Towards Practical Lipschitz Bandits

Tianyu Wang
tianyu@cs.duke.edu
Duke University
Durham, USA

Weicheng Ye
weicheny@andrew.cmu.edu
Carnegie Mellon University
Pittsburgh, USA

Dawei Geng*
dawei.geng@duke.edu
Autodesk, Inc.
San Francisco, USA

Cynthia Rudin
cynthia@cs.duke.edu
Duke University
Durham, USA

ABSTRACT

Stochastic Lipschitz bandit algorithms balance exploration and exploitation, and have been used for a variety of important task domains. In this paper, we present a framework for Lipschitz bandit methods that adaptively learns partitions of context- and arm-space. Due to this flexibility, the algorithm is able to efficiently optimize rewards and minimize regret, by focusing on the portions of the space that are most relevant. In our analysis, we link tree-based methods to Gaussian processes. In light of our analysis, we design a novel hierarchical Bayesian model for Lipschitz bandit problems. Our experiments show that our algorithms can achieve state-of-the-art performance in challenging real-world tasks such as neural network hyperparameter tuning.

CCS CONCEPTS

• **Theory of computation → Online learning algorithms**.

KEYWORDS

Bandit Algorithms, Lipschitzness, Gaussian processes, Hyperparameter Tuning

ACM Reference Format:
Tianyu Wang, Weicheng Ye, Dawei Geng, and Cynthia Rudin. 2020. Towards Practical Lipschitz Bandits. In *Proceedings of the 2020 ACM-IMS Foundations of Data Science Conference (FODS '20), October 19–20, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3412815.3416885

1 INTRODUCTION

Stochastic Lipschitz bandit algorithms are methods that balance exploration-exploitation tradeoffs. Their usage arises in important real-world scenarios. For example, in medical trials, a doctor might deliver a sequence of treatment options with the goal of achieving the best total treatment effect, or with the goal of allocating the best treatment option as efficiently as possible, without conducting too many trials.

A stochastic bandit problem assumes that payoffs are noisy and are drawn from an unchanging distribution. The study of stochastic bandit problems started with the discrete arm setting, where the agent is faced with a finite set of choices. Classic works on this problem include Thompson sampling [2, 27], Gittins index [13], ϵ-greedy strategies [26], and upper confidence bound (UCB) methods [5, 16]. One recent line of work on stochastic bandit problems considers the case where the arm space is infinite. In this setting, the arms are usually assumed to be in a subset of the Euclidean space (or a more general metric space), and the expected payoff function is assumed to be a function of the arms. Some works along this line model the expected payoff as a linear function of the arms [1, 3, 4, 11, 17]; some algorithms model the expected payoff as Gaussian processes over the arms [10, 12, 25]; some algorithms assume that the expected payoff is a Lipschitz function of the arms [9, 15, 19, 24]; and some assume locally Hölder payoffs on the real line [6]. When the arms are continuous and equipped with a metric, and the expected payoff is Lipschitz continuous in the arm space, we refer to the problem as a stochastic Lipschitz bandit problem. In addition, when the agent's decisions are made with the aid of contextual information, we refer to the problem as a contextual stochastic Lipschitz bandit problem. Not many works [9, 15, 19] have considered the general Lipschitz bandit problem without making strong assumptions on the smoothness of rewards in context-arm space. In this paper, we focus our study on this general (contextual) stochastic Lipschitz bandit problem, and provide practical algorithms for use in data science applications.

Specifically, we propose a framework that converts a general decision tree algorithm into an algorithm for stochastic Lipschitz bandit problems. We use a novel analysis that links our algorithms to Gaussian processes; though the underlying rewards do not need to be generated by any Gaussian process. Based on this connection, we can use a novel hierarchical Bayesian model to design a new (UCB) index. This new index solves two main problems suffered by partition based bandit algorithms. Namely, (1) within each bin of the partition, all arms are treated the same; (2) disjoint bins do not use information from each other.

