopting a better/more strategic level-0 strategy (instead of a non-strategic level-0 strategy such as random search) when reasoning at level 0.

For the EXP3 level-0 strategy, we follow the practice of the work of (Rahimi and Recht, 2007). That is, we firstly draw $d'_1 = 5$ samples of $[\boldsymbol{\omega}_i]_{i=1,\ldots,d'_1}$ from



(a) Common-payoff games. (b) General-sum games. (c) Constant-sum games.

Figure E.3: Mean regret of agent 1 in different types of synthetic games, with agent 2 taking the random search level-0 strategy.



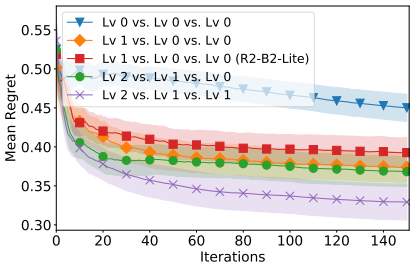
(a) Common-payoff games. (b) General-sum games. (c) Constant-sum games.

Figure E.4: Mean regret of agent 1 in different types of synthetic games, with agent 2 taking the EXP-3 level-0 strategy.

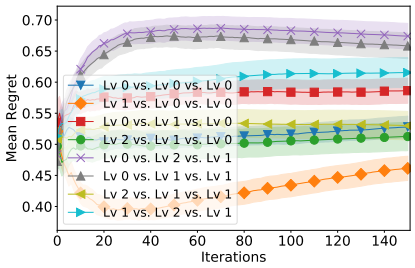
the spectral density of the GP kernel (i.e., the Squared Exponential kernel with length scale 0.1), and d_1' samples of $[b_i]_{i=1,\ldots,d_1'}$ from the uniform distribution over $[0, 2\pi]$; then, for every input $\mathbf{x}_1 \in \mathcal{X}_1$ in the domain, we use $[\sqrt{2/d_1'}\cos(\boldsymbol{\omega}_i^\top \mathbf{x}_1 + b_i)]_{i=1,\ldots,d_1'}$ as the d_1'-dimensional feature representing $\mathbf{x}_1$. Subsequently, the GP surrogate can be replaced with a linear surrogate model with the resulting features as inputs, and thus the EXP3 algorithm for adversarial linear bandit can be applied.

E.6.1.2 Synthetic Games with $M > 2$ Agents

We also use synthetic games with $M > 2$ agents to evaluate the effectiveness of our R2-B2 algorithm when more than two agents are involved. We consider two types of synthetic games involving three agents. In the first type of games, the payoff functions of the three agents are independently sampled from a GP. The second type of games includes one adversary and two (cooperating) agents, the payoff function for the adversary, $f_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, is a function sampled from a GP



(a) Mean regret of agent 1 in the three-agent game with independent payoff functions. The reasoning levels are in the form of agent 1 vs agent 2 vs agent 3.



(b) Mean regret of the adversary in the three-agent game with 1 adversary and 2 agents. The reasoning levels are in the form of adversary vs agent 1 vs agent 2.

Figure E.5: Mean regret in three-agent games.

(and scaled to the range $[0, 1]$), whereas the payoff functions for the two agents are identical and defined as $1 - f_1(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$. We use GP-MW as the level-0 strategy.

Fig. E.5a displays the mean regret of agent 1 in the first type of games, i.e., games with independent payoff functions. The figure shows that in games with more than two agents, agent 1 gains benefit by following the R2-B2 algorithm presented in Appendix E.2. Specifically, the orange and red curves demonstrate the advantage of level-1 reasoning using R2-B2 (E.1) and R2-B2-Lite (E.2) respectively, and the green and purple curves illustrate the benefit of level-$k > 2$ reasoning (E.3).

Fig. E.5b shows the mean regret of the adversary in the second type of games involving one adversary and two agents. Note that the mean regret of the two agents can be directly read from the figure since it is equal to 1− the mean regret of the adversary. A number of interesting insights can be drawn from Fig. E.5. Comparing the orange and blue curves (similarly the green and red curves, and the yellow and gray curves) shows that the adversary obtains smaller regret by reasoning at a higher level than both agents; similarly, comparison of the blue and red curves (as well as the blue vs the purple, gray, and cyan curves) demonstrates

that both agents enjoy a smaller regret when at least one of them reasons at a higher level than the adversary; comparing the gray and red curves reveals that when both agents reason at a higher level (in contrast to when one of them reasons at a higher level), the agents benefit more in terms of regret; comparison of the cyan and purple curves shows that given that the two agents reason at levels 2 and 1 respectively, the adversary reduces its deficit in regret by reasoning at level 1 instead of level 0.

