Lifelong Bandit Optimization: No Prior and No Regret

Felix Schur[*1] **Parnian Kassraie**[*1] **Jonas Rothfuss**[1] **Andreas Krause**[1]

[1]ETH Zurich, Switzerland

Abstract

Machine learning algorithms are often repeatedly applied to problems with similar structure over and over again. We focus on solving a sequence of bandit optimization tasks and develop LIBO, an algorithm which *adapts* to the environment by learning from past experience and becomes more sample-efficient in the process. We assume a kernelized structure where the kernel is *unknown* but *shared* across all tasks. LIBO sequentially meta-learns a kernel that approximates the true kernel and solves the incoming tasks with the latest kernel estimate. Our algorithm can be paired with *any* kernelized or linear bandit algorithm and guarantees *oracle optimal* performance, meaning that as more tasks are solved, the regret of LIBO on each task converges to the regret of the bandit algorithm with oracle knowledge of the true kernel. Naturally, if paired with a sublinear bandit algorithm, LIBO yields a sublinear lifelong regret. We also show that direct access to the data from each task is not necessary for attaining sublinear regret. We propose F-LIBO, which solves the lifelong problem in a federated manner.

1 INTRODUCTION

A key aspect of human intelligence is our ability to harness previous experience and quickly improve when repeatedly solving similar problems. In this paper, we study how to solve a *sequence of learning problems, on related instances*, and become more efficient in the process. In particular we focus on problems which are solved through Bayesian Optimization, a.k.a. kernelized bandit algorithms (BO), where the kernel captures regularity structure of the tasks. A motivating application are AutoML systems, which perform hyper-parameter tuning for the same model on different datasets, or different models on the same dataset. We expect that the more tasks our machine learning system solves, the better the system becomes at solving the next one.

We model this as *lifelong learning*, where an agent sequentially faces kernelized bandit problems with different unknown reward functions. While prior work assumes the kernel to be known (e.g., hand-designed), we consider the kernel k^* to be *unknown*, but *shared* between the problem instances. After each bandit task, we use the previously collected data to *meta-learn a kernel function* $\hat{k}$ as a proxy for the unknown $k^\star$. We transfer knowledge across tasks by sequentially updating the meta-learned kernel and using it to solve the next task. This way, we adapt to the environment and gradually improve the bandit performance. Ideally, we would like to reach the oracle-optimal performance, i.e. the performance of a bandit algorithm with complete knowledge of the environment.

Lifelong bandit optimization is a delicate problem for two reasons. First, the success of each round of BO depends on the validity of the meta-learned kernel: We only have guaranteed convergence and sublinear regret if the reproducing kernel Hilbert space (RKHS), induced by the estimated kernel $\hat{k}$, contains the reward functions. Second, the data that is used for meta-learning is collected at the previous BO tasks. Thus, during each BO round, we not only have to quickly find reward maximizing actions, but also have to collect exploratory data that is sufficiently informative for successful meta-learning of the kernel.

We address these challenges when the true kernel is a sparse convex combination of a large number of candidate kernels. We propose an approach for meta-learning a provably consistent estimator of the true kernel, given data from previous tasks (Theorem 3.3). To ensure that this data is sufficiently informative, we interlace the queries of the BO agent with purely exploratory queries. Combining these two key ideas, we design our main algorithm, the *Lifelong Bandit Optimizer (*LIBO*)*. This algorithm is

[*]Equal contribution.

Proceedings of the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023), PMLR 216:1847–1857.

versatile since it is *agnostic to the bandit policy*, i.e. it can be wrapped around *any* kernelized or linear base bandit algorithm to influence its policy and satisfy lifelong guarantees. We prove that it is *oracle-optimal*, i.e. that by using LIBO, we can eventually achieve the *same* worst-case performance as the base bandit algorithm which has oracle knowledge of the true kernel (Theorem 4.1). We *do not make assumptions* about the base bandit algorithm, and our convergence guarantees hold for many bandit solvers such as OFUL [Abbasi-Yadkori et al., 2011], GP-UCB [Srinivas et al., 2010] or GP-TS [Chowdhury and Gopalan, 2017]. Additionally, we consider a federated setting where each BO task is performed by a client node in a network and the data ought not to be exchanged with the server node due to privacy concerns. We propose the *Federated Lifelong Bandit Optimizer (*F-LIBO*)*, and show that it satisfies a guarantee similar to LIBO (Theorem 5.1). If we take GP-UCB as base bandit solver, LIBO and F-LIBO have the *same worst-case regret bound rates* as the GP-UCB solver when given *oracle knowledge* of the true kernel (Corollary 4.2 and 5.2). In Section 6 we support our theoretical findings by experiments on synthetic and real-world data in the AutoML context. Lastly, we discuss related works in Section 7.

2 PROBLEM STATEMENT

We consider a lifelong optimization setting, where an agent interacts with a sequence of black-box optimization problems, arriving one after another. Throughout the sequence of optimization tasks, the agent can adapt to the environment based on the previously collected data and improve its performance on the succeeding tasks. Formally, the agent iteratively faces bandit problems with unknown reward functions $f_1, ..., f_m$ residing in a RKHS $\mathcal{H}_{k^\star}$ that corresponds to an *unknown* kernel function $k^\star$. To impose regularity, we assume that the reward function has a bounded kernel norm $\|f\|_{k^\star} \leq B$ and that the domain $\mathcal{X} \subset \mathbb{R}^{d_0}$ is compact. The agent interacts with each task f_s for n time steps. For each task $s = 1, \dots, m$, at time step $i = 1, \dots, n$, the agent selects an action $\boldsymbol{x}_{s,i} \in \mathcal{X}$ and receives a stochastic reward via $y_{s,i} = f_s(\boldsymbol{x}_{s,i}) + \varepsilon_{s,i}$. Here, $\varepsilon_{s,i}$ are i.i.d. samples from a zero-mean sub-Gaussian noise with variance proxy σ^2. The goal of the agent is to maximize its rewards across all tasks. This can be formalized as minimizing the *lifelong regret* over m tasks of size n, defined as

$$R(m,n) \coloneqq \sum_{s=1}^{m} \sum_{i=1}^{n} f_s(\boldsymbol{x}_s^\star) - f_s(\boldsymbol{x}_{s,i}),$$

where $\boldsymbol{x}_s^\star$ is a global maximum of f_s. If $\frac{R(m,n)}{mn} \to 0$ as $m, n \to \infty$ then the agent eventually converges to the global optimum of each upcoming optimization task. This property is commonly referred to as *sublinearity* of the regret. To attain a small regret, the agent maintains an estimate of the unknown reward function f_s based on its history. Typically, a kernelized regression oracle (e.g. kernel ridge regression or Gaussian Processes) is employed for this task. The choice of the kernel function plays a key role in the success and data-efficiency of the bandit optimization. If the hypothesis space $\mathcal{H}_k$ induced by the kernel k is too restrictive and does not contain the true reward functions f_s, the agent will likely never find reward maximizing actions. To prevent this, most practitioners pick a kernel with a conservatively complex kernel with a large hypothesis space that is very likely to contain $\mathcal{H}_{k^\star}$. However, the larger $\mathcal{H}_k$, the more observations it takes to form a good reward estimate, making the finding an optimal solution less efficient.