Empirically, we show that using adaptively learned partitions, Lipschitz bandit algorithms can be used for hard real-world problems such as hyperparameter tuning for neural networks.

Relation to prior work: One general way of solving stochastic Lipschitz bandit problems is to finely discretize (partition) the arm

*Most work was done while at Duke University.

FODS '20, October 19–20, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8103-1/20/10…$15.00
https://doi.org/10.1145/3412815.3416885

space and treat the problem as a finite-arm problem. An Upper Confidence Bound (UCB) strategy can thus be used. Previous algorithms of this kind include the `UniformMesh` algorithm [15], the HOO algorithm [9], and the (contextual) Zooming Bandit algorithm [15, 24]. While all these algorithms employ different analysis techniques, we show that as long as a discretization of the arm space fulfills certain requirements (outlined in Theorem 1), these algorithms (or a possibly modified version) can be analyzed in a unified framework.

The practical problem with previous methods is that they require either a fine discretization of the full arm space or restrictive control of the partition formation (e.g., Zooming rule [15]), leading to implementations that are not flexible. By fitting decision trees that are grown adaptively during the run of the algorithm, our partition can be *learned* from data. This advantage enables the algorithm to outperform leading methods for Lipschitz bandits (e.g. [9, 15]) and for zeroth order optimization (e.g. [18, 20]) on hard real-world problems that can involve difficult arm space and reward landscape. As shown in the experiments, in neural network hyperparameter tuning, our methods can outperform the state-of-the-art benchmark packages that are tailored for hyperparameter selection.

In summary, our contributions are: **1)** We develop a novel stochastic Lipschitz bandit framework, *TreeUCB* and its contextual counterpart *Contextual TreeUCB*. Our framework converts a general decision tree algorithm into a stochastic Lipschitz bandit algorithm. Algorithms arising from this framework empirically outperform benchmarks methods. **2)** We develop a new analysis framework, which can be used to recover previous known bounds, and design a new principled acquisition function in bandits and zero-th order optimization.

2 MAIN RESULTS

2.1 The TreeUCB framework

Stochastic bandit algorithms, in an online fashion, explore the decision space while exploit seemingly good options. The performance of the algorithm is typically measured by regret. In this paper, we focus our study on the following setting. A payoff function is defined over an arm space that is a compact doubling metric space $(\mathcal{A}, d)$, the payoff function of interest is $f : \mathcal{A} \to [0, 1]$, and the actual observations are given by $y(a) = f(a) + \epsilon_a$. In our setting, the noise distribution ϵ_a could vary with a, as long as it is uniformly mean zero, almost surely bounded, and independent of f for every a. Our results easily generalize to sub-Gaussian noise [23]. In the analysis, we assume that the (expected) payoff function f is Lipschitz in the sense that $\forall a, a' \in \mathcal{A}$, $|f(a) - f(a')| \leq Ld(a, a')$ for some Lipschitz constant L. An agent is interacting with this environment in the following fashion. At each round t, based on past observations $(a_1, y_1, \cdots, a_{t-1}, y_{t-1})$, the agent makes a query at point a_t and observes the (noisy) payoff y_t, where y_t is revealed only after the agent has made a decision a_t. For an agent executing algorithm `Alg`, the regret incurred up to time T is defined to be:

$$R_T(\texttt{Alg}) = \sum_{t=1}^{T} \left(f(a^*) - f(a_t) \right),$$

where a^* is the global maximizer of f.

