Differentially Private Federated Bayesian Optimization with Distributed Exploration

Zhongxiang Dai[†], **Bryan Kian Hsiang Low**[†], **Patrick Jaillet**[§]
Dept. of Computer Science, National University of Singapore, Republic of Singapore[†]
Dept. of Electrical Engineering and Computer Science, MIT, USA[§]
{daizhongxiang,lowkh}@comp.nus.edu.sg[†],jaillet@mit.edu[§]

Abstract

Bayesian optimization (BO) has recently been extended to the *federated learning* (FL) setting by the *federated Thompson sampling* (FTS) algorithm, which has promising applications such as federated hyperparameter tuning. However, FTS is not equipped with a rigorous privacy guarantee which is an important consideration in FL. Recent works have incorporated *differential privacy* (DP) into the training of deep neural networks through a general framework for adding DP to iterative algorithms. Following this general DP framework, our work here integrates DP into FTS to preserve *user-level privacy*. We also 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 high utility (competitive performance) with a strong privacy guarantee (small privacy loss) and induces a trade-off between privacy and utility.

1 Introduction

Bayesian optimization (BO) has become popular for optimizing expensive-to-evaluate black-box functions, such as tuning the hyperparameters of *deep neural networks* (DNNs) [56]. Motivated by the growing computational capability of edge devices and concerns over sharing the raw data, BO has recently been extended to the *federated learning* (FL) setting [46] to derive the setting of *federated BO* (FBO) [12]. The FBO setting allows multiple agents with potentially heterogeneous objective functions to collaborate in black-box optimization tasks without requiring them to share their raw data. For example, mobile phone users can use FBO to collaborate in optimizing the hyperparameters of their DNN models used in a smart keyboard application without sharing their sensitive raw data. Hospitals can use FBO to collaborate with each other when selecting the patients to perform a medical test [67] without sharing the sensitive patient information. 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 [29]. Furthermore, incorporating rigorous privacy preservation into BO has recently attracted increasing attention due to its importance to real-world BO applications [33, 36, 48, 71]. However, the state-of-the-art algorithm in the FBO setting, *federated Thompson sampling* (FTS) [12], is not equipped with privacy guarantee and thus lacks rigorous protection of the sensitive agent information.

Differential privacy (DP) [20] provides a rigorous privacy guarantee for data release and has become the state-of-the-art method for designing privacy-preserving ML algorithms [28]. Recently, DP has been applied to the iterative training of DNNs using *stochastic gradient descent* (DP-SGD) [1]

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

and the FL algorithm of *federated averaging* (DP-FedAvg) [47], 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 [45] (referred to as *the general DP framework* hereafter), which applies a *subsampled Gaussian mechanism* (Sec. 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 [45] can hide the participation of any single record in the algorithm in a principled way. For example, DP-FedAvg [47] guarantees (with high probability) 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, FTS [12] is not amenable to a straightforward integration of the general DP framework [45] (Sec. 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 theoretical challenge of accounting for the impact of the integration of 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 [45], 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) [12]. 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 a 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 *Gaussian process* (GP) surrogate (i.e., the surrogate used by BO to model the objective function) can model the objective function more accurately in a smaller local sub-region [21]. 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 [45] (Sec. 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 (Sec. 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 paper.

In this paper, we introduce the *differentially private FTS with DE* (DP-FTS-DE) algorithm (Sec. 3), the first algorithm with a rigorous guarantee on the user-level privacy in the FBO setting. 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* (Sec. 4). Next, we empirically demonstrate that DP-FTS-DE delivers an effective performance with a strong privacy guarantee and induces a favorable trade-off between privacy and utility in real-world applications (Sec. 5).

2 Background

Bayesian optimization (BO). BO aims to maximize an objective function f on a domain $\mathcal{X} \subset \mathbb{R}^D$ through sequential queries, i.e., to find $\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$.[3] Specifically, in iteration $t \in [T]$ (we use $[N]$ to represent $\{1, \ldots, N\}$ for brevity), an input $\mathbf{x}_t \in \mathcal{X}$ is selected to be queried to yield

[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 [47].

[2] Following [47], we assume that the central server is trustworthy and that the clients are untrustworthy.

[3] For simplicity, we assume $\mathcal{X}$ to be discrete, but our theoretical analysis can be extended to compact domains through suitable discretizations [7].

an output observation: $y_t \triangleq f(\mathbf{x}_t) + \zeta$ where ζ is sampled from a Gaussian noise with variance σ^2: $\zeta \sim \mathcal{N}(0, \sigma^2)$. To select the $\mathbf{x}_t$'s intelligently, BO uses a *Gaussian process* (GP) [54] as a surrogate to model the objective function f. A GP is defined by its mean function μ and kernel function k. We assume w.l.o.g. that $\mu(\mathbf{x}) = 0$ and $k(\mathbf{x}, \mathbf{x}') \leq 1, \forall \mathbf{x}, \mathbf{x}' \in \mathcal{X}$. We mainly focus on the widely used *squared exponential* (SE) kernel in this work. In iteration $t+1$, given the first t input-output pairs, the GP posterior is given by $\mathcal{GP}(\mu_t(\cdot), \sigma_t^2(\cdot, \cdot))$ where

$$\mu_t(\mathbf{x}) \triangleq \mathbf{k}_t(\mathbf{x})^\top (\mathbf{K}_t + \lambda \mathbf{I})^{-1} \mathbf{y}_t \, , \sigma_t^2(\mathbf{x}, \mathbf{x}') \triangleq k(\mathbf{x}, \mathbf{x}') - \mathbf{k}_t(\mathbf{x})^\top (\mathbf{K}_t + \lambda \mathbf{I})^{-1} \mathbf{k}_t(\mathbf{x}') \tag{1}$$

in which $\mathbf{k}_t(\mathbf{x}) \triangleq (k(\mathbf{x}, \mathbf{x}_{t'}))_{t' \in [t]}^\top$, $\mathbf{y}_t \triangleq (y_{t'})_{t' \in [t]}^\top$, $\mathbf{K}_t \triangleq (k(\mathbf{x}_{t'}, \mathbf{x}_{t''}))_{t', t'' \in [t]}$ and $\lambda > 0$ is a regularization parameter [7]. In iteration $t+1$, the GP posterior (1) is used to select the next query $\mathbf{x}_{t+1}$. For example, the *Thompson sampling* (TS) [7] algorithm firstly samples a function f_{t+1} from the GP posterior (1) and then chooses $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} f_{t+1}(\mathbf{x})$. BO algorithms are usually analyzed in terms of *regret*. A hallmark for well-performing BO algorithms is to be asymptotically *no-regret*, which requires the *cumulative regret* $R_T \triangleq \sum_{t=1}^T (f(\mathbf{x}^*) - f(\mathbf{x}_t))$ to grow sub-linearly.

Random Fourier features (RFFs) [53] has been adopted to approximate the kernel function k using M-dimensional random features $\boldsymbol{\phi}$: $k(\mathbf{x}, \mathbf{x}') \approx \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')$. RFFs offers a high-probability guarantee on the approximation quality: $\sup_{\mathbf{x}, \mathbf{x}' \in \mathcal{X}} |k(\mathbf{x}, \mathbf{x}') - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}')| \leq \varepsilon$ where $\varepsilon = \mathcal{O}(M^{-1/2})$ [53]. Of note, RFFs makes it particularly convenient to approximately sample a function from the GP posterior. Specifically, define $\boldsymbol{\Phi}_t \triangleq (\boldsymbol{\phi}(\mathbf{x}_{t'}))_{t' \in [t]}^\top$ (i.e., a $t \times M$-dimensional matrix), $\boldsymbol{\Sigma}_t \triangleq \boldsymbol{\Phi}_t^\top \boldsymbol{\Phi}_t + \lambda \mathbf{I}$, and $\boldsymbol{\nu}_t \triangleq \boldsymbol{\Sigma}_t^{-1} \boldsymbol{\Phi}_t^\top \mathbf{y}_t$. To sample a function $\widetilde{f}$ from approximate GP posterior, we only need to sample

$$\boldsymbol{\omega} \sim \mathcal{N}(\boldsymbol{\nu}_t, \lambda \boldsymbol{\Sigma}_t^{-1}) \tag{2}$$

and set $\widetilde{f}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}, \forall \mathbf{x} \in \mathcal{X}$. RFFs has been adopted by FTS [12] since it allows avoiding the sharing of raw data and improves the communication efficiency [12].

Federated Bayesian Optimization (FBO). 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 $\mathbf{x}_t^n$ and observing $y_t^n, \forall t \in [T]$. 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 Sec. 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 the work of [12], 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, 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., $\|f^n\|_k \leq B, \forall n \in [N]$. The work of [12] has introduced the FTS algorithm for the FBO setting. In iteration $t+1$ of FTS, every agent $\mathcal{A}_n$ ($2 \leq n \leq N$) samples a vector $\boldsymbol{\omega}_{n,t}$ from its GP posterior (2) and sends it to $\mathcal{A}_1$. Next, with probability $p_t \in (0, 1]$, $\mathcal{A}_1$ chooses the next query using a function f_{t+1}^1 sampled from its own GP posterior: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} f_{t+1}^1(\mathbf{x})$; with probability $1 - p_t$, $\mathcal{A}_1$ firstly randomly samples an agent $\mathcal{A}_n$ ($2 \leq n \leq N$) and then chooses $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}$.

Differential Privacy (DP). DP provides a rigorous framework for privacy-preserving data release [19]. Consistent with that of [47], 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 1. *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 has an imperceptible impact on the output of the algorithm. DP-FedAvg [47] has added user-level DP into the FL setting by adopting a general DP framework [45]. 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 so that its L_2 norm is upper-bounded by S,
3. Add Gaussian noise (variance proportional to S^2) to the weighted average of 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 [1] to calculate the *privacy loss*.[4] More recently, the work of [45] has formalized the methods used by [1] and [47] as a general DP framework that is applicable to generic iterative algorithms. Notably, the general DP framework [45] 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 (Sec. 3.2) into our algorithm.

3 Differentially Private FTS with Distributed Exploration

Here we firstly introduce how we modify FTS to integrate DP to derive the DP-FTS algorithm (Sec. 3.1). Then, we describe distributed exploration which can be seamlessly integrated into DP-FTS to further improve utility (Sec. 3.2). Lastly, we present the complete DP-FTS-DE algorithm (Sec. 3.3).