We take a data-driven approach to select the kernel. In particular, we aim to sequentially meta-learn a kernel $\hat{k}$ which approximates the true kernel $k^\star$, using the data from previous bandit tasks. Let $\mathcal{D}_s \coloneqq \{(\boldsymbol{x}_{s,i}, y_{s,i})_{i \leq n}\}$ be the data corresponding to task s, and $\mathcal{D}_{1:s} \coloneqq \mathcal{D}_1 \cup \dots \cup \mathcal{D}_s$ be the collection of datasets from the first s tasks. Then once the agent solves task s, we pass $\mathcal{D}_{1:s}$ to the *meta-agent*, who meta-learns a kernel $\hat{k}_s$. This kernel is then provided to the agent who uses it for solving the next BO task. Our meta-learning algorithm can be paired with any kernelized bandit algorithm and achieves sublinear lifelong regret, if the bandit algorithm achieves sublinear single-task regret given oracle knowledge of the kernel.

3 META-LEARNING KERNELS

We first present *Meta Kernelized Group Lasso (*META-KGL*)*, our approach to estimate the kernel $k^\star$, given data from previous tasks. We consider a large set of eligible known base kernels $\{k_1, \dots, k_p\}$ where $k_j : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for all $j = 1, \dots, p$ and $k_j(\boldsymbol{x}, \boldsymbol{x}') \leq 1$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ without loss of generality. We assume that while $k^\star$ is unknown, it is a sparse linear combination of kernels selected from this set, i.e., there exists $J^\star \subset \{1, \dots, p\}$ and $\alpha_1, \dots \alpha_p \in \mathbb{R}$ such that

$$k^\star(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j \in J^\star} \alpha_j k_j(\boldsymbol{x}, \boldsymbol{x}'),$$

where $|J^\star| \ll p$. The set $\{k_1, \dots, k_p\}$ can be very large, since we prove that the cost of finding $J^\star$ depends only logarithmically on p. We further assume that each k_j corresponds to a d_j-dimensional feature map, i.e., $k_j(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}_j(\boldsymbol{x})^T \boldsymbol{\phi}_j(\boldsymbol{x}')$, where $\boldsymbol{\phi}_j \in \mathbb{R}^{d_j}$ and $d_j < \infty$. This setting generalizes the common linear bandit assumption, to also account for higher-order terms and interaction between coordinates of the input. Let $\boldsymbol{\phi}(\boldsymbol{x})$ denote the concatenated d-dimensional feature map, where $d \coloneqq \sum_{j=1}^{p} d_j$ and $\boldsymbol{\phi}(\boldsymbol{x}) \coloneqq [\boldsymbol{\phi}_j^T(\boldsymbol{x})]_{j \leq p}$. Then for $s = 1, \dots, m$ the reward functions can be written as $f_s(\cdot) = \sum_{j=1}^{p} \boldsymbol{\phi}_j^\top(\cdot) \boldsymbol{\beta}_s^{\star(j)}$ such that $\boldsymbol{\beta}_s^{\star(j)} = 0$ for all $j \notin J^\star$. Moreover, the RKHS norm of f_s will be equal to $\|f_s\|_{k^\star}^2 = \sum_{j=1}^{p} \|\boldsymbol{\beta}_s^{\star(j)}\|_2^2$. This kernel

model is inspired by Kassraie et al. [2022], who assume $k^\star$ lies in the convex cone of the base kernels.

In the lifelong setting, we sequentially form kernel estimates $\hat{k}_s$ based on $\mathcal{D}_{1:s}$ for $s = 1, \dots, m$. In this section, we consider one snapshot of this process for $s = m$, where we have fixed meta-training data $\mathcal{D}_{1:m}$ and meta-learn $\hat{k} := \hat{k}_m$. We assume without loss of generality (c.f. Appendix C),

$$k^\star(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{|J^\star|} \sum_{j \in J^\star} k_j(\boldsymbol{x}, \boldsymbol{x}'),$$

and minimize a sparsity inducing loss which allows us to discard kernels that do not appear in the above formulation. META-KGL first minimizes $\mathcal{L}(\boldsymbol{\beta}; \mathcal{D}_{1:m})$ over $\boldsymbol{\beta} \in \mathbb{R}^{md}$.

$$\begin{aligned} \mathcal{L}(\boldsymbol{\beta}; \mathcal{D}_{1:m}) :=& \frac{1}{|\mathcal{D}_{1:m}|} \|\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{j=1}^{p} \|\boldsymbol{\beta}^{(j)}\|_2 \qquad (1) \\ =& \frac{1}{mn} \sum_{s=1}^{m} \sum_{i=1}^{n} \left(y_{s,i} - \sum_{j=1}^{p} \boldsymbol{\phi}_j^T(\boldsymbol{x}_{s,i}) {\boldsymbol{\beta}_s}^{(j)}\right)^2 \\ &+ \lambda \sum_{j=1}^{p} \sqrt{\sum_{s=1}^{m} \|{\boldsymbol{\beta}_s}^{(j)}\|_2^2} \end{aligned}$$

The vectorized formulation uses the following notation

$$\begin{aligned} \boldsymbol{y} &:= \left[[y_{1,i}]_{i \le n}, \dots, [y_{m,i}]_{i \le n} \right] \in \mathbb{R}^{mn}, \\ \boldsymbol{\beta} &:= \left[[{\boldsymbol{\beta}_1}^{(j)}]_{j \le p}, \dots, [{\boldsymbol{\beta}_m}^{(j)}]_{j \le p} \right] \in \mathbb{R}^{md}, \\ \boldsymbol{\beta}^{(j)} &:= \left[{\boldsymbol{\beta}_1}^{(j)}, \dots, {\boldsymbol{\beta}_m}^{(j)} \right] \in \mathbb{R}^{md_j}, \\ \boldsymbol{\Phi} &:= \operatorname{diag}(\boldsymbol{\Phi}_1, \dots \boldsymbol{\Phi}_m) \in \mathbb{R}^{mn \times dm}. \end{aligned}$$

Here $\boldsymbol{\Phi}_s := (\boldsymbol{\phi}^\top(\boldsymbol{x}_{s,1}), \dots, \boldsymbol{\phi}^\top(\boldsymbol{x}_{s,n}))^\top$ is the $n \times d$ feature matrix of a task s, and therefore $\boldsymbol{\Phi}$ denotes a block diagonal matrix which gathers the features across all tasks. This meta-loss function is convex, and is equivalent to the well-known Group Lasso objective [Lounici et al., 2011]. Therefore, it can be efficiently optimized using Group Lasso solvers [e.g., Massias et al., 2018] and enjoys the statistical properties of the Group Lasso, e.g., consistency and variable selection. Let $\hat{\boldsymbol{\beta}} := \arg\min \mathcal{L}(\boldsymbol{\beta}; \mathcal{D}_{1:m})$. The first term in Eq. (1) represents the squared prediction error of $\hat{\boldsymbol{\beta}}$, while the second is a regularization term that induces group sparsity in $\hat{\boldsymbol{\beta}}$. Mainly, the solutions $\hat{\boldsymbol{\beta}} = (\hat{\boldsymbol{\beta}}^{(1)}, \dots, \hat{\boldsymbol{\beta}}^{(p)})$ to this problem are group sparse, i.e. $\hat{\boldsymbol{\beta}}^{(j)} = 0$ for many of the indices $j \in \{1, \dots, p\}$. META-KGL then constructs the set of plausible kernels $\hat{J}$, by thresholding $\|\hat{\boldsymbol{\beta}}^{(j)}\|_2$ and discarding the kernels that do no appear to be influencing the data, i.e.,

$$\hat{J} := \left\{ j \mid j \in \{1, \dots, p\} \text{ s.t. } \|\hat{\boldsymbol{\beta}}^{(j)}\|_2 > \omega\sqrt{m} \right\}.$$

where $\omega > 0$ is a hyperparamter of the algorithm. We then construct the estimated kernel as

$$\hat{k}(\boldsymbol{x}, \boldsymbol{x}') := \frac{1}{|\hat{J}|} \sum_{j \in \hat{J}} k(\boldsymbol{x}, \boldsymbol{x}').$$

META-KGL is summarized in Algorithm 1. Under mild assumptions on the dataset, we can show that $\hat{k}$ converges to the true kernel $k^\star$ in probability. Our first assumption ensures that if $j \in J^\star$, i.e., k_j is active in the true kernel, then the contribution of k_j to the data is large enough to be statistically detectable under noise.