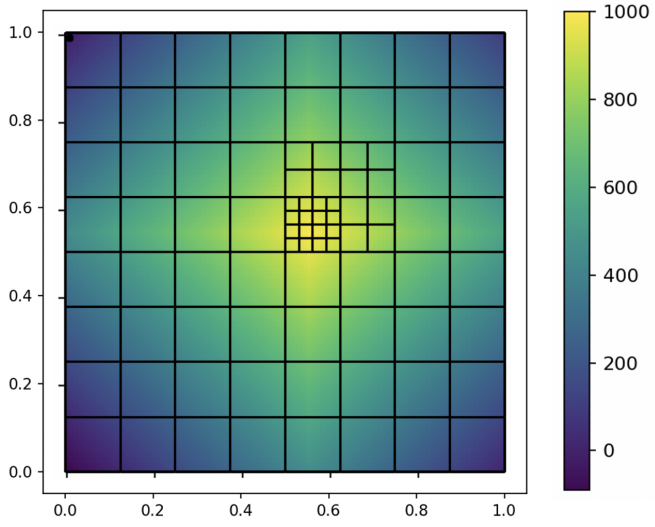


Figure 1: Example reward function (in color gradient) with an example partitioning.

Any TreeUCB algorithm runs by maintaining a sequence of finite partitions of the arm space. Intuitively, at each step t, TreeUCB treats the problem as a finite-arm bandit problem with respect to the partition bins at t, and chooses an arm uniformly at random within the chosen bin. The partition bins become smaller and smaller as the algorithm runs. Thus, at any time t, we maintain a partition $\mathcal{P}_t = \left\{ P_t^{(1)}, \cdots, P_t^{(k_t)} \right\}$ of the input space. That is, $P_t^{(1)}, \cdots, P_t^{(k_t)}$ are subsets of $\mathcal{A}$, are mutually disjoint and $\cup_{i=1}^{k_t} P_t^{(i)} = \mathcal{A}$.

As an example, Figure 1 shows an partitioning of the input space, with the underlying reward function shown by color gradient. In an algorithm run, we collect data and estimate the reward with respect to the partition. Based on the estimate, we select a "box" to play next.

Each element in the partition is called a *region* and by convention $\mathcal{P}_0 = \{\mathcal{A}\}$. The regions could be leaves in a tree, or chosen in some other way.

Given any t, if for any $P^{(i)} \in \mathcal{P}_{t+1}$, there exists $P^{(j)} \in \mathcal{P}_t$ such that $P^{(i)} \subset P^{(j)}$, we say that $\{\mathcal{P}_t\}_{t \geq 0}$ is a sequence of ***nested partitions.*** In words, at round t, some regions (or no regions) of the partition are split into multiple regions to form the partition at round $t + 1$. We also say that the partition ***grows finer.***

Based on the partition $\mathcal{P}_t$ at time t, we define an auxiliary function – the *Region Selection function.*

Definition 1 (Region Selection Function). *Given partition $\mathcal{P}_t$, function $p_t : \mathcal{A} \to \mathcal{P}_t$ is called a Region Selection Function with respect to $\mathcal{P}_t$ if for any $a \in \mathcal{A}$, $p_t(a)$ is the region in $\mathcal{P}_t$ containing a.*

As the name TreeUCB suggests, our framework follows an Upper Confidence Bound (UCB) strategy. In order to define our Upper Confidence Bound, we require several definitions.

Definition 2. *Let $\mathcal{P}_t$ be the partition of $\mathcal{A}$ at time t ($t \geq 1$) and let p_t be the Region Selection Function associated with $\mathcal{P}_t$. Let $(a_1, y_1, a_2, y_2, \cdots, a_{t'}, y_{t'})$ be the observations received up to time t' ($t' \geq 1$). We define*

• *the count function* $n^0_{t,t'} : \mathcal{A} \to \mathbb{R}$, *such that*

$$n^0_{t,t'}(x) = \sum_{i=1}^{t'} \mathbb{I}[x_i \in p_t(x)].$$

• *the corrected average function* $m_{t,t'} : \mathcal{A} \to \mathbb{R}$, *such that*

$$m_{t,t'}(a) = \begin{cases} \frac{\sum_{i=1}^{t'} y_i \mathbb{I}[a_i \in p_t(a)]}{n^0_{t,t'}(a)}, & \text{if } n^0_{t,t'}(a) > 0; \\ 1, \quad \text{otherwise.} \end{cases} \tag{1}$$

• *the corrected count function, such that*

$$n_{t,t'}(x) = \max\left(1, n^0_{t,t'}(x)\right). \tag{2}$$

When $t = t'$, *we shorten the notation from* $m_{t,t'}$ *to* m_t, $n^0_{t,t'}$ *to* n^0_t, *and* $n_{t,t'}$ *to* n_t.