3.1 Differentially Private Federated Thompson Sampling (DP-FTS)

The original FTS algorithm [12] is not amenable to a straightforward integration of the general DP framework. This is because FTS [12] requires an agent to receive all vectors $\boldsymbol{\omega}_{n,t}$'s from the other agents (Sec. 2), so the transformations to the vectors required by the general DP framework (e.g., subsampling and weighted averaging of the vectors) cannot be easily incorporated. Therefore, we modify FTS by (a) adding a central server for performing the privacy-preserving transformations, and (b) passing a single aggregated vector (instead of one vector from every agent) to the agents. Fig. 1a 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, ② (by Agents): Every agent $\mathcal{A}_n$ samples a vector $\boldsymbol{\omega}_{n,t}$ (2) 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, ④ (by Central Server): Next, the central server processes the N received vectors $\boldsymbol{\omega}_{n,t}$ 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 $\boldsymbol{\omega}_{n,t}$ of every selected agent s.t. its L_2 norm is upper-bounded by S, and (c) calculates a *weighted average* of the clipped vectors using weights $\{\varphi_n, \forall n \in [N]\}$, 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 of $\mathcal{GP}(\mu^n_t(\cdot), \beta^2_{t+1}\sigma^n_t(\cdot,\cdot)^2)$ (1) where $\beta_t \triangleq B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)}$,[5] 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 that of [12], $(p_t)_{t\in\mathbb{Z}^+}$ is chosen as a monotonically increasing sequence such that $p_t \in (0, 1], \forall t$ and $p_t \to 1$ as $t \to \infty$. This choice of the sequence of $(p_t)_{t\in\mathbb{Z}^+}$ ensures that in the early stage (i.e., when $1 - p_t$ is large), an agent can leverage the information from the other agents (via $\boldsymbol{\omega}_t$) to improve its convergence by accelerating its exploration.

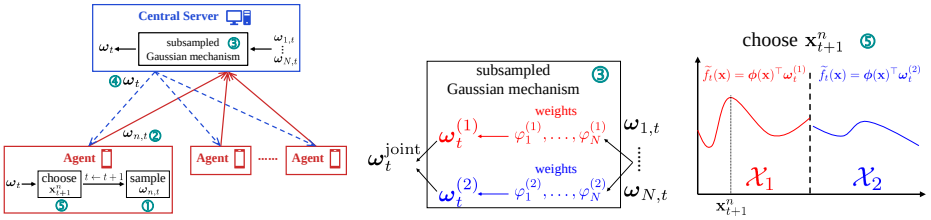
After choosing $\mathbf{x}^n_{t+1}$ and observing y^n_{t+1}, $\mathcal{A}_n$ adds $(\mathbf{x}^n_{t+1}, y^n_{t+1})$ to its history and samples a new vector $\boldsymbol{\omega}_{n,t+1}$ (step ①). Next, $\mathcal{A}_n$ sends $\boldsymbol{\omega}_{n,t+1}$ to the central server (step ②), and the algorithm is repeated. The detailed algorithm will be presented in Sec. 3.3 since DP-FTS is equivalent to the DP-FTS-DE algorithm with $P = 1$ sub-region.

3.2 Distributed Exploration (DE)

To accelerate the practical convergence (utility) of DP-FTS, we introduce DE to further accelerate the exploration in the early stages (Sec. 1). At the beginning of BO, 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 [21], and leverage the ability of the general DP framework to handle different parameter vectors [45] to integrate DE into DP-FTS in a seamless way.

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

[5] γ_{t-1} is the max information about the objective function from any $t-1$ queries and $\delta \in (0, 1)$ (Theorem 1).



(a) (b) (c)

Figure 1: (a) DP-FTS algorithm (without distributed exploration). (b-c) Replacing steps ③ and ⑤ in (a) with that in (b) and (c) respectively to derive the DP-FTS-DE algorithm ($P = 2$).

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 $\mathcal{X}_{i_n}$.[6] 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 $\boldsymbol{\omega}_{n,t}$ (2) sent to the central server is more informative about its objective function *in this sub-region* $\mathcal{X}_{i_n}$.[7] As a result, for a sub-region $\mathcal{X}_i$, the vectors from those 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 $\{\boldsymbol{\omega}_t^{(i)}, \forall i \in [P]\}$ with each $\boldsymbol{\omega}_t^{(i)}$ using a separate set of weights $\{\varphi_n^{(i)}, \forall n \in [N]\}$. Interestingly, from the perspective of the general DP framework [45], the different vectors $\{\boldsymbol{\omega}_t^{(i)}, \forall i \in [P]\}$ 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, 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 (Fig. 1c); see more details in Sec. 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 $\{\varphi_n^{(i)}, \forall n \in [N]\}$ *adaptive* such that they (a) give more weights to those agents exploring $\mathcal{X}_i$ when t is small and (b) gradually become uniform among all agents as t becomes large. We adopt the widely used softmax weighting with temperature $\mathcal{T}$, which has been shown to provide well-calibrated uncertainty for weighted ensemble of GP experts [9]. Concretely, we let $\varphi_n^{(i)} = \frac{\exp((a\mathbb{I}_n^{(i)}+1)/\mathcal{T})}{\sum_{n=1}^N \exp((a\mathbb{I}_n^{(i)}+1)/\mathcal{T})}$,[8] and gradually increase the temperature $\mathcal{T}$ from 1 to $+\infty$ (more details in App. B). The dependence of the weights on t only requires minimal modifications to the algorithm and the theoretical results. So, we drop this dependence for simplicity.

3.3 DP-FTS-DE Algorithm

Our complete DP-FTS-DE algorithm after integrating DE (Sec. 3.2) into DP-FTS (Sec. 3.1) is presented in Algo. 1 (central server's role) and Algo. 2 (agent's role), with the steps in circle corresponding to those in Fig. 1a. 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 Algo. 2). Secondly, instead of a single vector $\boldsymbol{\omega}_t$, the central server produces and broadcasts P vectors $\{\boldsymbol{\omega}_t^{(i)}, \forall i \in [P]\}$, each corresponding to a different sub-region $\mathcal{X}_i$ and using a different set of weights. Applying different transformations to different vectors (e.g., parameters of different DNN layers) can be naturally incorporated into the general DP framework [45]. Different transformations performed by our central server to produce P vectors can be interpreted as *a single subsampled*

[6] 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.

[7] Because its GP surrogate can model the objective function in this local sub-region more accurately [21].

[8] $\mathbb{I}_n^{(i)}$ is an indicator variable that equals 1 if agent n is assigned to explore $\mathcal{X}_i$ and equals 0 otherwise. $a > 0$ is a constant and we set $a = 15$ in all our experiments.

Gaussian mechanism producing a single joint vector $\boldsymbol{\omega}_t^{\text{joint}} \triangleq (\boldsymbol{\omega}_t^{(i)})_{i\in[P]}$ (Fig. 1b). Next, we present these transformations performed by the central server (lines 5-12 of Algo. 1) from this perspective.

Subsampling: To begin with, after receiving the vectors $\boldsymbol{\omega}_{n,t}$'s from the agents (lines 3-4 of Algo. 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: $\boldsymbol{\omega}_{n,t}^{\text{joint}} \triangleq (N\varphi_n^{(i)}\boldsymbol{\omega}_{n,t})_{i\in[P]}$.

Algorithm 1 DP-FTS-DE (central server)

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

Algorithm 2 BO-Agent-$\boldsymbol{\mathcal{A}_n}(t, \boldsymbol{\omega}_{t-1}^{\text{joint}} = (\boldsymbol{\omega}_{t-1}^{(i)})_{i\in[P]})$

1: **if** $t = 0$ **then**
2: Randomly select and query N_{init} initial points from sub-region $\mathcal{X}_{i_n}$
3: **else**
4: Sample r from the uniform distribution over $[0, 1]$: $r \sim U(0, 1)$
5: **if** $r \leq p_t$ **then**
6: $\mathbf{x}_t^n = \arg\max_{\mathbf{x}\in\mathcal{X}} f_t^n(\mathbf{x})$, where $f_t^n \sim \mathcal{GP}(\mu_t^n(\cdot), \beta_{t+1}^2\sigma_t^n(\cdot,\cdot)^2)$ ⑤
7: **else**
8: $\mathbf{x}_t^n = \arg\max_{\mathbf{x}\in\mathcal{X}} \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{t-1}^{(i^{[\mathbf{x}]})}$.[9]
9: Query $\mathbf{x}_t^n$ to observe y_t^n
10: Sample $\boldsymbol{\omega}_{n,t}$ and send it to central server ①, ②

Clipping: Next, clip every selected vector $\boldsymbol{\omega}_{n,t}$ to obtain $\widehat{\boldsymbol{\omega}}_{n,t}$ such that $||\widehat{\boldsymbol{\omega}}_{n,t}||_2 \leq S/\sqrt{P}$. This is equivalent to clipping $\boldsymbol{\omega}_{n,t}^{\text{joint}}$ to obtain: $\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}} \triangleq (N\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t})_{i\in[P]}$, whose L_2 norm is bounded by $||\widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}||_2 \leq (N^2\varphi_{\max}^2 \sum_{i=1}^P \|\widehat{\boldsymbol{\omega}}_{n,t}\|_2^2)^{1/2} \leq N\varphi_{\max}S$ where $\varphi_{\max} \triangleq \max_{i\in[P],n\in[N]} \varphi_n^{(i)}$.

Weighted Average: Next, calculate a weighted average of the clipped joint vectors by giving equal weights[10] to all agents: $\boldsymbol{\omega}_t^{\text{joint}} = (qN)^{-1} \sum_{n\in\mathcal{S}_t} \widehat{\boldsymbol{\omega}}_{n,t}^{\text{joint}}$.[11] Note that $\boldsymbol{\omega}_t^{\text{joint}}$ results from the concatenation of the vectors from all sub-regions: $\boldsymbol{\omega}_t^{\text{joint}} = (\boldsymbol{\omega}_t^{(i)})_{i\in[P]}$ where $\boldsymbol{\omega}_t^{(i)} = (qN)^{-1} \sum_{n\in\mathcal{S}_t} N\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t} = q^{-1} \sum_{n\in\mathcal{S}_t} \varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}$ (line 10 of Algo. 1).