Assumption 3.1 (Beta-min)**.** *There exists $c_1 > 0$ such that for all $j \in J^\star$,*

$$\|\boldsymbol{\beta}^{\star(j)}\|_2 \ge c_1\sqrt{m}.$$

This assumption is commonly used in the high-dimensional statistics literature [Bühlmann and Van De Geer, 2011, Bunea et al., 2013, Zhao and Yu, 2006]. Our second assumption requires that the meta-training data is sufficiently diverse. In Proposition 4.3, we propose a policy which provably satisfies this assumption.

Assumption 3.2 (Sufficiently Informative Data)**.** *The feature matrix $\boldsymbol{\Phi} \in \mathbb{R}^{mn \times d}$ is sufficiently informative if there exists a constant $c_\kappa > 0$ such that $\kappa(\boldsymbol{\Phi}) \ge c_\kappa$ where*

$$\begin{aligned} \kappa(\boldsymbol{\Phi}) :=& \inf_{(J, \boldsymbol{b})} \frac{1}{\sqrt{n}} \frac{\|\boldsymbol{\Phi}\boldsymbol{b}\|_2}{\sum_{j \in J} \|\boldsymbol{b}^{(j)}\|_2} \\ & \text{s.t. } \boldsymbol{b} \in \mathbb{R}^d \backslash \{0\}, \sum_{j \notin J} \|\boldsymbol{b}^{(j)}\|_2 \le 3 \sum_{j \in J} \|\boldsymbol{b}^{(j)}\|_2, \\ & J \subset \{1, \dots, p\}, \ |J| \le |J^\star|. \end{aligned}$$

Intuitively, $\kappa(\Phi)$ measures the quality of the data: data points that are almost identical decrease $\kappa(\Phi)$, and $\kappa(\Phi)$ is large when data points are diverse. If the minimum eigenvalue of $\boldsymbol{\Phi}$ is positive, Assumption 3.2 is automatically fulfilled. This type of assumption is common in the literature on representation/meta-learning for sequential decision-making [Yang et al., 2021, Cella and Pontil, 2021, Kassraie et al., 2022] and sparse linear bandits [Bastani and Bayati, 2020, Hao et al., 2020, Kim and Paik, 2019]. It is also known in the Lasso literature as the compatibility condition [Bühlmann and Van De Geer, 2011]. Given these assumptions, we show that META-KGL recovers the true kernel with high probability.

Theorem 3.3 (Consistency of META-KGL)**.** *Suppose $D_{1:m}$ satisfies Assumption 3.1 and 3.2 with constants c_1 and c_κ respectively. Set $\omega \in (0, c_1)$ and define $\bar{\omega} = \min\{\omega, c_1 - \omega\}$. Choose $\lambda = \bar{\omega}c_\kappa^2/(8\sqrt{m})$. Then for $\sqrt{n} > 32\sigma/(\bar{\omega}c_\kappa^2)$,* META-KGL *satisfies*

$$\mathbb{P}\left[\hat{J} = J^\star\right] \ge 1 - p\exp\left(-m\left(\frac{\bar{\omega}c_\kappa^2\sqrt{n}}{32\sigma} - 1\right)^2\right).$$

In particular, $\hat{J}$ is a consistent estimator both in n and m,

$$\lim_{n \to \infty} \mathbb{P}\left[\hat{J} = J^\star\right] = 1, \text{ and } \lim_{m \to \infty} \mathbb{P}\left[\hat{J} = J^\star\right] = 1.$$

Appendix D presents the proof to Theorem 3.3. This theorem shows that our meta-learned kernel converges to $k^\star$

as the number of meta-training tasks increases. First, this implies that the meta-learned hypothesis space includes the unknown reward functions allowing downstream bandit algorithms to provably converge to the optimum. Second, all candidate kernels k_j that are not active in $k^\star$ are eventually excluded from $\hat{k}$. By excluding all k_j with $j \notin J^\star$ which are not necessary for estimating $f_s \in \mathcal{H}_{k^\star}$, we effectively shrink the size of the hypothesis space, thereby reducing the uncertainty of the reward function estimates during bandit optimization. Compared to $k^{\text{full}} := \frac{1}{p}\sum_{j=1}^{p} k_j$, which naively uses all kernels, this leads to significant improvements in the query efficiency and performance of the bandit optimization.

Comparison with Prior Work. Kassraie et al. [2022] propose META-KEL, a Lasso-equivalent loss for meta-learning a sparse kernel, given i.i.d. offline data from i.i.d. tasks. We emphasize that is not possible to achieve lifelong guarantees by sequentially applying this algorithm. META-KGL differs from META-KEL in key points, and satisfies stronger consistency guarantees: 1) It converges to $k^\star$ as either n the number of samples per task, or m number of tasks grow. In contrast, META-KEL converges in m only. 2) META-KGL satisfies the exact recovery guarantee for $k^\star$ since $J^* = \hat{J}$ with high probability. While META-KEL only guarantees that $J^\star \subset \hat{J}$. This is not sufficient to show that meta-learning improves upon the trivial kernel choice k_{full}. Both of these properties are required in the lifelong analysis.

4 LIFELONG BANDIT OPTIMIZATION

We now use META-KGL as a building block to develop the *Lifelong Bandit Optimizer* (LIBO), an algorithm for lifelong bandit or Bayesian optimization. LIBO is paired with a BASEBO agent which can be instantiated by any kernelized bandit algorithm, e.g., GP-UCB [Srinivas et al., 2010] or GP-TS [Chowdhury and Gopalan, 2017]. For each task f_s, the BASEBO agent is given the kernel $\hat{k}_{s-1}$ meta-learned on the $s-1$ first tasks. Equipped with the kernel, BASEBO interacts with the current bandit environment, aiming to optimize its payoff by balancing exploration and exploitation.