E.6.2 Adversarial ML

E.6.2.1 R2-B2 for Adversarial ML

(a) Detailed Experimental Setting

We focus on the standard black-box setting, i.e., both $\mathcal{A}$ (the attacker) and $\mathcal{D}$ (the defender) can only access the target ML model by querying the model and observing the corresponding predictive probabilities for different classes (Tu et al., 2019). Query efficiency is of critical importance for a black-box attacker since each query of the target ML model can be costly and an excessive number of queries might lead to the risk of being detected. Similarly, when defending against an attacker who adopts a query-efficient algorithm, it is also reasonable for the defender to defend in a query-efficient manner. This justifies the use of BO-based methods for both adversarial attack and defense methods, since BO has been repeatedly demonstrated to be sample-efficient (Shahriari et al., 2016) and has been successfully applied to black-box adversarial attacks (Ru et al., 2020). The GP hyperparameters are optimized by maximizing the marginal likelihood after every 10 iterations.

Both the MNIST and CIFAR-10 datasets can be downloaded using the Keras package in Python[1]. All pixel values of all images are normalized into the range $[0, 1]$. For the MNIST dataset, we use a convolutional neural network

[1] https://keras.io/

(CNN) model[2] with 99.25% validation accuracy (trained on $60,000$ samples and validated using $10,000$ samples) as the target ML model, and for CIFAR-10, we use a ResNet model[3] with 92.32% validation accuracy (trained using $50,000$ samples and validated on $10,000$ samples, data augmentation is used). All test images used in the experiments for attack/defense are randomly selected among those correctly classified images from the validation set. To improve the query efficiency of black-box adversarial attacks, different dimensionality reduction techniques such as autoencoder have been adopted to reduce the dimensionality of image data (Tu et al., 2019). In this work, we let both $\mathcal{A}$ and $\mathcal{D}$ use Variational Autoencoders (VAEs) (Kingma and Welling, 2014) for dimensionality reduction in a realistic setting: In every iteration of the repeated game, $\mathcal{A}$ encodes the test image into a low-dimensional latent vector (i.e., the mean vector of the encoded latent distribution) using a VAE, perturbs the vector, and then decodes the perturbed vector to obtained the resulting image with perturbations; next, $\mathcal{D}$ receives the perturbed image, uses a VAE to encode the perturbed image to obtain a low-dimensional latent vector (i.e., the mean vector of the encoded latent distribution), adds transformations (perturbations) to the latent vector, and finally decodes the vector into the final image to be passed as input to the target ML model. In the experiments, the same VAE is used by both $\mathcal{A}$ and $\mathcal{D}$, but the use of different VAEs can be easily achieved. The latent dimension (LD) is $d_1 = d_2 = 2$ for MNIST and $d_1 = d_2 = 8$ for CIFAR-10; the action space for both $\mathcal{A}$ and $\mathcal{D}$ (i.e., the space of allowed perturbations to the latent vectors) is $[-2, 2]^2$ for MNIST, and $[-2, 2]^8$ for CIFAR-10. For MNIST, the VAE[4] is a multi-layer perceptron (MLP) with ReLU activation, in which the input image is flattened into a 28×28-dimensional vector and both the encoder and decoder

[2] https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py

[3] https://github.com/keras-team/keras/blob/master/examples/cifar10_resnet.py

[4] https://github.com/keras-team/keras/blob/master/examples/variational_autoencoder.py

consist of a 512-dimensional hidden layer. Regarding CIFAR-10, the encoder of the VAE uses 3 convolutional layers followed by a fully connected layer, whereas the decoder uses 2 fully connected layers followed by 3 de-convolutional layers[5].