Gaussian Noise: Finally, add to each element of $\boldsymbol{\omega}_t^{\text{joint}} = (\boldsymbol{\omega}_t^{(i)})_{i\in[P]}$ 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 $\boldsymbol{\omega}_t^{\text{joint}} = (\boldsymbol{\omega}_t^{(i)})_{i\in[P]}$ 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 $\mathcal{GP}(\mu_t^n(\cdot), \beta_{t+1}^2\sigma_t^n(\cdot,\cdot)^2)$ (1) (line 6 of Algo. 2, $\beta_t \triangleq B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)^5)}$; with probability $1 - p_{t+1}$, $\mathcal{A}_n$ uses $\boldsymbol{\omega}_t^{\text{joint}} = [\boldsymbol{\omega}_t^{(i)}]_{i\in[P]}$ received from

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

[10] This weight is only used to aid interpretation and is different from the weights $\varphi_n^{(i)}$'s used in our algorithm.

[11] The summation is divided by qN (i.e., expected number of agents selected) to make it unbiased [47].

the central server to choose $\mathbf{x}_{t+1}^n$ by maximizing the reconstructed functions for all sub-regions (line 8 of Algo. 2), as illustrated in Fig. 1c. 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 10 of Algo. 2), and the algorithm is repeated.

4 Theoretical Analysis

In this section, we present theoretical guarantees on both the privacy and utility of our DP-FTS-DE, which combine to yield interesting insights about the *privacy-utility trade-off*.

Proposition 1 (Privacy Guarantee)**.** *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 (Algo. 1) is (ϵ, δ)-DP if $z \geq c_2 q \sqrt{T \log(1/\delta)}/\epsilon$.*

Proposition 1 formalizes our privacy guarantee, and its proof (App. A.1) follows directly from Theorem 1 of [1]. Proposition 1 shows that a larger z (i.e., larger variance for Gaussian noise), a smaller q (i.e., smaller expected no. of selected agents) and a smaller T (i.e., smaller no. of iterations) all improve the privacy guarantee because for a fixed δ, they all allow ϵ to be smaller.

Theorem 1 (Utility Guarantee)**.** *Assume that all f^n's lie in the RKHS of kernel k: $\|f^n\|_k \leq B, \forall n \in [N]$. Let γ_t be the max information gain on f^1 from any set of t observations, ε denote an upper bound on the approximation error of RFFs (Sec. 2), $\mathcal{C}_t \triangleq \{n \in [N] \big| ||\boldsymbol{\omega}_{n,t}||_2 > S/\sqrt{P}\}$, $\delta \in (0,1)$, $\lambda = 1 + 2/T$ and $\beta_t \triangleq B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)}$ (Algo. 2). Choose $(p_t)_{t \in \mathbb{Z}^+}$ to be monotonically increasing s.t. $1 - p_t = \mathcal{O}(1/t^2)$. Then, with probability of at least $1 - \delta$ ($\tilde{\mathcal{O}}$ hides all logarithmic factors),*

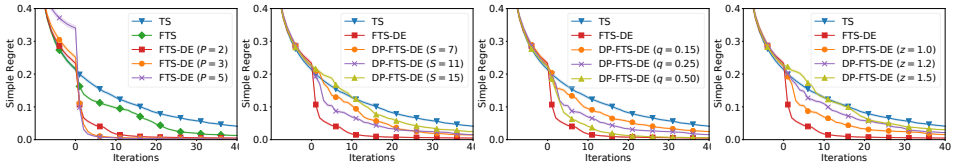
$$R_T^1 = \tilde{\mathcal{O}}\Big(\big(B + 1/p_1\big)\, \gamma_T \sqrt{T} + \textstyle\sum_{t=1}^T \psi_t + B \sum_{t=1}^T \vartheta_t \Big)$$

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

Theorem 1 (proof in App. A.2) gives an upper bound on the cumulative regret of agent $\mathcal{A}_1$. Note that all three terms in the regret upper bound grow sub-linearly: The first term is sub-linear because $\gamma_T = \mathcal{O}((\log T)^{D+1})$ for the SE kernel (D is the dimension of the input space), and the second and third terms are both sub-linear since we have chosen $1 - p_t = \mathcal{O}(1/t^2)$.[12] So, DP-FTS-DE preserves the no-regret property of FTS [12]. That is, agent $\mathcal{A}_1$ is asymptotically *no-regret* even if all other agents are heterogeneous, i.e., all other agents have significantly different objective functions from $\mathcal{A}_1$. This is particularly desirable because it ensures the robustness of DP-FTS-DE against the heterogeneity of agents, which is an important challenge in both FL and FBO [12, 29] and has also been an important consideration for other works on the theoretical convergence of FL [43, 44]. To prove this robust guarantee, we have upper-bounded the *worst-case error* induced by the use of *any* set of agents, which explains the dependence of the regret bound on $d_n \triangleq \max_{\mathbf{x} \in \mathcal{X}} |f^1(\mathbf{x}) - f^n(\mathbf{x})|$ 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 information from the other agents (line 8 of Algo. 2 becomes more likely to be executed), hence inflating the worst-case error resulting from these information.

Theorem 1, when interpreted together with Proposition 1, also reveals 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 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 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) [1]. 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 (Theorem 1). Intuitively, a

[12] The requirement of $1 - p_t = \mathcal{O}(1/t^2)$ is needed only to ensure that DP-FTS-DE is no-regret even if all other agents are heterogeneous and is hence a conservative choice. Therefore, it is reasonable to make $1 - p_t$ decrease slower in practice as we show in our experiments (Sec. 5).



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

Figure 2: (a) Benefit of DE. (b) Impact of S. (c) Impact of q; privacy losses after $T = 40$ iterations are $5.93, 9.91, 20.12$ for the respective $q = 0.15, 0.25, 0.5$. (d) Impact of z; privacy loss are $9.91, 7.39, 5.22$ for the respective $z = 1.0, 1.2, 1.5$. Every curve is averaged over $N = 200$ agents, each further averaged over 5 runs with different random initializations of size $N_{\text{init}} = 10$. The results before iteration 0 correspond to the initialization period.

smaller S impacts the performance positively by reducing the noise variance (line 11 of Algo. 1) and yet negatively by causing more vectors to be clipped (line 9 of Algo. 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.

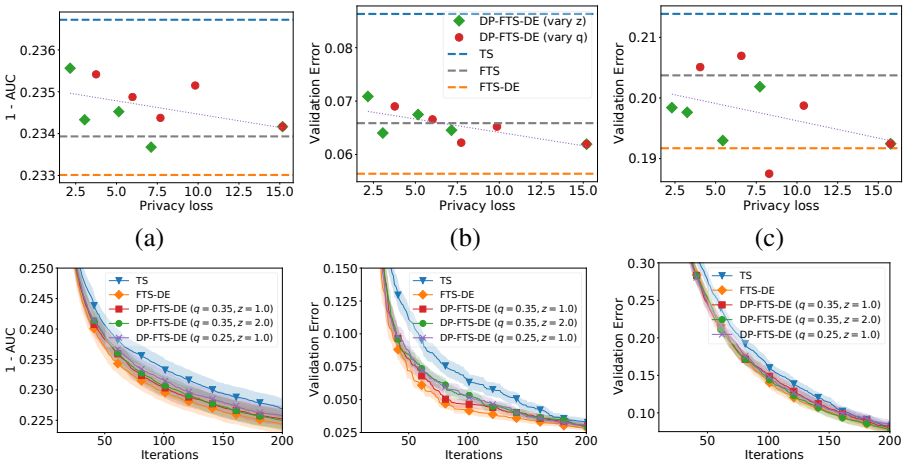
Regarding the dependence of the regret upper bound on the number of random features M, in addition to the dependence through $\Delta_{n,t}$ which has been analyzed by [12], the integration of DP introduces another dependence that implicitly affects the **privacy-utility trade-off**. Besides increasing 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 (Theorem 1). So, the additional dependence due to the integration of DP loosens the regret upper bound with an increasing M. As a result, if M is larger, we can either *sacrifice privacy to preserve utility* by reducing z or increasing q (both can counter the increase of ψ_t), or *sacrifice utility to preserve privacy* by keeping z and q unchanged. The number of sub-regions P also induces a trade-off about the performance of our algorithm, which is partially reflected by Theorem 1. Specifically, the regret bound depends on P through three terms. Two of the terms (P in ψ_t and the summation of P terms in ϑ_t) arise due to the worst-case nature of our regret 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: $\{\boldsymbol{\omega}_t^{(i)}, \forall i \in [P]\}$. Regarding the third term (in the definition of $\mathcal{C}_t$), a larger 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 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 (Sec. 3.2). As a result of the worst-case nature of the regret bound mentioned earlier, the latter positive effect leading to better practical utility is not reflected by Theorem 1. Therefore, we instead verify this trade-off induced by P about the practical performance in our experiments (Fig. 6 in App. B.1).

5 Experiments

Note that the privacy loss calculated by the moments accountant method is an upper bound on the value of ϵ for a given value of δ.[4] When calculating the privacy loss, we follow the practice of [47] and set $\delta = 1/N^{1.1}$. All error bars denote standard errors. Due to space constraints, some experimental details are deferred to App. B.

5.1 Synthetic Experiments

In synthetic experiments, we firstly 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. 2a, FTS converges faster than standard TS and more importantly, FTS-DE (Sec. 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 (Sec. 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) than DP-FTS for the same level of privacy guarantee (Fig. 4 in App. B.1). These results justify the practical significance of our DE technique (Sec. 3.2). We have also shown (Fig. 5, App. B.1) that both components in our



(a) (b) (c)

(d) (e) (f)

Figure 3: (a,b,c) 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). (d,e,f) Convergence results for some settings in each experiment. Results are averaged over N agents, each further averaged over 100 (a,d) and 10 (b,c,e,f) random initializations ($N_{\text{init}} = 10$). Some error bars overlap because the agents are highly heterogeneous and thus have significantly different scales.