In the lifelong setting, we not only have to explore for the sake of optimizing the current reward function f_s, but also we need to make sure to that the sequence of action-reward pairs will be sufficiently informative (in the sense of Assumption 3.2) for meta-learning $\hat{k}_s$ in the next stage. To this end, LIBO forces the base agent to select purely exploratory actions for the first n_s steps of the task, by i.i.d. sampling from uniform distribution on $\mathcal{X}$. Following Basu et al. [2021], we refer to this as *forced exploration* and use $\mathcal{D}_s^{\text{exp}} := \{(\boldsymbol{x}_{s,i}, y_{s,i}), i \le n_s\}$ to refer to the collected exploratory data of task f_s. We use a decreasing sequence $(n_1, \dots, n_m)$ as detailed below, since less exploration by BASEBO will be required once more multi-task data is collected. For steps $i > n_s$, BASEBO selects actions according to its normal bandit policy. After the agent has interacted with the current task for n steps, we pass the exploratory data $\mathcal{D}_{1:s}^{\text{exp}}$ to META-KGL to meta-learn $\hat{k}_s$. We then announce this new kernel estimate to the BASEBO agent for solving the next task $s+1$. Figure 1 visualizes this process and Algorithm 2 summarizes LIBO.

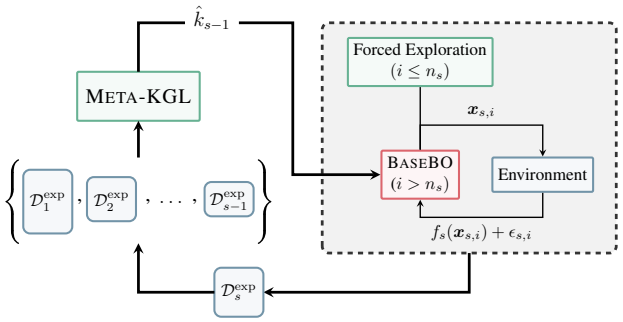

Figure 1: Overview of LIBO.

4.1 REGRET BOUNDS

Let $R^\star(n)$ be the worst-case regret of BASEBO with oracle knowledge of true kernel $k^\star$ on single tasks when the reward resides in $\mathcal{H}_{k^\star}$. When employed sequentially on m bandit tasks, the worst-case lifelong regret $R(m,n)$ will be of the order $mR^\star(n)$ with high probability. We refer to this as oracle regret, since the BASEBO has access to the true kernel k^* which does not hold in practice. Since our meta-learned kernels $\hat{k}_s$ are an approximations of k^*, the oracle regret is a natural lower bound on the regret of LIBO.

In the following, we show that if $R^\star(n)$ the single-task oracle regret of the base bandit algorithm is sublinear (e.g., as for GP-UCB or GP-TS), then so is the lifelong regret $R(m,n)$ of LIBO. Importantly, $R(m,n)$ is not only sublinear in n, but also converges with high probability to $R^\star(m,n)$. Theorem 4.1 presents this guarantee, assuming that the forced exploration datasets $\mathcal{D}_s^{\text{exp}}$ satisfy assumption Assumption 3.2 which META-KGL requires to yield a provably consistent estimator of $k^\star$. Later in Proposition 4.3, we show that exploration by i.i.d. sampling from a uniform distribution over $\mathcal{X}$ will guarantee this assumption.

Theorem 4.1. *For all tasks $s = 1, \dots, m$, assume that the reward function $f_s \in \mathcal{H}_{k^\star}$ has bounded RKHS norm $\|f_s\|_{k^\star} \le B$. Set the number of forced exploration actions as $n_s = \frac{\sqrt{n}}{s^{1/4}}$, and assume that Assumption 3.1 and 3.2 hold for the data $\mathcal{D}_{1:s}^{\text{exp}}$ for all $s = 1, \dots, m$. Suppose, with probability greater than $1-\delta/2$,* BASEBO *has worst-case oracle regret $R^\star(m,n)$. Then, the lifelong regret of* LIBO *satisfies*

$$R(m,n) - R^\star(m,n) = \mathcal{O}\Big(\underbrace{Bm^{3/4}\sqrt{n}}_{\text{forced exp.}} + \underbrace{B(nm)^{1/3}\log^{3/4}(mp/\delta)}_{\text{kernel mismatch}}\Big)$$

with probability greater than $1-\delta$.

The explicit inequality without the $\mathcal{O}$-notation can be found in Appendix E, together with the proof. In the following, we give a sketch of the proof, aiming to explain the source of each term in the bound. For every forced exploration step, in the worst-case, we suffer regret of $2B$. When accumulated over a total of $\sum_{s=1}^{m} n_s$ such steps, this gives the first term in the bound. If $\hat{k}_s \neq k^\star$, it is possible to suffer from linear regret in the worse-case. To account for this, we calculate the smallest integer m_0, for which, with high probability, $\hat{k}_s = k^\star$ for all $m_0 < s \leq m$. Based on Theorem 3.3, we show that $m_0 = \mathcal{O}((m/n^2)^{1/3} \log^{3/4}(mp/\delta))$. For every task $s \leq m_0$ we suffer a linear regret of $2Bnm_0$ in the worst-case. This is upper bounded by the second term in Theorem 4.1, which can be regarded as the cost of learning $J^\star$. Notably, it grows only logarithmically with p the number of considered features/kernels, offering a significant improvement about the polynomial rates given by prior works [e.g., Yang et al., 2021, Hong et al., 2022]. Table 1 and Table 2 present a comprehensive list of the related regret bounds.

We highlight that the excess regret of LIBO in Theorem 4.1 is sublinear in both m and n. This implies that the algorithm is *oracle optimal*, meaning that as $m \to \infty$, the single-task regret *without* knowledge of $k^\star$, eventually approaches the oracle single-task regret. Recall that $R^\star(m, n) = mR^\star(n)$ and therefore, $R(m, n)/m \to R^\star(n)$. This guarantee is stronger than that of [Basu et al., 2021, Peleg et al., 2022], where the excess regret depends linearly on m due to excessive forced exploration. By decreasing $n_s \propto s^{-1/4}$ the number of exploratory steps vanishes throughout the sequence of tasks.

As an example, we analyze the performance if GP-UCB[1] [Srinivas et al., 2010] is used as the BASEBO algorithm. In this case, we demonstrate that the worst-case lifelong regret of LIBO is of the same rate as the corresponding oracle regret. To highlight the benefit of this oracle optimality we compare to a naive baseline which uses $\hat{k}_s = k^{\text{full}} = \sum_{j=1}^{p} \frac{1}{p} k_j$ for all tasks instead of meta-learning $\hat{k}_s$ sequentially. In particular, we consider solving a sequence of m tasks in three scenarios: 1) running LIBO paired with GP-UCB 2) repeatedly running GP-UCB with oracle access to k^*, and 3) repeatedly running GP-UCB with k^{full}. The following corollary shows that the worst-case upper bound for the first two scenarios match in $\mathcal{O}$-notation. Appendix E.2 presents the proof.

Corollary 4.2 (Lifelong GP-UCB)**.** *Consider the setting of Theorem 4.1 with* GP-UCB *as* BASEBO *agent. Then, with probability at least* $1-\delta$*, the lifelong regret of* LIBO *paired with* GP-UCB *satisfies*

$$R(m,n) = \mathcal{O}\big(R^\star(m,n)\big) = \mathcal{O}\left(Bmd^\star\sqrt{n}\log\tfrac{n}{d^\star} + m\sqrt{nd^\star\log\tfrac{n}{d^\star}\log\tfrac{1}{\delta}}\right)$$

where $d^\star \coloneqq \sum_{j\in J^\star} d_j$.