For both $\mathcal{A}$ and $\mathcal{D}$, the image produced by the decoder of their VAE is clipped such that the requirement of bounded perturbations in terms of the infinity norm (as mentioned in Section 8.4.2.1 of the main text) is satisfied. We consider *untargeted attacks* in this work, i.e., the attacker's (defender's) goal is to cause (prevent) misclassification of the ML model. However, our framework can also deal with *targeted attacks* (i.e., the attacker aims at causing the target ML model to misclassify a test image into a particular class) through slight modifications to the payoff functions. The payoff function value for $\mathcal{A}$ ($f_1(\mathbf{x}_1, \mathbf{x}_2)$, referred to as the *attack score*) for a pair of perturbations selected by $\mathcal{A}$ ($\mathbf{x}_1$) and $\mathcal{D}$ ($\mathbf{x}_2$) is the maximum predictive probability (corresponding to the probability that test input belongs to a class) among all *incorrect classes*, which is bounded in $(0, 1)$. For example, in a 10-class classification model (i.e., for both MNIST and CIFAR-10), if the correct/ground-truth class for a test image is 0, the value of the payoff function for $\mathcal{A}$ is the maximum predictive probability among classes 1 to 9. The payoff function for $\mathcal{D}$ is $f_2(\mathbf{x}_1, \mathbf{x}_2) = 1 - f_1(\mathbf{x}_1, \mathbf{x}_2)$ since the defender attempts to make sure that the predictive probability of the correct class remains the largest by minimizing the maximum predictive probability among all incorrect classes.

As reported in the main text (Section 8.4.2.1), we use GP-MW and random search as the level-0 strategies for MNIST, and only use random search for CIFAR-10. The reason is that GP-MW requires a discrete input domain (or a discretized continuous input domain) since it needs to maintain and update a discrete distribution over the input domain. Therefore, it is difficult to apply GP-MW to a high-dimensional continuous input domain (e.g., the 8-dimensional domain in the CIFAR-10 experiment) since an accurate discretization of the high-

[5] https://github.com/chaitanya100100/VAE-for-Image-Generation

dimensional domain would lead to an intractably large domain for the discrete distribution, making it intractable to update and sample from the distribution. Similarly, the application of the EXP3 algorithm is also limited to low-dimensional input domains for the same reason.

(b) Results Using Multiple Images

Note that different images may be associated with different degrees of difficulty to attack and to defend, i.e., some images are easier to attack (and thus harder to defend) and others may be easier to defend (and thus harder to attack). Therefore, for those images that are easier to attack than to defend, it is easier for the attacker to increase the attack score than for the defender to reduce the attack score; as a result, the advantage achieved by the defender (i.e., lower attack score) when the defender reasons at one level higher would be less discernible since the defender's task (i.e., to decrease the attack score) is more difficult. On the other hand, for those images that are easier to defend than to attack (e.g., the MNIST dataset as demonstrated below), the benefit obtained by the attacker (i.e., higher attack score) when it reasons at one level higher would be harder to delineate since the attacker's task of increasing the attack score is more difficult. The image from MNIST/CIFAR-10 that is used to produce the results reported in the main text (Fig. 8.3a to c) is selected to ensure that the difficulties of attack and defense are comparable such that the effects of both attack and defense can be clearly illustrated.

Figs. E.6 and E.7 show the attack scores on the MNIST and CIFAR-10 datasets averaged over multiple randomly selected images (30 images for MNIST and 9 images for CIFAR-10). These figures yield consistent observations with those presented in the main text, except that for MNIST (Fig. E.6), the attack scores are generally lower (compared with the blue curve where both $\mathcal{A}$ and $\mathcal{D}$ reason at level 0), which could be explained by the fact that the images in the MNIST dataset are generally easier to defend than to attack (i.e., it is easier to make the

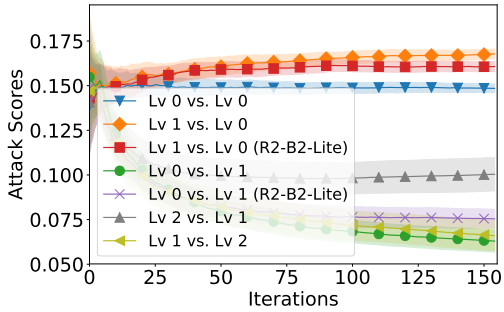


Figure E.6: Attack scores averaged over 30 images from MNIST. Each image is again averaged over 5 initializations of 5 randomly selected actions.