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. 2b explores the impact of the clipping threshold S. From Fig. 2b, 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 Sec. 4. The value of $S = 11$, which delivers the best performance in Fig. 2b, has been chosen such that only a small percentage (0.8%) of the vectors are clipped. Figs. 2c and d show 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 (Sec. 4), i.e., a larger q (smaller z) leads to a better utility at the expense of a greater privacy loss. Lastly, we also verify the robustness of our algorithm against agent heterogeneity. In particular, we show that when the objective functions of different agents are significantly different, our FTS-DE algorithm is still able to perform comparably with standard TS and letting the impact of the other agents decay faster (i.e., letting $1 - p_t$ decrease faster) can improve the performance of FTS-DE in this scenario (see Fig. 7 in App. B.1.2).

5.2 Real-World Experiments

We adopt 3 commonly used datasets in FL and FBO [12, 60]. We firstly use a landmine detection dataset with $N = 29$ landmine fields [66] and tune 2 hyperparameters of SVM for landmine detection. Next, we use data collected using mobile phone sensors when $N = 30$ subjects are performing 6 activities [2] and tune 3 hyperparameters of logistic regression for activity classification. Lastly, we use the images of handwritten characters by $N = 50$ persons from EMNIST (a commonly used benchmark in FL) [8] and tune 3 hyperparameters of a convolutional neural network used for image classification. In all 3 experiments, we choose $P = 4$, $S = 22.0$, $M = 100$, and $1 - p_t = 1/t$. In the practical deployment of our algorithm, if the values of these parameters are tuned by running additional experiments, the additional privacy loss can be easily accounted for using existing techniques [1].

Figs. 3a,b,c plot the privacy (horizontal axis, more to the left is better) and utility (vertical axis, lower is better) after 60 iterations[13]. 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 (in the single digit range), DP-FTS-DE achieves a competitive performance (utility) and significantly outperforms standard TS in all settings. The figures also reveal a clear trade-off between privacy and

[13] In practice, we recommend that the agents switch to local TS after the value of $1 - p_t$ becomes so small (e.g., after 60 iterations) that the probability of using the information from the central server is negligible (i.e., line 8 of Algo. 2 is unlikely to be executed), after which no privacy loss is incurred.

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. 3d,e,f which plot some convergence results: 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. 3a,b,c also justify the importance of DE (Sec. 3.2) since FTS-DE (and some settings of DP-FTS-DE) outperforms FTS without DE in all experiments. We also verify the importance of DE when DP is integrated in App. B.2 (Fig. 8). Furthermore, we demonstrate the robustness of our results against the choices of the weights and the number of sub-regions in App. B.2.3. Lastly, our DP-FTS-DE can be easily adapted to use Rényi DP [63][14], which, although requires modifications to our theoretical analysis, offers slightly better privacy loss with comparable performances (Fig. 11 in App. B.2).

6 Related Works

BO has been extensively studied recently under different settings [3, 7, 25, 34, 51, 52, 56, 61]. Recent works have added privacy preservation to BO by applying DP to the output of BO [36], using a different notion of privacy other than DP [48], adding DP to outsourced BO [33], or adding local DP to BO [71]. However, none of these works can tackle the FBO setting considered in this paper. Collaborative BO involving multiple agents has also been considered by the work of [59], however, [59] has assumed that all agents share the same objective function and focused on the issue of fairness. Moreover, BO in the multi-agent setting has also been studied from the perspective of game theory [11, 55]. Our method also shares similarity with parallel BO [10, 14, 15, 23, 24, 31]. However, parallel BO optimizes a single objective function while we allow agents to have different objective functions. Our algorithm is also related to multi-fidelity BO [13, 30, 68, 69] because utilizing information received from the central server can be viewed as querying a low-fidelity function. Our DE technique bears similarity to [21] which has also used separate GP surrogates to model different local sub-regions (hyper-rectangles) and shown significantly improved performance.

FL has attracted significant attention in recent years [29, 37, 38, 39, 40, 42, 46]. In particular, privacy preservation using DP has been an important topic for FL [27, 41, 62, 65], including both central DP (with trusted central server) [47] and local [32, 64, 70] or distributed DP [4, 6, 18, 58] (without trusted central server). In addition to our setting of FBO which can be equivalently called *federated GP bandit*, other sequential decision-making problems have also been extended to the federated setting, including federated multi-armed bandit [57, 72], federated linear bandit [17], and federated reinforcement learning [22]. Lastly, federated hyperparameter tuning (i.e., hyperparameter tuning of ML models in the federated setting) has been attracting growing attention recently [26, 35].

7 Conclusion and Future Works

We introduce DP-FTS-DE, which equips FBO with rigorous privacy guarantee and is amenable to privacy-utility trade-off both theoretically and empirically. Since our method aims to promote larger-scale adoption of FBO, it may bring the negative societal impact of fairness: some users may become discriminated against by the algorithm. This can be mitigated by extending our method to consider fairness [59], suggesting an interesting future work. A limitation of our work is that the privacy loss offered by the moments accountant method [1] is not state-of-the-art. For example, the more advanced privacy-preserving technique of Gaussian DP [5, 16] can deliver smaller privacy losses than moments accountant and has been widely applied in practice. So, a potential future work is to extend our method to adopt more advanced privacy-preserving techniques such as Gaussian DP. Another limitation of our paper is that we have not accounted for the different network capabilities of different agents, which is a common problem in the federated setting. That is, our algorithm requires every agent to send its updated parameter vector to the central server in every iteration (lines 3 and 4 of Algo. 1), which may not be realistic in some scenarios since the messages from some agents may not be received by the central server. Therefore, accounting for this issue using a principled method represents an interesting future work. Other potential future works include adapting our method for other sequential decision-making problems such as reinforcement learning [22], and incorporating risk aversion [49, 50] into our method for safety-critical domains such as clinical applications (Sec. 1).

[14] We only need to modify step 6 of Algo. 1 s.t. we randomly select a fixed number of Nq agents.

Acknowledgments and Disclosure of Funding

This research/project is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References

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

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

[3] S. Balakrishnan, Q. P. Nguyen, B. K. H. Low, and H. Soh. Efficient exploration of reward functions in inverse reinforcement learning via Bayesian optimization. In *Proc. NeurIPS*, 2020.

[4] A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnes, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *Proc. SOSP*, pages 441–459, 2017.

[5] Z. Bu, J. Dong, Q. Long, and W. J. Su. Deep learning with gaussian differential privacy. *Harvard data science review*, 2020(23), 2020.

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

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

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

[9] S. Cohen, R. Mbuvha, T. Marwala, and M. Deisenroth. Healing products of gaussian process experts. In *International Conference on Machine Learning*, pages 2068–2077. PMLR, 2020.

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

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

[12] Z. Dai, B. K. H. Low, and P. Jaillet. Federated Bayesian optimization via Thompson sampling. In *Proc. NeurIPS*, 2020.

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

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

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

[16] J. Dong, A. Roth, and W. J. Su. Gaussian differential privacy. *arXiv preprint arXiv:1905.02383*, 2019.

[17] A. Dubey and A. Pentland. Differentially-private federated linear bandits. In *Proc. NeurIPS*, 2020.

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

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

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

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

[22] X. Fan, Y. Ma, Z. Dai, W. Jing, C. Tan, and B. K. H. Low. Fault-tolerant federated reinforcement learning with theoretical guarantee. In *Proc. NeurIPS*, 2021.

[23] J. Garcia-Barcos and R. Martinez-Cantin. Fully distributed Bayesian optimization with stochastic policies. In *Proc. IJCAI*, 2019.

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

[25] T. N. Hoang, Q. M. Hoang, R. Ouyang, and B. K. H. Low. Decentralized high-dimensional Bayesian optimization with factor graphs. In *Proc. AAAI*, volume 32, 2018.

[26] S. Holly, T. Hiessl, S. R. Lakani, D. Schall, C. Heitzinger, and J. Kemnitz. Evaluation of hyperparameter-optimization approaches in an industrial federated learning system. arXiv:2110.08202, 2021.

[27] R. Hu, Y. Guo, H. Li, Q. Pei, and Y. Gong. Personalized federated learning with differential privacy. *IEEE Internet of Things Journal*, 7(10):9530–9539, 2020.

[28] Z. Ji, Z. C. Lipton, and C. Elkan. Differential privacy and machine learning: a survey and review. arXiv:1412.7584, 2014.

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

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

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

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

[33] D. Kharkovskii, Z. Dai, and B. K. H. Low. Private outsourced Bayesian optimization. In *Proc. ICML*, pages 5231–5242. PMLR, 2020.

[34] D. Kharkovskii, C. K. Ling, and B. K. H. Low. Nonmyopic Gaussian process optimization with macro-actions. In *Proc. AISTATS*, pages 4593–4604. PMLR, 2020.

[35] M. Khodak, R. Tu, T. Li, L. Li, M.-F. Balcan, V. Smith, and A. Talwalkar. Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing. arXiv:2106.04502, 2021.

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

[37] Q. Li, Y. Diao, Q. Chen, and B. He. Federated learning on non-iid data silos: An experimental study. arXiv:2102.02079, 2021.

[38] Q. Li, B. He, and D. Song. Model-contrastive federated learning. In *Proc. CVPR*, pages 10713–10722, 2021.

[39] Q. Li, Z. Wen, and B. He. Federated learning systems: Vision, hype and reality for data privacy and protection. arXiv:1907.09693, 2019.

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

[41] Q. Li, Z. Wu, Z. Wen, and B. He. Privacy-preserving gradient boosting decision trees. In *Proc. AAAI*, pages 784–791, 2020.

[42] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. arXiv:1908.07873, 2014.

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

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

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

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

[47] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang. Learning differentially private recurrent language models. In *Proc. ICLR*, 2018.

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

[49] Q. P. Nguyen, Z. Dai, B. K. H. Low, and P. Jaillet. Optimizing conditional Value-At-Risk of black-box functions. In *Proc. NeurIPS*, 2021.

[50] Q. P. Nguyen, Z. Dai, B. K. H. Low, and P. Jaillet. Value-at-Risk optimization with Gaussian processes. In *Proc. ICML*, 2021.

[51] Q. P. Nguyen, S. Tay, B. K. H. Low, and P. Jaillet. Top-k ranking Bayesian optimization. In *Proc. AAAI*, volume 35, pages 9135–9143, 2021.