In words, $n^0_{t,t'}(a)$ is the number of points among $(a_1, a_2, \cdots, a_{t'})$ that are in the same region as arm a, with regions as elements in $\mathcal{P}_t$. We also denote by $D(\mathcal{S})$ the diameter of $\mathcal{S} \subset \mathcal{A}$, and $D(\mathcal{S}) := \sup_{a', a'' \in \mathcal{S}} d(a', a'')$.

At time t, based on the partition and observations, our bandit algorithm uses, for $a \in \mathcal{A}$

$$U_t(a) = m_{t-1}(a) + C\sqrt{\frac{4\log t}{n_{t-1}(a)}} + M \cdot D(p_t(a)), \tag{3}$$

for some C and M as the Upper Confidence Bound of arm a; and we play an arm with the highest U_t value (with ties broken uniformly at random).

Remark 1. *As we will discuss in Section 2.3.2, the upper confidence index for our decision can take different forms other than (3).*

Here C depends on the almost sure bound on the reward, and M depends on the Lipschitz constant of the expected reward, which are both problem intrinsics.

Since U_t is a piece-wise constant function in the arm-space and is constant within each region, playing an arm with the highest U_t with random tie-breaking is equivalent to selecting the best region (under UCB) and randomly selecting an arm within the region. After deciding which arm to play, we update the partition into a finer one if eligible. This strategy, TreeUCB, is summarized in Algorithm 1. We also provide a provable guarantee for TreeUCB algorithms in Theorem 1.

Theorem 1. *Suppose that the payoff function* f *defined on a compact domain* $\mathcal{A}$ *satisfies* $f(a) \in [0, 1]$ *for all* a *and is Lipschitz. Let* $\mathcal{P}_t$ *be the partition at time* t *in Algorithm 1. If the tree fitting rule* $\mathcal{R}$ *satisfies*
(1) $\{\mathcal{P}_t\}_{t \geq 0}$ *is a sequence of nested partitions (or the partition grows finer);*
(2) $|\mathcal{P}_t| = o(t^\gamma)$ *for some* $\gamma < 1$;
(3) $D(p_t(a)) = o(1)$ *for all* $a \in \mathcal{A}$, *where*

$$D(p_t(a)) := \sup_{a', a'' \in p_t(a)} d(a', a'')$$

is the diameter of region $p_t(a)$;
(4) given all realized observations $\{(a_t, y_t)\}_{t=1}^T$, *the partitions* $\{\mathcal{P}_t\}_{t=1}^T$ *are deterministic; then the regret for Algorithm 1 satisfies*

$$\lim_{T \to \infty} \frac{R_T(TUCB)}{T} = 0$$

Algorithm 1 TreeUCB (TUCB)

Parameter: $M \geq 0$ ($M \geq L$). $C > 0$. Tree fitting rule $\mathcal{R}$ that satisfies 1–4 in Theorem 1.
▷ C depends on the a.s. bound of the reward.
▷ M depends on the Lipschitz constant of the expected reward.
for $t = 1, 2, \ldots, T$ **do**
  Fit the tree f_{t-1} using rule $\mathcal{R}$ on observations $(a_1, y_1, a_2, y_2, \ldots, a_{t-1}, y_{t-1})$.
  With respect to the partition $\mathcal{P}_{t-1}$ defined by leaves of f_{t-1}, define m_{t-1}, n_{t-1} as in (1) and (2). Play

$$a_t \in \arg\max_{a \in \mathcal{A}} \{U_t(a)\}, \tag{4}$$

  where U_t is defined in (3). Ties are broken uniformly at random.
  Observe the reward y_t.

with probability 1.