In the third scenario, we conservatively set $\hat{k}_s = k^{\text{full}}$ for all $s = 1, \dots, m$. While this is sufficient for attaining a lifelong regret that is sublinear in n, the performance will *not* be oracle optimal. In particular, this algorithm suffers from a regret of

$$R(m,n) = \mathcal{O}\left(B\tfrac{mdp}{|J^\star|}\sqrt{n}\log\tfrac{n}{d} + \sqrt{nd\log\tfrac{n}{d}\log\tfrac{1}{\delta}}\right)$$

where $d = \sum_{j=1}^{p} d_j \gg d^\star$ and $p/|J^\star|$ can be very large. Our experiments confirm that the performance of the naive approach is significantly worse than the other variants. This is due to the fact that confidence bounds constructed using k^{full} tend to contract slower than the ones constructed with the sparse meta-learned $\hat{k}_s$.

4.2 FORCED EXPLORATION

Our forced exploration scheme ensures that the collected data is sufficiently informative to guarantee successful meta-learning. From a technical perspective, it ensures that Assumption 3.2 is met and allows for a consistent estimator of $k^\star$. The cost of this exploration in the regret of each task is smaller in rate than the minimax regret bound [Lattimore and Szepesvári, 2020]. Therefore it has only a negligible effect on the overall performance guarantees of LIBO (see Corollary 4.2). We show that by uniformly drawing actions from the domain, the collected data satisfies this assumption:

Proposition 4.3. *Assume that* $\phi_j \in L^2(\mathcal{X})$, $j \in \{1, \dots, p\}$ *are orthonormal and let* $d_j = 1$*. Draw* $\boldsymbol{x}_1, \dots, \boldsymbol{x}_{n_1}$ *independently and uniformly from* $\mathcal{X}$*, and repeatedly use them to construct* $\mathcal{D}_{1:s}^{\text{exp}}$*. Then with probability at least* $1-\delta$*,* $\mathcal{D}_{1:s}^{\text{exp}}$ *satisfies Assumption 3.2, for* $s = 1, \dots, m$*.*

The proof can be found in Appendix E.3. The $d_j = 1$ condition is met without loss of generality, by splitting the higher dimensional feature maps and introducing more base features, which will increase p. Moreover, the orthonormality condition is met by orthogonalizing and re-scaling the feature maps. Basis functions such as Legendre polynomials and Fourier features [Rahimi et al., 2007] satisfy these conditions.

Generally, it is natural to require BASEBO to explore more in lifelong setting compared to when it is used in isolation and with a known kernel. We observe in our experiments that LIBO has a better empirical performance with forced exploration (i.e., $n_s > 0$) than without. This additional exploration is also required in the Representation Learning [Yang et al., 2021, 2022, Cella and Pontil, 2021, Cella et al., 2022] and hierarchical Bayesian bandit literature [Basu et al., 2021, Peleg et al., 2022, Hong et al., 2022], where it is assumed that either the context distribution or the

[1] Appendix E.1 provides a background on GP-UCB.

chosen actions are diverse enough. In the case of contextual bandits, if there is sufficient randomness, the BASEBO can be greedy and yet sample diverse enough actions [Bastani et al., 2021]. Table 3 gives a detailed overview of how the related works rely on uniform exploration.

5 FEDERATED LIBO

We consider a federated extension of the lifelong learning problem. Here, each BO task is performed by a peer in a network and the corresponding data is not exchanged due to privacy concerns, limited bandwidth, etc. Operations are mainly done at the client level, and the central server only performs light computations. This setting formalizes problems such as optimizing the user experience of a software product on each user's device, e.g., for making better recommendations. Limiting the client-server communication reduces the transmit overhead time and motivates faster federated computation. Moreover, sending detailed data on user preferences and interaction patterns to the central server may jeopardize the user's privacy. However, we want to harness the statistical patterns across the user pool to improve the automated tailoring of the software product to new users. We interpret such a federated learning problem [Kairouz et al., 2021] as a client-server adaptation of our lifelong setting as described in Section 2. The meta-agent represents the server and BO tasks arise sequentially at a client node $s = 1, ..., m$ with a client specific reward function f_s.

We propose the *Federated Lifelong Bandit Optimizer* (F-LIBO) to solve this problem without directly sharing $\mathcal{D}_s$ the data corresponding to each client with the server. F-LIBO, pairs the clients and the server as follows. First, the client node s receives $\hat{k}_{s-1}$, the most recent estimate of the true kernel, and the required number of forced exploration queries n_s from the server. After taking some exploratory steps, the client performs actions according to its BASEBO policy. In contrast to LIBO, once the task is over after n steps, the client keeps $\mathcal{D}_s^{\text{exp}}$ to itself, instead of passing it back the server. The client node optimizes for a local loss

$$\begin{aligned}\hat{\boldsymbol{\beta}}_s^{(\text{client})} &\coloneqq \arg\min_{\boldsymbol{\beta}_s \in \mathbb{R}^d} \mathcal{L}\left(\boldsymbol{\beta}_s; \mathcal{D}_s^{\text{exp}}\right) \\ &= \arg\min_{\boldsymbol{\beta}_s \in \mathbb{R}^d} \frac{1}{n_s}\|\boldsymbol{y}_s - \boldsymbol{\Phi}_s\boldsymbol{\beta}_s\|_2^2 + \lambda \sum_{j=1}^{p} \|{\boldsymbol{\beta}_s}^{(j)}\|_2,\end{aligned}$$

and calculates a local estimate of $J^\star$ by thresholding $\hat{\boldsymbol{\beta}}_s^{(\text{client})}$ with the hyperparamter $\omega > 0$

$$\hat{J}_s^{(\text{client})} \coloneqq \left\{ j \in \{1, \dots, p\} \text{ s.t. } \|\hat{\boldsymbol{\beta}}_s^{(\text{client})(j)}\|_2 > \omega \right\}.$$

It then sends *only* the indices $\hat{J}_s^{(\text{client})}$ back to the server. This leaves the server with the simple task of taking a α-majority vote among the s first clients, to decide which base kernels to include in $\hat{k}_s$. Formally, the server chooses

$$\hat{k}_s(\boldsymbol{x}, \boldsymbol{x}') \coloneqq \frac{1}{|\hat{J}_s|} \sum_{j \in \hat{J}_s} k_j(\boldsymbol{x}, \boldsymbol{x}')$$

where for $\alpha \in [0, 1]$,

$$\hat{J}_s \coloneqq \left\{ j \in \{1, \dots, p\} \text{ s.t. } \sum_{r=1}^{s} \mathbb{1}(j \in \hat{J}_r^{(\text{client})}) \geq s\alpha \right\}.$$

In other words, after client s finishes its job, the server includes the j-th kernel into its updated estimate $\hat{J}_s$, if and only if more than $s\alpha$ of the clients so far believe that it should be included. Figure 1 in the appendix visualizes this process and Algorithm 3 presents the pseudo-code to F-LIBO. Similar to LIBO, we show that if $R^\star(n)$ the worst-case oracle regret of the base bandit algorithm is sublinear in n, then so is the lifelong regret $R(n, m)$ of F-LIBO:

Theorem 5.1. *For all tasks $s = 1, \dots, m$, assume that the reward function $f_s \in \mathcal{H}_{k^\star}$ has bounded RKHS norm $\|f_s\|_{k^\star} \leq B$. Set the number of forced exploration actions as $n_s = \sqrt{n}$, and assume that Assumption 3.1 and 3.2 hold for the data $\mathcal{D}_s^{\text{exp}}$. Suppose, with probability $> 1 - \delta/2$, that* BASEBO *has worst-case oracle regret $R^\star(m, n)$. Then the lifelong regret of* F-LIBO *satisfies*

$$R(m, n) - R^\star(m, n) = \mathcal{O}\Big(Bm\sqrt{n} + B\sqrt{n}\log(mp/\delta)\Big)$$

with probability greater than $1 - \delta$.