[52] Q. P. Nguyen, Z. Wu, B. K. H. Low, and P. Jaillet. Trusted-maximizers entropy search for efficient Bayesian optimization. In *Proc. UAI*, 2021.

[53] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proc. NeurIPS*, pages 1177–1184, 2008.

[54] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

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

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

[57] C. Shi, C. Shen, and J. Yang. Federated multi-armed bandits with personalization. In *Proc. AISTATS*, pages 2917–2925. PMLR, 2021.

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

[59] R. H. L. Sim, Y. Zhang, B. K. H. Low, and P. Jaillet. Collaborative Bayesian optimization with fair regret. In *Proc. ICML*, pages 9691–9701. PMLR, 2021.

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

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

[62] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, and Y. Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 1–11, 2019.

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

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

[65] K. Wei, J. Li, M. Ding, C. Ma, H. H. Yang, F. Farokhi, S. Jin, T. Q. Quek, and H. V. Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020.

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

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

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

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

[70] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal*, 8(11):8836–8853, 2020.

[71] X. Zhou and J. Tan. Local differential privacy for Bayesian optimization. arXiv:2010.06709, 2020.

[72] Z. Zhu, J. Zhu, J. Liu, and Y. Liu. Federated bandit: A gossiping approach. In *Abstract Proceedings of the 2021 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, pages 3–4, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Sec. 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Sections 2 and 4.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A.
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our code is here: `https://github.com/daizhongxiang/Differentially-Private-Federated-Bayesian-Optimization`

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Figures 2 (the error bars can be better seen after zooming in) and 3.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] See Appendix B.

(b) Did you mention the license of the assets? [Yes] See Appendix B.

(c) Did you include any new assets either in the supplemental material or as a URL? [No]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] See Appendix B.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Appendix B.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proof of Theoretical Results

A.1 Proof of Proposition 1

Proposition 1 follows directly from the DP guarantee of the works of [1] and [47] (e.g., Theorem 1 of [1]). Therefore, to prove the validity of Proposition 1, we only need to show that the joint subsampled Gaussian mechanism we apply in every iteration (Sec. 3.3) is the same as the one adopted by [1] and [47]. Therefore, we demonstrate here that the interpretation of our privacy-preserving transformations as a single subsampled Gaussian mechanism, which we have described in Sec. 3.3, is equivalent to the transformations adopted by the work of [47].

Firstly, our subsampling step (step 6 of Algo. 1) is the same as the one adopted by [47] 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 Algo. 1) to ensure that its L_2 norm is 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 [47], i.e., the unbiased estimator. Specifically, we set the weight (we follow [47] 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 [47], i.e., using Lemma 1 of [47]. 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 [47], 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 Algo. 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 [47]. Therefore, the DP guarantee of [47] and [1] is also valid for our DP-FTS-DE algorithm, hence justifying the validity of our Proposition 1.

A.2 Proof of Theorem 1

In this section, we prove Theorem 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 1 makes use of the proof of [12], 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 Sec. 3.1), (b) the privacy-preserving transformations (lines 5-11 of Algo. 1), and (c) distributed exploration (DE) (Sec. 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 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 2 and 4, 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.

A.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 $\mathbb{I}_n, \forall n \in [N]$, where $\mathbb{P}(\mathbb{I}_n = 1) = q, \forall n \in [N]$. These indicator random variables will be used to account for the subsampling step (i.e., step 6 of Algo. 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

Algo. 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 $\forall \mathbf{x} \in \mathcal{X}$:

$$|\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}| = |\boldsymbol{\phi}(\mathbf{x})^\top \frac{\boldsymbol{\omega}_{n,t}}{\max(1, \frac{\|\boldsymbol{\omega}_{n,t}\|_2}{S/\sqrt{P}})}| = |\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}| \frac{1}{\max(1, \frac{\|\boldsymbol{\omega}_{n,t}\|_2}{S/\sqrt{P}})} \leq |\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}|. \quad (3)$$

Denote by $\boldsymbol{\eta}$ the added Gaussian noise vector (i.e., step 11 of Algo. 1): $\boldsymbol{\eta} \sim \mathcal{N}\left(\mathbf{0}, (z\varphi_{\max} S/q)^2 \mathbf{I}\right)$. Next, define

$$\boldsymbol{\omega}_t^{(i)} \triangleq \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q} + \boldsymbol{\eta}. \quad (4)$$

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 8 of Algo. 2. Note that to simplify the notations in the subsequent analyses, we have slightly deviated from the indexing from Algo. 2 by using $\boldsymbol{\omega}_t^{(i^{[\mathbf{x}]})}$ instead of $\boldsymbol{\omega}_{t-1}^{(i^{[\mathbf{x}]})}$. To be consistent with Algo. 2, we can simply replace all appearances of $\boldsymbol{\omega}_t^{(i^{[\mathbf{x}]})}$ by $\boldsymbol{\omega}_{t-1}^{(i^{[\mathbf{x}]})}$ in our proof.

Let $\delta \in (0, 1)$, recall that we have defined in Theorem 1 that $\beta_t \triangleq B + \sigma\sqrt{2(\gamma_{t-1} + 1 + \log(4/\delta)}$ and define $c_t \triangleq \beta_t(1 + \sqrt{2\log(|\mathcal{X}|t^2)})$ 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 6 of Algo. 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 8 of Algo. 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. Now we define two events that are $\mathcal{F}_{t-1}$-measurable.

Lemma 1 (Lemma 1 of [12]). *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 2 (Lemma 2 of [12]). *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 $t \geq 1$:

$$\begin{aligned} |f(\mathbf{x}) - f_t(\mathbf{x})| &\leq |f(\mathbf{x}) - \mu_{t-1}(\mathbf{x})| + |\mu_{t-1}(\mathbf{x}) - f_t(\mathbf{x})| \\ &= \beta_t \sigma_{t-1}(\mathbf{x}) + \beta_t\sqrt{2\log(|\mathcal{X}|t^2)}\sigma_{t-1}(\mathbf{x}) = c_t \sigma_{t-1}(\mathbf{x}). \end{aligned} \quad (5)$$

Next, 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 2. *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, 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 RFFs approximation at iteration t.

Lemma 3 (Lemma 4 of [12]). *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*

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

Note that a difference between our Lemma 3 above and Lemma 4 of [12] 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

$$\tilde{\Delta}_t^{(i)} \triangleq \sum_{n=1}^{N} \varphi_n^{(i)} \tilde{\Delta}_{n,t}. \tag{6}$$

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 $\mathcal{X}_i, \forall i \in [P]$.

Lemma 4. *Denote by* ε *an upper bound on the approximation error of RFFs approximation (Sec. 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*

$$\begin{aligned}\Delta_{n,t} &\triangleq \tilde{\Delta}_{n,t} + d_n + c_t \\ &= \varepsilon\frac{(t+1)^2}{\lambda}\left(B + \sqrt{2\log\left(\frac{4\pi^2t^2N}{3\delta}\right)}\right) + \beta'_{t+1} + \sqrt{2\log\frac{2\pi^2t^2N}{3\delta} + M} + d_n + c_t \\ &= \tilde{\mathcal{O}}(\varepsilon Bt^2 + B + \sqrt{M} + d_n + \sqrt{\gamma_t}).\end{aligned} \tag{7}$$

Proof.

$$\begin{aligned}|\sum_{n=1}^{N} \varphi_n^{(i)} \widetilde{f}_t^n(\mathbf{x}) - f_t(\mathbf{x})| = |\sum_{n=1}^{N} \varphi_n^{(i)} \widetilde{f}_t^n(\mathbf{x}) - \sum_{n=1}^{N} \varphi_n^{(i)} f_t(\mathbf{x})| &\leq \sum_{n=1}^{N} \varphi_n^{(i)} |\widetilde{f}_t^n(\mathbf{x}) - f_t(\mathbf{x})| \\ &\leq \sum_{n=1}^{N} \varphi_n^{(i)} \Delta_{n,t},\end{aligned} \tag{8}$$

where the last inequality results from Lemma 5 of [12]. □

Note that Lemma 4 above takes into account our modifications to the original FTS algorithm [12] by including a central server and using an aggregation (i.e., weighted average) of the vectors from all agents (first paragraph of Sec. 3.1). Next, define $\Delta_t \triangleq \sum_{n=1}^{N} \Delta_{n,t}$. Note that $\tilde{\Delta}_t^{(i)} \leq \Delta_t^{(i)} \leq \Delta_t, \forall i \in [P]$, and that

$$\sum_{n=1}^{N} \tilde{\Delta}_{n,t} \leq \sum_{n=1}^{N} \Delta_{n,t} = \Delta_t, \tag{9}$$

which will be useful in subsequent proofs.

A.2.2 Main Proof

The following lemma lower-bounds the probability that the selected input $\mathbf{x}_t$ is unsaturated.

Lemma 5 (Lemma 7 of [12]). *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 $P_t \triangleq p_t(p - 1/t^2)$ and $p = \frac{1}{4e\sqrt{\pi}}$.

Proof. Firstly, we have that

$$\mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}\right) \geq \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) \mathbb{P}(A_t) = \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}, A_t\right) p_t. \quad (10)$$

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 [12], 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 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:

$$\overline{\mathbf{x}}_t \triangleq \arg\min_{\mathbf{x} \in \mathcal{X} \setminus S_t} \sigma_{t-1}(\mathbf{x}). \quad (11)$$

Following this definition, for any $\mathcal{F}_{t-1}$ such that $E^f(t)$ is true, we have that

$$\begin{aligned} \mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t) | \mathcal{F}_{t-1}\right] &\geq \mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t) | \mathcal{F}_{t-1}, \mathbf{x}_t \in \mathcal{X} \setminus S_t\right] \mathbb{P}\left(\mathbf{x}_t \in \mathcal{X} \setminus S_t | \mathcal{F}_{t-1}\right) \\ &\geq \sigma_{t-1}(\overline{\mathbf{x}}_t) P_t, \end{aligned} \quad (12)$$

in which the last inequality follows from the definition of $\overline{\mathbf{x}}_t$ and Lemma 5.