attack score lower than to make it higher, as explained in the previous paragraph) because of the simplicity of the dataset and the high accuracy of the target ML model (i.e., a validation accuracy of 99.25%). As a result, when $\mathcal{A}$ reasons at level 2 and $\mathcal{D}$ reasons at level 1, the attack score is lower than when both agents reason at level 0 (compare the gray and blue curves in Fig. E.6). In addition to the above-mentioned factor that the MNIST dataset is in general harder to attack (i.e., harder to make the attack score higher than to make it lower), this deviation from our theoretical result (Theorem 8.3) might also be attributed to the error in approximating the expectation operator in level-1 reasoning. However, the benefit of reasoning at one level higher can still be observed in this case, since when the reasoning level of $\mathcal{D}$ is fixed at 1, it is still beneficial for $\mathcal{A}$ to reason at level 2 (i.e., the gray curve) instead of level 0 (i.e., the green curves). The corresponding average number of successful attacks in 150 iterations for different reasoning levels yield the same observations and interpretations as Figs. E.6 and E.7: For MNIST (Fig. E.6), the number of successful attacks are (in the order of the figure legend from top to bottom) $20.4, 23.0, 21.3, 9.7, 11.0, 12.4, 7.9$, for CIFAR-10 (Fig. E.7), they are $32.9, 43.0, 38.8, 12.2, 21.0$.

(c) Impact of the Number of Samples Used for Approximating the Expectation in Level-1 Reasoning

For the results reported in the main text, the number of samples used to approximate the expectation in level-1 reasoning are 500 for MNIST (Fig. 8.3a and b) and

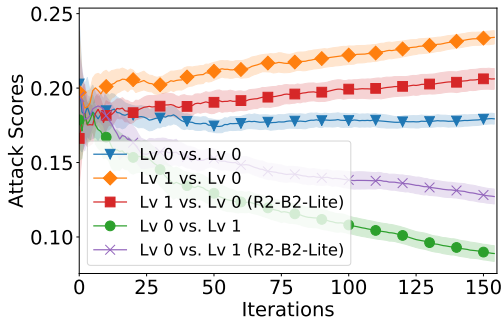


Figure E.7: Attack scores averaged over 9 images from CIFAR-10. Each image is again averaged over 5 initializations of 5 randomly selected actions.

1,000 for CIFAR-10 (Fig. 8.3c). Note that since the input dimension is higher for CIFAR-10, a larger number of samples is needed to accurately approximate the level-0 mixed strategy (over which the expectation in level-1 reasoning is taken). Here, we further investigate the impact of the number of samples used in the approximation of the expectation operator in level-1 reasoning (8.3). Fig. E.8 shows the attack scores for the MNIST dataset when $\mathcal{A}$ and $\mathcal{D}$ reason at levels 2 and 1 respectively when different number of samples are used for the approximation. Random search is used as the level-0 mixed strategy. The figure, as well as the corresponding number of successful attacks, demonstrates that the attack becomes more effective as more samples are used for the approximation. The benefit offered by using more samples for the approximation results from the fact that with a better accuracy at estimating $\mathcal{D}$'s level-1 action (8.5) (i.e., the level-1 action of $\mathcal{D}$ simulated by $\mathcal{A}$ is more likely to be the same as the actual level-1 action selected by $\mathcal{D}$), the attacker is able to best-respond to $\mathcal{D}$'s action more accurately (8.4), thus leading to an improved performance.

E.6.2.2 Defense against State-of-the-art Adversarial Attack Methods

(a) Against the Parsimonious Attacker[6]

Since the Parsimonious algorithm is deterministic (assuming that the random seed is fixed), it corresponds to a level-0 pure strategy, which is equivalent to

[6] https://github.com/snu-mllab/parsimonious-blackbox-attack

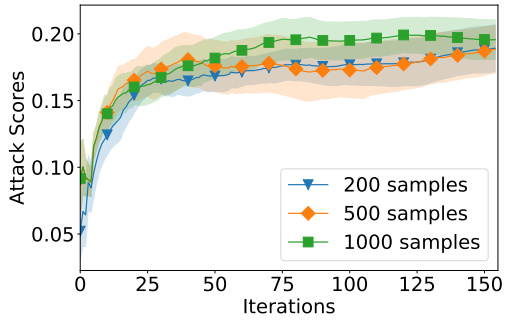


Figure E.8: Attack scores for MNIST when $\mathcal{A}$ (the attacker) and $\mathcal{D}$ (the defender) reason at levels 2 and 1 respectively, with different number of samples used for approximating the expectation for level 1 reasoning. The corresponding number of successful attacks (for 200, 500 and 1000 samples) are 2.6, 3.0 and 3.3.