See Appendix F.2 for the proof. Theorem 5.1 demonstrates that even without direct access to the data, the lifelong regret of F-LIBO will be sublinear in n. This theorem does not imply oracle optimality, since $R(m, n)/m - R^\star(n) \not\to 0$ for $m \to \infty$. This is due to the linear dependency of the first term on m, which arises from forced exploration. In the federated setting, we require all clients to take a fixed number of exploratory action $n_s = \sqrt{n}$, so that they have equal resources for estimating $J^\star$ and the server's majority vote is fair. We conjecture that with simple modifications, LIBO can become provably differentially private. Replacing the majority voting step with GNMax Aggregator [Papernot et al., 2018] or PRIME [Liu et al., 2021] yields a differential private voting mechanism to select $\hat{J}_s$, while preserving the lifelong regret guarantee.

Consider an example where we instantiate F-LIBO with GP-UCB as BASEBO. The worst-case regret bound of F-LIBO, which neither has knowledge of $k^\star$ nor direct access to $\mathcal{D}_s^{\text{exp}}$, matches the worst-case regret of the oracle GP-UCB in $\mathcal{O}$-notation. Corollary 5.2 formalizes this claim. Here, $R^\star(m, n)$ is the same as in Corollary 4.2.

Corollary 5.2 (Federated Lifelong GP-UCB)**.** *Consider the setting of Theorem 5.1 with* GP-UCB *as* BASEBO*. Then, with probability at least $1 - \delta$,* F-LIBO *paired with* GP-UCB *satisfies*

$$R(m, n) = \mathcal{O}\big(R^\star(m, n)\big).$$

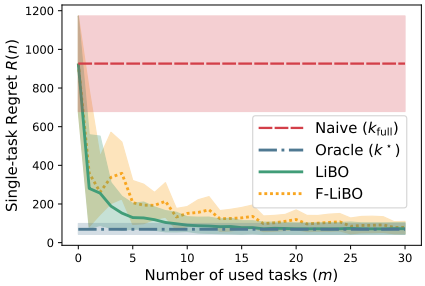

Figure 2: Singe-task cumulative regret of GP-UCB with meta-learned kernel $\hat{k}_m$ on an increasing number of meta-training tasks m.

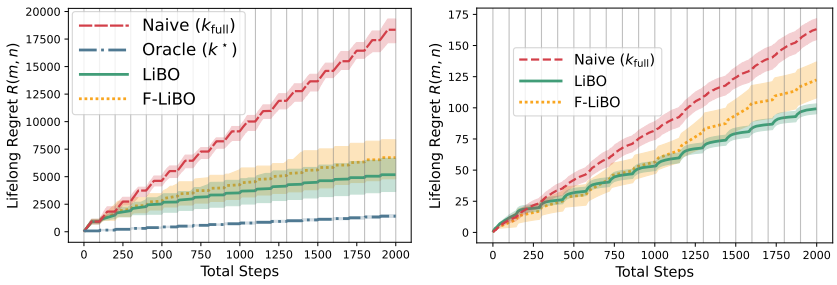

Figure 3: Lifelong cumulative regret of GP-UCB for synthetic tasks (left) and GLMNET hyperparameter tuning (right). Vertical lines indicate the beginning of a new task.

6 EXPERIMENTS

In all experiments, we use GP-UCB as the BASEBO. We repeat all experiments with 20 random seeds and report the corresponding mean outcome with standard error. To evaluate the proposed algorithms, we use a synthetic as well as a hyper-parameter tuning environment.

Synthetic environment. The synthetic environment is based on our data model from Section 2. We choose $\mathcal{X} = [0, 1]$ as the domain and use the first $p = 50$ cosine basis functions $\phi_j(x) = \cos(j\pi x)$ as feature maps which form the kernels $k_j(x, x') = \phi_j(x)^\top \phi_j(x')$ for $j \in \{1, ..., 50\}$. The active kernel indices $J^\star$ are sampled uniformly from the set of 5-element subsets of $\{1, ..., 50\}$, i.e., $|J^\star| = 5$. We sample the reward functions f_s independently and uniformly from $\mathcal{H}_{k^\star}$ such that $||f||_{k^\star} \leq 10$ and beta-min condition of $c_1 \geq 0.5$ holds. To the function evaluations we add i.i.d. Gaussian noise with a standard deviation of $\sigma = 0.1$.

AutoML data. A common application of Bayesian Optimization is AutoML, i.e., optimizing the hyper-parameters of machine learning algorithms. In this setting, $\mathcal{X}$ is the learning algorithm's hyper-parameter space, and f_s represents the test performance of the machine learning system. In our experiments, we consider a realistic lifelong AutoML setting where we face a sequence of hyper-parameter optimization problems. Here, each task corresponds to tuning the hyper-parameters of the GLMNET learning algorithm [Friedman et al., 2010] for a different dataset. Following previous work [e.g. Perrone et al., 2018, Rothfuss et al., 2021b], we replace the evaluation step by a table lookup based on a large number of hyper-parameter evaluations [Kühn et al., 2018] on 38 classification datasets from the OpenML platform [Bischl et al., 2017].

6.1 EXPERIMENT WITH OFFLINE DATA

We investigate how meta-leaning kernels with META-KGL and its federated variant (F-META-KGL) affects the performance of test tasks. In particular, we use meta-training data that was generated offline, based on the synthetic environment. We create data for $m = 30$ synthetic tasks, each of size $n = 10$ to be used as offline meta-data. The tasks are generated according to our synthetic environment and the details can be found in Appendix G.1. Note that $n \ll p$ i.e., we are in the overparameterized setting. We meta-learn a kernel with META-KGL and F-META-KGL using the meta-training data $\mathcal{D}_{1:s}$ for $s = 1, ..., m$, and evaluate the estimated kernel $\hat{k}$ by running GP-UCB, equipped with $\hat{k}$, for $n = 70$ iterations. Figure 2 illustrates the corresponding single-task regrets in response to increasing the number of meta-training tasks in the offline data. We report the performance of the agent that uses k^{full} (red) as a naive baseline, and the performance of an oracle agent that uses the true kernel (blue) as a natural lower bound for the achievable regret. Figure 2 shows that the regret of both meta-learned agents quickly converges to the regret of the oracle agent as the number of meta-training tasks increases. META-KGL performs slightly better than F-META-KGL, since it has direct access to all the data while the federated algorithm loses information during the voting mechanism. In Appendix G.3, we evaluate META-KGL and F-META-KGL with other choices of base kernels and higher dimensional action domains. Similar to Figure 2, we observe fast convergence to the oracle regret. Further details about the experiments are provided in Appendix G.1.