Next, conditioned on both events $E^f(t)$ and $E^{f_t}(t)$, we have that

$$\begin{aligned} r_t = \Delta(\mathbf{x}_t) &= f(\mathbf{x}^*) - f(\overline{\mathbf{x}}_t) + f(\overline{\mathbf{x}}_t) - f(\mathbf{x}_t) \\ &\overset{(a)}{\leq} \Delta(\overline{\mathbf{x}}_t) + f_t(\overline{\mathbf{x}}_t) + c_t\sigma_{t-1}(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t) + c_t\sigma_{t-1}(\mathbf{x}_t) \\ &\overset{(b)}{\leq} c_t\sigma_{t-1}(\overline{\mathbf{x}}_t) + c_t\sigma_{t-1}(\overline{\mathbf{x}}_t) + c_t\sigma_{t-1}(\mathbf{x}_t) + f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t) \\ &= c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t) + \sigma_{t-1}(\mathbf{x}_t)) + \underline{f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)}, \end{aligned} \quad (13)$$

in which (a) follows from the definition of $\Delta(\mathbf{x})$ and equation (5), 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 $\mathcal{F}_{t-1}$:

$$
\begin{aligned}
&\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right] \\
&\overset{(a)}{=} \mathbb{P}\left(A_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, A_t\right] + \mathbb{P}\left(B_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t\right] \\
&\overset{(b)}{\leq} \mathbb{P}\left(B_t\right)\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t\right] \\
&\overset{(c)}{\leq} \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{P}\left[\mathbf{x}_t \in \mathcal{X}_i\right]\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right] \\
&\leq \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t) - f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right] \\
&\overset{(d)}{\leq} \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{E}\left[\sum_{n=1}^{N}\varphi_n^{(i)}\widetilde{f}_t^n(\overline{\mathbf{x}}_t) + \Delta_t^{(i)} - \sum_{n=1}^{N}\varphi_n^{(i)}\widetilde{f}_t^n(\mathbf{x}_t) + \Delta_t^{(i)}\middle|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right] \\
&\overset{(e)}{\leq} \mathbb{P}\left(B_t\right)\sum_{i=1}^{P}\mathbb{E}\left[\left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right]\underline{\sum_{n=1}^{N}\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}} + 2\Delta_t^{(i)}\middle|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\right],
\end{aligned}
\tag{14}
$$

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 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:

$$
\begin{aligned}
\sum_{n=1}^{N}\varphi_n^{(i)}\boldsymbol{\omega}_{n,t} &= \frac{\sum_{n=1}^{N} q\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} = \frac{\sum_{n=1}^{N}\mathbb{E}_{\mathbb{I}_n}[\mathbb{I}_n]\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} = \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q}\right] \\
&= \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} - \boldsymbol{\eta} + \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} + \boldsymbol{\eta}\right] \\
&= \mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\boldsymbol{\omega}_{n,t}}{q} - \frac{\sum_{n=1}^{N}\mathbb{I}_n\varphi_n^{(i)}\widehat{\boldsymbol{\omega}}_{n,t}}{q} - \boldsymbol{\eta} + \boldsymbol{\omega}_t^{(i)}\right],
\end{aligned}
\tag{15}
$$

where in the last equality we have made use of the definition of $\boldsymbol{\omega}_t^{(i)}$ (4). Next, we plug (15) back into (14):

$$
\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)}\Bigg|\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} + \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} + \\
&\qquad\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] \underbrace{- \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right]\boldsymbol{\eta}}_{A_6} + 2\Delta_t^{(i)}\Bigg|\mathcal{F}_{t-1}, B_t, \mathbf{x}_t \in \mathcal{X}_i\Bigg]
\end{aligned}
\tag{16}
$$

Next, we separately upper-bound the terms A_1 to A_6. Firstly, we bound the term A_1. Define $\mathcal{C}_t \triangleq \{n \in [N] \Big| \left\|\boldsymbol{\omega}_{n,t}\right\|_2 > S/\sqrt{P}\}$, which is the same as the definition in Theorem 1. 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{17}
$$

in which (a) follows since $\mathbb{E}_n[\mathbb{I}_n] = q, \forall n \in [N]$; (b) follows since for those agents $n \notin \mathcal{C}_t$, $\boldsymbol{\omega}_{n,t} - \widehat{\boldsymbol{\omega}}_{n,t} = \mathbf{0}$ because the vector $\boldsymbol{\omega}_{n,t}$ is not clipped; (c) results from Lemma 3 and that $|f^n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}, n \in [N]$ (this is because of our assumption that $\|f^n\|_k \leq B, \forall n \in [N]$, Sec. 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 $\tilde{\Delta}_t^{(i)}$ (6).

Subsequently, we upper-bound the terms A_2 and A_5. For any $\mathbf{x} \in \mathcal{X}$, we have that

$$
\begin{aligned}
&\left|\boldsymbol{\phi}(\mathbf{x})^\top \mathbb{E}_{\mathbb{I}_{1:n}}[\boldsymbol{\omega}_t^{(i)}] - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i)}\right| = \left|\boldsymbol{\phi}(\mathbf{x})^\top \left(\mathbb{E}_{\mathbb{I}_{1:n}}\left[\frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right] - \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right)\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^\top \left(\frac{\sum_{n=1}^N q \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q} - \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right)\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^\top \frac{1}{q} \sum_{n=1}^N (q - \mathbb{I}_n) \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}\right| \leq \frac{1}{q} \sum_{n=1}^N \left|(q - \mathbb{I}_n) \varphi_n^{(i)} \boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| \\
&\overset{(a)}{\leq} \frac{1}{q} \sum_{n=1}^N \varphi_n^{(i)} \left|\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| \overset{(b)}{\leq} \frac{1}{q} \sum_{n=1}^N \varphi_n^{(i)} \left|\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}\right| \\
&= \frac{1}{q} \sum_{n=1}^N \varphi_n^{(i)} \left|\widetilde{f}_t^n(\mathbf{x})\right| = \frac{1}{q} \sum_{n=1}^N \varphi_n^{(i)} \left|\widetilde{f}_t^n(\mathbf{x}) - f^n(\mathbf{x}) + f^n(\mathbf{x})\right| \\
&\leq \frac{1}{q} \sum_{n=1}^N \varphi_n^{(i)} \left(|\widetilde{f}_t^n(\mathbf{x}) - f^n(\mathbf{x})| + |f^n(\mathbf{x})|\right) \\
&\overset{(c)}{\leq} \frac{1}{q} \sum_{n=1}^N \varphi_n^{(i)} \left(\tilde{\Delta}_{n,t} + B\right) \\
&\overset{(d)}{=} \frac{1}{q} \left(\tilde{\Delta}_t^{(i)} + B\right),
\end{aligned}
\tag{18}
$$

in which (a) follows since $|q - \mathbb{I}_n| \leq 1$; (b) results from (3); (c) results from Lemma 3 and that $|f^n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}, n \in [N]$; (d) results from the definition of $\tilde{\Delta}_t^{(i)}$ (6).

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 $\mathbf{x} \in \mathcal{X}$,

$$
\begin{aligned}
\left|\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(i)} - \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_t^{(\bar{i})}\right| &= \left|\boldsymbol{\phi}(\mathbf{x})^\top \left(\boldsymbol{\omega}_t^{(i)} - \boldsymbol{\omega}_t^{(\bar{i})}\right)\right| \\
&= \left|\boldsymbol{\phi}(\mathbf{x})^\top \left(\frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(i)} \widehat{\boldsymbol{\omega}}_{n,t}}{q} - \frac{\sum_{n=1}^N \mathbb{I}_n \varphi_n^{(\bar{i})} \widehat{\boldsymbol{\omega}}_{n,t}}{q}\right)\right| \\
&\leq \frac{1}{q} \sum_{n=1}^N \mathbb{I}_n \left|\varphi_n^{(i)} - \varphi_n^{(\bar{i})}\right| \left|\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| \\
&\leq \frac{1}{q} \sum_{n=1}^N \varphi_{\max} \left|\boldsymbol{\phi}(\mathbf{x})^\top \widehat{\boldsymbol{\omega}}_{n,t}\right| \\
&\overset{(a)}{\leq} \frac{1}{q} \sum_{n=1}^N \varphi_{\max} \left|\boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\omega}_{n,t}\right| \\
&\overset{(b)}{\leq} \frac{1}{q} \sum_{n=1}^N \varphi_{\max} \left(\tilde{\Delta}_{n,t} + B\right) \\
&\overset{(c)}{\leq} \frac{\varphi_{\max}}{q} \left(\Delta_t + NB\right),
\end{aligned}
\tag{19}
$$

(a) follows because of (3); (b) results from Lemma 3 and that $|f^n(\mathbf{x})| \leq B, \forall \mathbf{x} \in \mathcal{X}, n \in [N]$; (c) follows from (9).

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, be-

cause $\overline{\mathbf{x}}_t \in \mathcal{X}_{\bar{i}}$ 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:

$$\begin{aligned}\left| \left[\boldsymbol{\phi}(\overline{\mathbf{x}}_t)^\top - \boldsymbol{\phi}(\mathbf{x}_t)^\top\right] \boldsymbol{\eta} \right| &\leq \left\|\boldsymbol{\phi}(\overline{\mathbf{x}}_t) - \boldsymbol{\phi}(\mathbf{x}_t)\right\|_2 \|\boldsymbol{\eta}\|_2 \\ &\leq \left(\left\|\boldsymbol{\phi}(\overline{\mathbf{x}}_t)\right\|_2 + \left\|\boldsymbol{\phi}(\mathbf{x}_t)\right\|_2\right)\|\boldsymbol{\eta}\|_2 \overset{(a)}{\leq} 2\|\boldsymbol{\eta}\|_2 \\ &\overset{(b)}{\leq} \frac{2zS\varphi_{\max}}{q}\sqrt{2M\log\frac{8M}{\delta}},\end{aligned} \tag{20}$$

where (a) follows since the random features have been constructed such that $\left\|\boldsymbol{\phi}(\mathbf{x})\right\|_2^2 = \sigma_0^2 \leq 1$ [12], 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 (17), (18), (19), (20)), 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 (16):

$$\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{21}$$

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 4 (14) which holds with probability $\geq 1 - \delta/2$, and Gaussian concentration inequality (20) which holds with probability $\geq 1 - \delta/8$, equation (21) holds with probability $\geq 1 - \delta/2 - \delta/8 = 1 - 5\delta/8$.