a mixed strategy with all probability measure concentrated on a single action. Therefore, in our setting, when $\mathcal{D}$ (the defender) is selecting its level-1 strategy in iteration t using R2-B2, it knows exactly the action (perturbations) that $\mathcal{A}$ (the attacker) will select in the current iteration t. To make the setting more practical, we use the (encoded) image perturbed by $\mathcal{A}$ (instead of the encoded perturbations as in the experiments in Section 8.4.2.1) as the action of $\mathcal{A}$, $\mathbf{x}_1$. Specifically, every time $\mathcal{D}$ receives the perturbed image from $\mathcal{A}$, $\mathcal{D}$ encodes the image using its VAE, and use the encoded latent vector (i.e., the mean vector of the encoded latent distribution) as the input from $\mathcal{A}$ in the current iteration (i.e., $\mathbf{x}_{1,t}$). As a result, in every iteration, $\mathcal{D}$ naturally gains access to the action of $\mathcal{A}$ in the current iteration $\mathbf{x}_{1,t}$ and can thus reason at level 1 by best-responding to $\mathbf{x}_{1,t}$. Therefore, $\mathcal{D}$ has natural access to $\mathcal{A}$'s history of selected actions, which, combined with the fact that the game is constant-sum (which allows $\mathcal{D}$ to know $\mathcal{A}$'s payoff by observing $\mathcal{D}$'s own payoff), satisfies the requirement of perfect monitoring. Note that Parsimonious maximizes the loss (instead of the attack score as in the experiments in Section 8.4.2.1) of a test image as the objective of attack, so to be consistent with their algorithm, we use the negative loss as the payoff function of our level-1 R2-B2 defender. Refer to Fig. E.9 for the loss values achieved by Parsimonious with and without our level-1 R2-B2 defender for

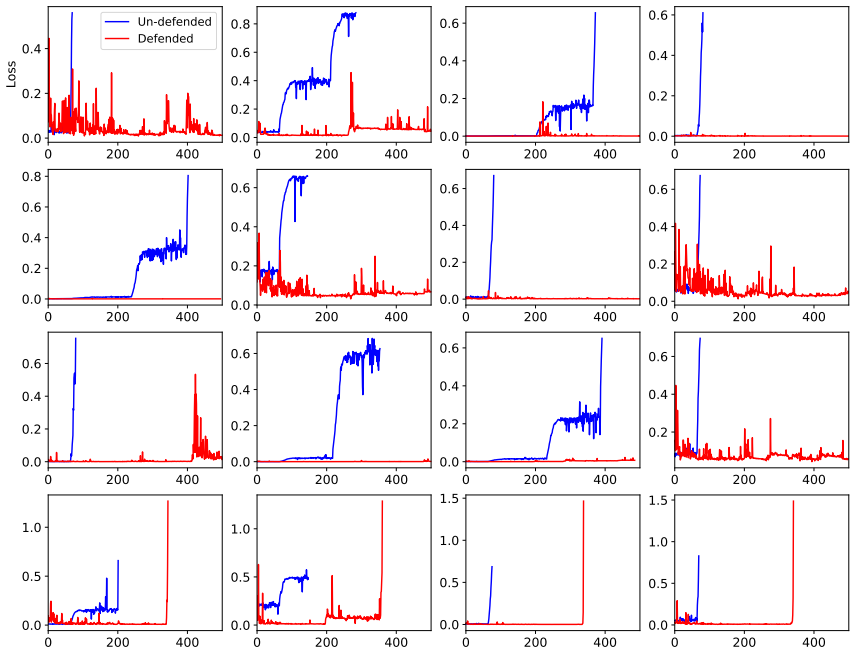


Figure E.9: The loss of the Parsimonious algorithm with and without our level-1 R2-B2 defender on some selected images. For the images on the first three rows, Parsimonious fails to achieve any successful attack; for the images on the last row, our level-1 R2-B2 defender requires Parsimonious to use a significantly larger number of queries to obtain a successful attack.

some selected images. The losses for different images are reported individually since they are highly disparate across different images, thus making their average losses hard to visualize.

(b) Against the BO Attacker

In addition to evaluating the effectiveness of our level-1 R2-B2 defender using the state-of-the-art Parsimonious algorithm (Section 8.4.2.2), we also investigate whether our level-1 R2-B2 defender is able to defend against black-box adversarial attacks using BO, which has recently become popular as a sample-efficient black-box method for adversarial attacks (Ru et al., 2020). Specifically, as a gradient-free technique to optimize black-box functions, BO can be naturally used to maximize the attack score (i.e., the output) over the space of adversarial perturbations (i.e., the input). Note that in contrast to the attacker in Section 8.4.2.1, the BO attacker here is not aware of the existence of the defender and thus the input to its