6.2 LIFELONG EXPERIMENTS

We return to the lifelong setting where the tasks arrive sequentially and evaluate our algorithms. We consider both the synthetic and AutoML environments. The horizon of each task is set to $n = 100$ time steps. To solve the synthetic problem, we consider the $p = 50$ first 1-dimensional cosine bases as the candidate feature maps. Since GLMNET has two hyper-parameters to tune, i.e., $\mathcal{X} \subset \mathbb{R}^2$, here we use the $p = 100$ first 2-dimensional cosine bases, i.e. $\phi_{i,j}(\boldsymbol{x}) = \cos(i\pi x_1)\cos(j\pi x_2)$ for $i, j = 1, ..., 10$.

Figure 3 illustrates the cumulative lifelong regrets achieved by LIBO, F-LIBO, the baseline GP-UCB with k^{full}, and

	oracle optimal	policy agnostic	learns sparsity	meta cost	tasks
Hu et al.	✗	✗	✗	poly d	conc
Yang et al.	✓	✗	✗	poly d	conc
Peleg et al.	✗	✗	✗	poly d	seq
Hong et al.	✓	✗	✗	poly d	seq
	✗	✗	✗	poly d	conc
LIBO	✓	✓	✓	log d	seq

Table 1: Related work (Table 3 gives a comprehensive list.)

oracle GP-UCB with access to $k^\star$. Note that in the AutoML environment, we do not know the true kernel $k^\star$ and thus, cannot report the oracle performance. As we would expect, LIBO and F-LIBO initially suffer the same regret as the the naive actor with k^{full} since no meta-learning data is available yet. However, as more tasks are attempted, the estimated kernel is improved and in turn, the base algorithm becomes more sample efficient on future tasks. In case of LIBO (green), over time, the forced exploration decreases and the estimated kernel converges to the true kernel. As a result, the behavior of the actor paired with the LIBO becomes indistinguishable from the actor using the oracle kernel, reflected by the same slope of the regret curves. Compared to the naive actor (red), our lifelong BO methods significantly improve the efficiency of the base agent as they accumulate more experience. In the AutoML setting, this means that we can find good hyper-parameters with fewer costly function evaluations. This showcases how incorporating knowledge transfer into deployed machine learning systems can yield significant performance gains and cost savings.

7 RELATED WORK

The lifelong bandit optimization problem addresses key shortcomings of classic kernelized bandits and Bayesian optimization. Early approaches assume that the agent knows the true kernel [Srinivas et al., 2010, Valko et al., 2013, Chowdhury and Gopalan, 2017], which is often not the case in practice. Recent work addresses this problem, either by studying the implications of misspecified kernels [Foster et al., 2020, Simchowitz et al., 2021, Bogunovic and Krause, 2021, Camilleri et al., 2021] or proposing methods for adapting kernel parameters during the optimization [Wang and de Freitas, 2014, Berkenkamp et al., 2019]. Alternatively, the appropriate kernel can be learned from related data. To this end, a number of algorithms are developed for meta-learning a kernelized Gaussian process (GP) prior [Harrison et al., 2018, Perrone et al., 2018, Rothfuss et al., 2021a,b, 2022]. However, they come without theoretical guarantees.

Theory of knowledge transfer between concurrent or sequential linear bandits has received recent attention from multiple perspectives. Representation Learning literature [Yang et al., 2021, Hu et al., 2021, Yang et al., 2022, Cella et al., 2022] assumes existence of a shared low-dimensional linear representation for the reward function, i.e. $f_s(\boldsymbol{x}) = \langle \boldsymbol{\theta}_s, \boldsymbol{B}^T \boldsymbol{x} \rangle$ where $\boldsymbol{B} \in \mathbb{R}^{d \times d^\star}$ is shared by the tasks. This matrix is unknown, however $d^\star$ is known and $d^\star \ll d$. Feature selection [Cella and Pontil, 2021] takes a similar approach by assuming that $f_s(\boldsymbol{x}) = \langle \boldsymbol{\theta}_s, \boldsymbol{S}^T \boldsymbol{x} \rangle$, where the unknown matrix $\boldsymbol{S}$ screens the relevant features $\{\boldsymbol{x}_j, j \in J^\star\}$. The elements of this matrix are 0 or 1, but contrary to representation learning, $d^\star = |J^\star|$ is unknown. Alternatively, works on Bayesian Prior learning assume existence of a shared Gaussian prior over the parameter vector, i.e. $f_s(\boldsymbol{x}) = \langle \boldsymbol{\theta}_s, \boldsymbol{x} \rangle$, where $\boldsymbol{\theta}_s \sim \mathcal{N}(\mu, \Sigma)$. This formulation does not aim for a low-dimensional solution. Following this model, Basu et al. [2021] and Hong et al. [2022] assume that Σ is known and learn distribution of μ. Peleg et al. [2022] estimate both the mean and the covariance. We consider a more relaxed setup where the mean is not shared, and meta-learn a shared covariance function. Appendix B goes into more depth to formally compare the mentioned work.

We compare LIBO with prior algorithms based on the following properties. In the context of meta-learning for BO, a desirable method is 1) oracle optimal, i.e., attains the regret guarantee of the oracle solver as m grows, 2) able to utilize any BO algorithm, 3) sample efficient, i.e., pays a small cost for meta-learning the prior/relevant features and 4) recovers low-dimensional solutions, since the effective dimension influences the sample efficiency of the base algorithm. Table 1 compares LIBO with previous work applicable to infinite action domains. Works limited to finite action sets are considered in Table 3. LIBO is the only oracle optimal algorithm that learns the effective dimension $d^\star$, while paying a cost that scales only logarithmically with the Euclidean dimension d. This is an exponential improvement compared to the polynomial dependency of prior work; moreover, it also applies to reward functions that are a linear combinations of non-linear features $f_s(\boldsymbol{x}) = \langle \boldsymbol{\theta}_s, \boldsymbol{\phi}(\boldsymbol{x}) \rangle$. Further, it can be wrapped around any linear or kernelized bandit algorithm, while earlier work require a specific bandit policy.

F-LIBO contributes to recent literature on federated learning which studies how agents can cooperate to solve a single bandit task [Dubey and Pentland, 2020, Shi et al., 2021, Huang et al., 2021, Dai et al., 2022]. In federated lifelong learning, each agent interacts with a different environment, but collaborates with others to learn relevant features.

Our work builds on ideas from Multiple-Kernel Learning [Cristianini et al., 2001, Bach et al., 2004, Ong et al., 2005, Xu et al., 2010, Gönen and Alpaydın, 2011] and Multi-Task Lasso [Obozinski et al., 2006, Argyriou et al., 2006, Lounici et al., 2011] which address consistency of model selection for offline supervised learning. Our contribution is lifelong uncertainty quantification, using a meta-learned kernel.

8 CONCLUSION

We introduce LIBO, an algorithm which allows for lifelong knowledge transfer across BO tasks trough meta-learned kernels. We show theoretically and empirically that, if paired with LIBO, the performance of a base bandit algorithm improves as more experience is gained on previous tasks. In particular, we prove that LIBO is oracle optimal in the limit. With F-LIBO, the federated variant of our main algorithm, we establish that sublinear knowledge transfer is possible even without direct access to the bandit data.

This work opens up directions of future research such as quantifying the cost of privacy in Lifelong Learning, understanding the necessity of exploration in lifelong setting, or using large neural networks to extract relevant features from prior tasks instead of working with pre-determined features.

Acknowledgements

This research was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program grant agreement no. 815943. Jonas Rothfuss was supported by the Apple Scholars in AI/ML fellowship.