Finally, we plug (21) back into (13):

$$\begin{aligned}
&\mathbb{E}\left[r_t|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))+f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+2B\mathbb{P}\left[\overline{E^{f_t}(t)}|\mathcal{F}_{t-1}\right] \\
&\leq \mathbb{E}\left[c_t(2\sigma_{t-1}(\overline{\mathbf{x}}_t)+\sigma_{t-1}(\mathbf{x}_t))|\mathcal{F}_{t-1}\right]+\mathbb{E}\left[f_t(\overline{\mathbf{x}}_t)-f_t(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+2B\mathbb{P}\left[\overline{E^{f_t}(t)}|\mathcal{F}_{t-1}\right] \\
&\overset{(a)}{\leq} \frac{2c_t}{P_t}\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+c_t\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2} \\
&\leq c_t\left(1+\frac{2}{P_t}\right)\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2} \\
&\overset{(b)}{\leq} c_t\left(1+\frac{10}{pp_1}\right)\mathbb{E}\left[\sigma_{t-1}(\mathbf{x}_t)|\mathcal{F}_{t-1}\right]+4B\mathbb{E}\left[\vartheta_t|\mathcal{F}_{t-1}\right]+\psi_t+\frac{2B}{t^2},
\end{aligned} \tag{22}$$

in which (a) follows from (12) and (21), and (b) follows since:

$$\frac{2}{P_t}=\frac{2}{p_t(p-\frac{1}{t^2})}\leq\frac{10}{pp_t}\leq\frac{10}{pp_1}, \tag{23}$$

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 (22) makes use of (21), therefore, (22), as well as Lemma 6, also holds with probability of $\geq 1-5\delta/8$.

□

Lemma 7. *Given $\delta\in(0,1)$, then with probability of at least $1-\delta$,*

$$\begin{aligned}
R_T \leq & c_T\left(1+\frac{10}{pp_1}\right)\mathcal{O}(\sqrt{T\gamma_T})+\sum_{t=1}^{T}\psi_t+\frac{B\pi^2}{3}+4B\sum_{t=1}^{T}\vartheta_t+ \\
&\left[c_T\left(1+\frac{4B}{p}+\frac{10}{pp_1}\right)+\psi_1+4B\right]\sqrt{2T\log\frac{8}{\delta}},
\end{aligned}$$

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 [12], and is hence omitted. A difference from Lemma 11 of [12] is that an error probability of $\delta/8$ has been used in the Azuma-Hoeffding Inequality in the proof here. □

Finally, we are ready to prove Theorem 1. Recall that $c_t=\mathcal{O}\left(\left(B+\sqrt{\gamma_t+\log(1/\delta)}\right)\sqrt{\log t}\right)$. Therefore,

$$\begin{aligned}
R_T &= \mathcal{O}\Bigg(\frac{1}{p_1}\left(B+\sqrt{\gamma_T+\log\frac{1}{\delta}}\right)\sqrt{\log T}\sqrt{T\gamma_T}+\sum_{t=1}^{T}\psi_t+B\sum_{t=1}^{T}\vartheta_t+ \\
&\qquad \left(B+\frac{1}{p_1}\right)\left(B+\sqrt{\gamma_T+\log\frac{1}{\delta}}\right)\sqrt{\log T}\sqrt{T\log\frac{1}{\delta}}\Bigg) \\
&= \mathcal{O}\left(\left(B+\frac{1}{p_1}\right)\sqrt{T\log T\gamma_T\log\frac{1}{\delta}\left(\gamma_T+\log\frac{1}{\delta}\right)}+\sum_{t=1}^{T}\psi_t+B\sum_{t=1}^{T}\vartheta_t\right) \\
&= \tilde{\mathcal{O}}\left(\left(B+\frac{1}{p_1}\right)\gamma_T\sqrt{T}+\sum_{t=1}^{T}\psi_t+B\sum_{t=1}^{T}\vartheta_t\right),
\end{aligned} \tag{24}$$

which finally completes the proof.

B Experiments

As we have mentioned in the main text (last paragraph of Sec. 3.2), we choose the weights for sub-region i to be $\varphi_n^{(i)} = \frac{\exp((a\mathbb{I}_n^{(i)}+1)/\mathcal{T})}{\sum_{n=1}^{N}\exp((a\mathbb{I}_n^{(i)}+1)/\mathcal{T})}$, where $\mathbb{I}_n^{(i)}$ is an indicator variable that equals 1 if agent n is assigned to explore $\mathcal{X}_i$ and equals 0 otherwise. We set $a = 15$ in all our experiments, and gradually increase the temperature $\mathcal{T}$ from 1 to $+\infty$. Specifically, for the synthetic experiments, we choose the temperature $\mathcal{T}$ as $\mathcal{T}_t = a/(a_t - 1), \forall t \geq 1$; we set $a_t = a + 1 = 16$ for the first 5 iterations ($t \leq 5$), decay the value of a_t linearly to 1 in the next 5 iterations (i.e., $a_t = 16, 12.25, 8.5, 4.75, 1$ for $t = 6, \ldots, 10$), and fix $a_t = 1, \forall t > 10$. Note that when $a_t = 1$, $\mathcal{T}_t = +\infty$ (i.e., after 10 iterations) and the distribution becomes uniform among all agents. Similarly, for all real-world experiments, we use the same softmax weighting scheme except that we fix $a_t = a + 1 = 16$ for the first 10 iterations, decay the value of a_t linearly to 1 in the next 30 iterations, and fix $a_t = 1$ afterwards. That is, the distribution becomes uniform among all agents after 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.

B.1 Synthetic Experiments

B.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 $\mathcal{A}_n, \forall n \in [N]$, 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, 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 $\sigma^2 = 0.01$ (Sec. 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.

B.1.2 More Experimental Results

Comparison between DP-FTS-DE and DP-FTS. We have shown in the main text (Fig. 2a) that FTS-DE significantly outperforms FTS without DE. Here, we demonstrate in Fig. 4 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 (Sec. 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.

Investigation of DE. We also investigate the importance of both of the major components of the DE technique (Sec. 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. 5, 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 Sec. 4, 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. 6. 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

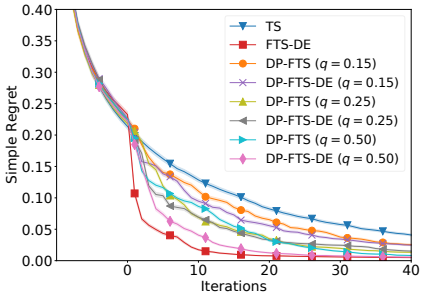


Figure 4: Comparisons of the performances of DP-FTS (without DE) and DP-FTS-DE. For a fair comparison, we have used $S = 8$ and $S = 11$ for DP-FTS and DP-FTS-DE respective, such that a similarly small percentage vectors are clipped for both algorithms (0.31% for DP-FTS and 0.80% for DP-FTS-DE). We have used $z = 1.0$ for both algorithms.

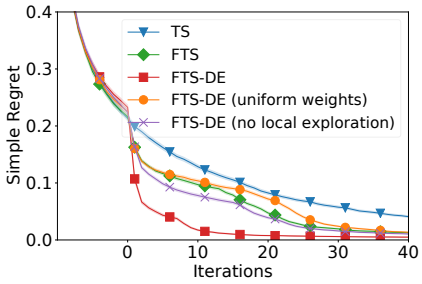


Figure 5: Investigating the importance of both major components of the technique of distributed exploration (DE). The orange curve is obtained by giving equal weights to all agent for every sub-region, and the purple curve is derived by letting every agent explore the entire domain at initialization instead of a local sub-region.

may also result in a worse performance (yellow curve) since it causes the vectors from more agents to be clipped (Sec. 4). These observations verify our discussions in the last paragraph of Sec. 4.

Robustness Against Heterogeneous Agents. We use another experiment to explore the robustness of our algorithm against agent heterogeneity, i.e., how our algorithm performs when the objective functions of different agents are significantly different. To begin with, we sample a function f_{base} from a GP (the detailed setups such as the domain and the SE kernel are the same as those used in App. B.1.1). Next, for every agent $i = 1, \ldots, 50$, we independently sample another function f^i and then use $f^i \leftarrow \alpha f^i + (1 - \alpha) f_{\text{base}}$ as the objective function for agent i in which $\alpha \in [0, 1]$. As a result, the parameter α controls the difference between the objective functions f^i's of different

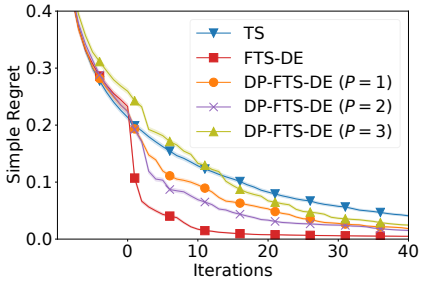
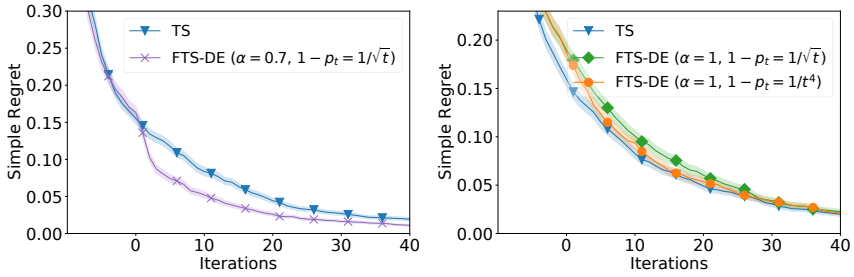


Figure 6: Trade-off induced by P regarding the practical performance of our DP-FTS-DE algorithm. Note that a larger P reduces the size of every local sub-region and hence leads to a better modeling by the GP surrogates, yet also negatively impacts the performance by causing more vectors to be clipped. Here we have used $q = 0.25$, $z = 1.0$, $S = 11.0$ for all values of P.