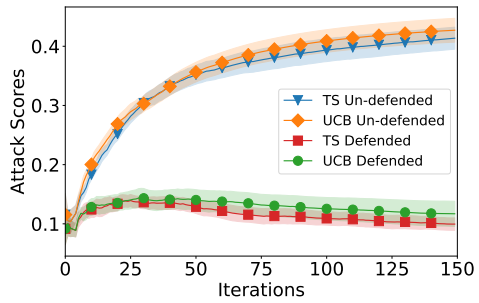


Figure E.10: Attack scores achieved by the black-box attacker using BO with the GP-UCB and Thompson sampling acquisition functions, with and without our level-1 R2-B2-Lite defender. The corresponding number of successful attacks are 70.1, 67.0, 0.8 and 0.7 respectively (in the order of the figure legend from top to bottom).

GP surrogate only consists of the (encoded) perturbations of the attacker. We adopt two commonly used acquisition functions for BO: (a) Thompson sampling (TS) which, as a randomized algorithm, corresponds to a level-0 mixed strategy, and (b) GP-UCB, which represents a level-0 pure strategy. For both types of adversarial attacks, we let our level-1 defender run the R2-B2-Lite algorithm. In particular, when the attacker uses the GP-UCB acquisition function, in each iteration, the defender calculates/simulates the action (perturbations) that would be selected by the attacker in the current iteration, and best-responds to it; when TS is adopted by the attacker as the acquisition function, the defender draws a sample using the attacker's randomized level-0 TS strategy in the current iteration, and best-responds to it. Fig. E.10 shows the results of adversarial attacks using the TS and GP-UCB acquisition functions with and without our level-1 R2-B2-Lite defender. As demonstrated in the figure, our level-1 R2-B2-Lite defender is able to effectively defend against and almost eliminate the impact of both types of adversarial attacks (i.e., allow the attacker to succeed for less than once over 150 iterations).

E.6.3 Multi-Agent Reinforcement Learning

The multi-agent particle environment adopted in our experiment can be found at https://github.com/openai/multiagent-particle-envs. The state and action of the two predators (referred to as predator 1 and predator 2 for simplicity), are represented by a 14-dimensional vector and a 5-dimensional vector respectively, whereas the state and action of the prey are represented by a 12-dimensional vector and a 5-dimensional vector correspondingly. For simplicity, we perform direct policy search using a linear policy space. That is, the policy of each predator is represented by a 14×5 matrix, which maps a 14-dimensional state vector to a 5-dimensional action vector, thus producing the action to be taken by the predator according to the current policy when the predator is in a particular state. Similarly, the policy of the prey corresponds to a 12×5 matrix, which is able to map a 12-dimensional state vector to a 5-dimensional action vector. To further simplify the setting and reduce the dimensionality of the policy space, we use rank-1 approximations of the policy matrices. That is, the 14×5 policy matrix of each predator is obtained by the outer product of a 14-dimensional vector and a 5-dimensional vector, whereas the 12×5 policy matrix of the prey is attained by the outer product of a 12-dimensional vector and a 5-dimensional vector. As a result, the policy of each predator is represented by $14 + 5 = 19$ parameters, whereas the policy of the prey is characterized by $12 + 5 = 17$ parameters. Therefore, the dimension of the input to the GP surrogate models is $19 + 19 + 17 = 55$. For every one of the 55 input dimensions, the search space is $[-1, 1]$. In each iteration of the repeated game, after all agents have selected their policy parameters, the agents use their respective policies to interact in the environment for 50 steps and use their obtained returns (i.e., cumulative rewards) as the corresponding payoff; every iteration of the repeated game involves 5 independent runs in the environment (with different initializations) using the selected policy parameters, and the averaged return over

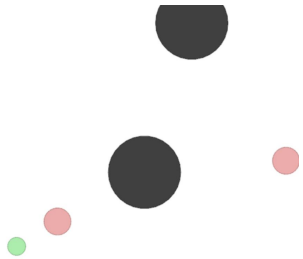


Figure E.11: Illustration of the predator-prey game. Red: predators; green: prey; black: obstacles.

the 5 independent runs is reported as the corresponding observed payoff. For ease of visualization, the returns are clipped and scaled into the range $[0, 1]$. All agents use random search as the level-0 strategy due to the high dimension of input action space; refer to Appendix E.6.2.1a for a detailed explanation about this choice. The GP hyperparameters are optimized via maximizing the marginal likelihood after every 10 iterations.