References

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. *NeurIPS*, 24, 2011.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *NeurIPS*, 2006.

Francis R Bach, Gert RG Lanckriet, and Michael I Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *ICML*, 2004.

Hamsa Bastani and Mohsen Bayati. Online decision making with high-dimensional covariates. *Operations Research*, 2020.

Hamsa Bastani, Mohsen Bayati, and Khashayar Khosravi. Mostly exploration-free algorithms for contextual bandits. *Management Science*, 2021.

Soumya Basu, Branislav Kveton, Manzil Zaheer, and Csaba Szepesvári. No regrets for learning the prior in bandits. *NeurIPS*, 2021.

Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. No-regret bayesian optimization with unknown hyperparameters. *JMLR*, 2019.

B. Bischl, Giuseppe Casalicchio, Matthias Feurer, F. Hutter, Michel Lang, R. Mantovani, J. N. Rijn, and J. Vanschoren. Openml benchmarking suites and the openml100. *arXiv preprint*, 2017.

Ilija Bogunovic and Andreas Krause. Misspecified gaussian process bandit optimization. *NeurIPS*, 2021.

Peter Bühlmann and Sara Van De Geer. *Statistics for high-dimensional data: methods, theory and applications*. Springer Science & Business Media, 2011.

Florentina Bunea, Johannes Lederer, and Yiyuan She. The group square-root lasso: Theoretical properties and fast algorithms. *IEEE Transactions on Information Theory*, 2013.

Romain Camilleri, Kevin Jamieson, and Julian Katz-Samuels. High-dimensional experimental design and kernel bandits. In *ICML*, 2021.

Leonardo Cella and Massimiliano Pontil. Multi-task and meta-learning with sparse linear bandits. In *UAI*, 2021.

Leonardo Cella, Karim Lounici, and Massimiliano Pontil. Meta representation learning with contextual linear bandits. *arXiv preprint*, 2022.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *ICML*, 2017.

Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *NeurIPS*, 2001.

Zhongxiang Dai, Yao Shu, Arun Verma, Flint Xiaofeng Fan, Bryan Kian Hsiang Low, and Patrick Jaillet. Federated neural bandit. *arXiv preprint*, 2022.

Abhimanyu Dubey and AlexSandy Pentland. Differentially-private federated linear bandits. *NeurIPS*, 2020.

Dylan J Foster, Claudio Gentile, Mehryar Mohri, and Julian Zimmert. Adapting to misspecification in contextual bandits. In *NeurIPS*, 2020.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 2010.

Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *JMLR*, 2011.

Botao Hao, Tor Lattimore, and Mengdi Wang. High-dimensional sparse linear bandits. *NeurIPS*, 2020.

James Harrison, Apoorva Sharma, and Marco Pavone. Meta-learning priors for efficient online bayesian regression. In *International Workshop on the Algorithmic Foundations of Robotics*, 2018.

Joey Hong, Branislav Kveton, Manzil Zaheer, and Mohammad Ghavamzadeh. Hierarchical bayesian bandits. In *AISTATS*, 2022.

Jiachen Hu, Xiaoyu Chen, Chi Jin, Lihong Li, and Liwei Wang. Near-optimal representation learning for linear bandits and linear rl. In *ICML*, 2021.

Ruiquan Huang, Weiqiang Wu, Jing Yang, and Cong Shen. Federated linear contextual bandits. *NeurIPS*, 2021.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 2021.

Parnian Kassraie, Jonas Rothfuss, and Andreas Krause. Meta-Learning Hypothesis Spaces for Sequential Decision-making. In *ICML*, 2022.

Gi-Soo Kim and Myunghee Cho Paik. Doubly-robust lasso bandit. *NeurIPS*, 2019.

Daniel Kühn, Philipp Probst, Janek Thomas, and Bernd Bischl. Automatic exploration of machine learning experiments on openml. *arXiv preprint*, 2018.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Yingkai Li, Yining Wang, and Yuan Zhou. Nearly minimax-optimal regret for linearly parameterized bandits. In *Conference on Learning Theory*, 2019.

Yingkai Li, Yining Wang, Xi Chen, and Yuan Zhou. Tight regret bounds for infinite-armed linear contextual bandits. In *AISTATS*, 2021.

Xiyang Liu, Weihao Kong, Sham Kakade, and Sewoong Oh. Robust and differentially private mean estimation. *NeurIPS*, 2021.

Karim Lounici, Massimiliano Pontil, Sara Van De Geer, and Alexandre B Tsybakov. Oracle inequalities and optimal inference under group sparsity. *The annals of statistics*, 2011.

Mathurin Massias, Alexandre Gramfort, and Joseph Salmon. Celer: a fast solver for the lasso with dual extrapolation. In *ICML*, 2018.

Guillaume Obozinski, Ben Taskar, and Michael Jordan. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2006.

Cheng Soon Ong, Alexander J. Smola, and Robert C. Williamson. Learning the kernel with hyperkernels. *JMLR*, 2005.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Ulfar Erlingsson. Scalable private learning with PATE. In *ICLR*, 2018.

Amit Peleg, Naama Pearl, and Ron Meir. Metalearning linear bandits by prior update. In *AISTATS*, 2022.

Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *NeurIPS*, 2018.

Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *NeurIPS*, 2007.

Jonas Rothfuss, Vincent Fortuin, Martin Josifoski, and Andreas Krause. PACOH: Bayes-optimal meta-learning with PAC-guarantees. In *ICML*, 2021a.

Jonas Rothfuss, Dominique Heyn, Andreas Krause, et al. Meta-learning Reliable Priors in the Function Space. In *NeurIPS*, 2021b.

Jonas Rothfuss, Christopher Koenig, Alisa Rupenyan, and Andreas Krause. Meta-learning priors for safe bayesian optimization. In *Conference on Robot Learning (CoRL)*, 2022.

Daniel Russo and Benjamin Van Roy. Learning to optimize via information-directed sampling. *NeurIPS*, 2014.

Chengshuai Shi, Cong Shen, and Jing Yang. Federated multi-armed bandits with personalization. In *AISTATS*, 2021.

Max Simchowitz, Christopher Tosh, Akshay Krishnamurthy, Daniel J Hsu, Thodoris Lykouris, Miro Dudik, and Robert E Schapire. Bayesian decision-making under misspecified priors with applications to meta-learning. *NeurIPS*, 2021.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 1933.

Sattar Vakili, Kia Khezeli, and Victor Picheny. On information gain and regret bounds in gaussian process bandits. In *AISTATS*. PMLR, 2021.

Michal Valko, Nathaniel Korda, Rémi Munos, Ilias Flaounas, and Nelo Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint*, 2013.

Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

Ziyu Wang and Nando de Freitas. Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters. *arXiv preprint*, 2014.

Zenglin Xu, Rong Jin, Haiqin Yang, Irwin King, and Michael R Lyu. Simple and efficient multiple kernel learning by group lasso. In *ICML*, 2010.

Jiaqi Yang, Wei Hu, Jason D. Lee, and Simon Shaolei Du. Impact of representation learning in linear bandits. In *ICLR*, 2021.

Jiaqi Yang, Qi Lei, Jason D Lee, and Simon S Du. Nearly minimax algorithms for linear bandits with shared representation. *arXiv preprint*, 2022.

Peng Zhao and Bin Yu. On model selection consistency of lasso. *JMLR*, 2006.