(a) (b)

Figure 7: Results when the agents are heterogeneous, i.e., when the objective functions of different agents are significantly different. (a) and (b) correspond to $\alpha = 0.7$ and $\alpha = 1.0$, respectively.

agents such that a larger α means that the f^i's are more different. Fig. 7a shows the performances of standard TS and our FTS-DE when $\alpha = 0.7$, in which FTS-DE is still able to outperform TS although the performance improvement is significantly smaller than that observed in Fig. 2. Fig. 7b plots the results when the objective functions f^i's are extremely heterogeneous, i.e., when $\alpha = 1.0$ which implies that the f^i's are *independent*. The figure shows that in this adverse scenario, when $1 - p_t = 1/\sqrt{t}$, FTS-DE (green curve) performs worse than standard TS (blue curve) due to the extremely high degree of heterogeneity among the agents. However, as shown by the orange curve, we can improve the performance of FTS-DE to make it comparable with standard TS by letting $1 - p_t$ decrease faster such that the impact of the other agents are diminished faster.

B.2 Real-world Experiments

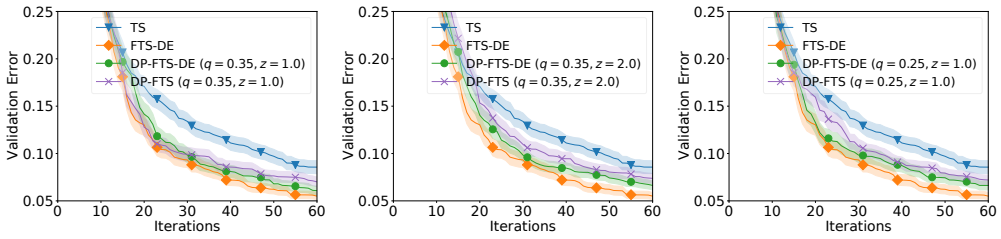
B.2.1 More Experimental Details

In all real-world experiments, when generating the random features for the RFFs 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 [12] and [53] 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 *landmine detection* dataset[15] used in this experiment has also been used by the works of [12, 60] which focus on FBO and FL respectively. This dataset consists of the landmine detection data from $N = 29$ landmine fields (agents), and the task of every agent is to use a support vector machine (SVM) to detect (classify) whether a location in its landmine field contains landmines or not (i.e., binary classification). We tune two hyperparameters of SVM, i.e., the RBF kernel parameter ($[0.01, 10.0]$) and the penalty parameter ($[10^{-4}, 10.0]$). For every landmine field, we use half of its dataset as the training set and the remaining half as the validation set. In this experiment, we report the area under the receiver operating curve (AUC) as the performance metric, instead of validation error, because this dataset is significantly imbalanced, i.e., the vast majority of the locations do not contain landmines. No data is excluded. Refer to [66] for more details on this dataset. The dataset is publicly available, and contains no personally identifiable information or offensive content.

[15] `http://www.ee.duke.edu/~lcarin/LandmineData.zip`.



(a) (b) (c)

Figure 8: 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.

The *human activity recognition* dataset[16] was originally introduced by the work of [2] and has also been adopted by the works of [12, 60]. The licensing requirement for this dataset requires that the use of this dataset in publications must be acknowledged by referencing the paper of [2]. This dataset consists of the data collected using mobile phone sensors when $N = 30$ subjects (agents) are performing six different activities. The task of every agent (subject) is to use the dataset generated by the subject to perform activity recognition, i.e., to predict which one of the six activities the subject is performing using logistic regression (LR). We tune three hyperparameters of LR: the batch size ($[128, 512]$), L2 regularization parameter ($[10^{-6}, 10]$) and learning rate ($[10^{-6}, 1]$). For every subject, we again use half of its data as the training set and the other half as the validation set, and the validation error is reported as the performance metric. The inputs for every agent are standardized by removing the mean and dividing by the standard deviation of its training set, which is a common pre-processing step for LR. No data is excluded. Refer to [2] for more details on this dataset. The dataset is publicly available as described above, and does not contain personally identifiable information or offensive content.

EMNIST[17] is a dataset of images of handwritten characters from different persons, and is a widely used benchmark in FL [29]. The EMNIST dataset is under the CC0 License. Here we use the images from the first $N = 50$ subjects (agents) which can be accessed from the TensorFlow Federated library[18]. 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 [8] for more details on this dataset. The dataset is publicly available, and contains no personally identifiable information or offensive content.

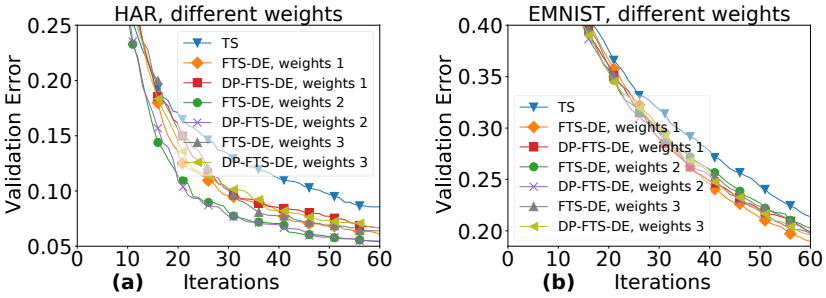
B.2.2 Comparison between DP-FTS-DE and DP-FTS

We have shown in the main text (Figs. 3a,b,c) that FTS-DE significantly outperforms FTS without DE. Here we further verify in Fig. 8 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, 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. 4, 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.

[16] `https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones`.

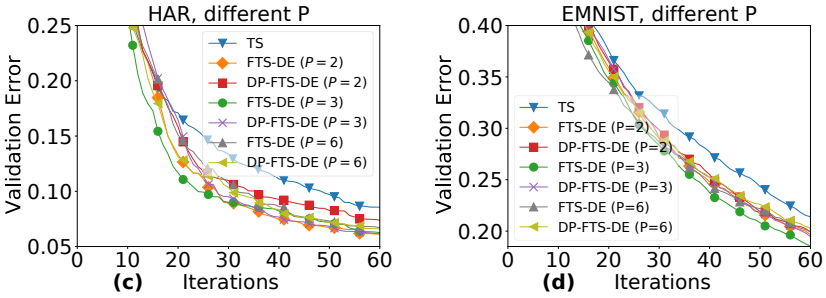
[17] `https://www.nist.gov/itl/products-and-services/emnist-dataset`.

[18] `https://www.tensorflow.org/federated`.



(a) (b)

Figure 9: Robustness of our FTS-DE and DP-FTS-DE algorithms against the choice of weights.



(a) (b)

Figure 10: Robustness of our FTS-DE and DP-FTS-DE algorithms against the number P of sub-regions.

B.2.3 Robustness against the Choice of Weights and Number of Sub-regions

In this section, we evaluate the robustness of our experimental results against the choice of the weights assigned to different agents and the number P of sub-regions, using the human activity recognition and EMNIST experiments.

Here we test three other methods for designing the weights: (1) We use the same softmax weighting scheme with a different parameter $a = 9$ instead of $a = 15$ ("weights 1" in Fig. 9). (2) For a sub-region $\mathcal{X}_i$, we assign a weight $\propto a$ to those agents exploring $\mathcal{X}_i$ and $\propto b$ to the other agents, and similarly gradually decay the value of $a = 1,000$ to $b = 1$ ("weights 2" in Fig. 9). (3) For a sub-region $\mathcal{X}_i$, we assign a weight $\propto a^2$ to those agents exploring $\mathcal{X}_i$ and $\propto b$ to the other agents, and similarly gradually decay the value of $a = 40$ to $b = 1$ ("weights 3" in Fig. 9). Moreover, in addition to the results using $P = 4$ sub-regions reported in the main text, here we also evaluate the performance of FTS-DE and DP-FTS-DE with $P = 2, 3, 6$ sub-regions (Fig. 10). All DP-FTS-DE methods in Fig. 9 and Fig. 10 correspond to $q = 0.35, z = 1.0$. The results demonstrate the robustness of our experimental results against the choice of the weights and the number P of sub-regions.

B.2.4 Rényi DP

Fig. 11 shows the privacy-utility trade-off in the landmine detection experiment using Rényi DP [63]. The results demonstrate that Rényi DP, despite requiring modifications to our theoretical analysis (i.e., proof of Theorem 1), leads to slightly better privacy losses (compared with Fig. 3a) with comparable utilities.

B.2.5 Adaptive Weights vs. Non-adaptive Weights

As we have discussed in the last paragraph of Sec. 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 $\{\varphi_n^{(i)}, \forall n \in [N]\}$ for all $t \in [T]$. In particular, we

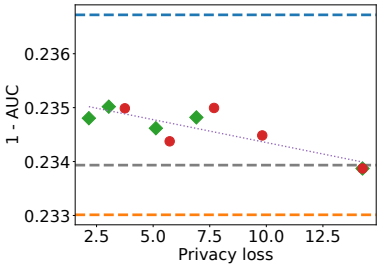


Figure 11: Results for the landmine detection experiment using Rényi DP [63].

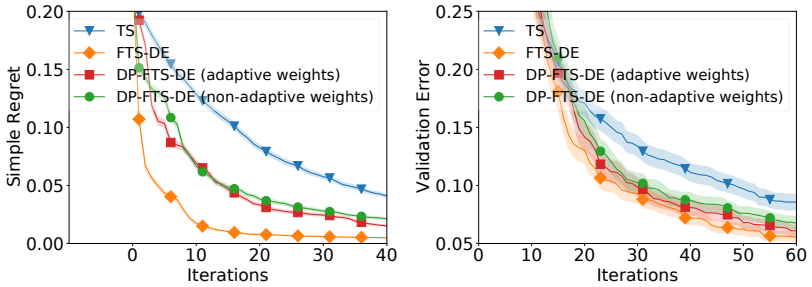


Figure 12: Comparison between DP-FTS-DE with adaptive weights and non-adaptive weights, using (a) the synthetic experiment and (b) human activity recognition experiment.

adopt the same softmax weighting scheme as described in the first paragraph of App. B, but fix the temperature $\mathcal{T}_t = 1, \forall t \geq 1$ such that the same set of weights is used for all $t \geq 1$. That is, for every sub-region $\mathcal{X}_i$, we assign more weights to those agents exploring $\mathcal{X}_i$ throughout all iterations $t \geq 1$.

Fig. 12 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).

B.2.6 Computational Cost

When maximizing the acquisition function to select the next query (lines 6 and 8 of Algo. 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 Algo. 1). For agents, the incorporation of DP brings no additional computational cost to them. Meanwhile, the addition of DE, which affects line 8 of Algo. 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.