The above assumptions are all mild and reasonable. For item 1, we can use incremental tree learning [28] to enforce nested partitions. For item 2, we may put a cap (that may depend on t) on the depth of the tree to constrain it. For item 3, we may put a cap (that may depend on t) on tree leaf diameters to ensure it. For item 4, any non-random tree learning rule meets this criteria, since in this case, the randomness only comes from the data (and/or number of data points observed).

We now discuss the proof of Theorem 1. Throughout the rest of the paper, we use $\widetilde{O}$ to omit constants and poly-log terms unless otherwise noted. To prove Theorem 1, we first use Claims 1 and 2 to bound the single step regret, we then use Lemma 1 and Assumptions (1) – (3) to bound the total regret.

To start with, we first present the following two claims, which may also be carefully extracted from previous works [e.g. 9].

Claim 1. *For an arbitrary arm* a, *and time* t, *with probability at least* $1 - \frac{1}{t^4}$, *we have,*

$$|m_{t-1}(a) - f(a)| \leq L \cdot D(p_{t-1}(a)) + C\sqrt{\frac{4\log t}{n_{t-1}(a)}}$$

for a constant C *that depends only on the a.s. bound of the reward.*

Claim 2. *At any* t, *with probability at least* $1 - \frac{1}{t^4}$, *the single step regret satisfies:*

$$f(a^*) - f(a_t) \leq 2L \cdot D(p_{t-1}(a_t)) + 2C\sqrt{\frac{4\log t}{n_{t-1}(a_t)}} \tag{5}$$

for a constant C, *that depends only on the a.s. bound of the reward.*

In Section 2.2, we prove general versions of Claims 1 and 2.

As the tree (partition) grows finer, the term $n_{t-1}(a)$ is not necessarily increasing with t (for an arbitrary fixed a). Therefore part of the difficulty is in bounding $\sum_{t=1}^T \frac{1}{n_{t-1}(a_t)}$. Next, we introduce a new set of inequalities, which we call "point scattering" inequalities in Lemma 1 to bound this term.

Lemma 1 (Point Scattering Inequalities). *For an arbitrary sequence of points* $a_1, a_2, \cdots$ *in a space* $\mathcal{A}$, *and any sequence of nested*

partitions $\mathcal{P}_1, \mathcal{P}_2, \cdots$ of the same space $\mathcal{A}$, we have, for any T,

$$\sum_{t=1}^{T} \frac{1}{n_{t-1}(a_t)} \le e|\mathcal{P}_T| \log\left(1 + (e-1)\frac{T}{|\mathcal{P}_T|}\right), \tag{6}$$

$$\sum_{t=1}^{T} \frac{1}{1 + n^0_{t-1}(a_t)} \le |\mathcal{P}_T| \left(1 + \log \frac{T}{|\mathcal{P}_T|}\right), \tag{7}$$

$$\sum_{t=1}^{T} \left(\frac{1}{1 + n^0_{t-1}(a_t)}\right)^{\alpha} \le \frac{1}{1-\alpha} |\mathcal{P}_T|^{\alpha} T^{1-\alpha}, \ 0 < \alpha < 1, \tag{8}$$

where n^0_{t-1} and n_{t-1} are the count and corrected count function as in Definition 2, and $|\mathcal{P}_T|$ is the cardinality of the finite partition $\mathcal{P}_T$.

As defined in Definition 2, $n^0_{t-1}(a_t)$ is the number of points that are in the same bin (in partition $\mathcal{P}_{t-1}$) as a_t. Also, $n_{t-1}(a_t)$ is the "corrected" version of $n^0_{t-1}(a_t)$: $n_{t-1}(a_t) = \max(1, n^0_{t-1}(a_t))$.

Remark 2. *We shall notice that (6) allows us to somewhat "look one step ahead of time", since it uses the values $\{n_{t-1}(a_t)\}_t$ - the corrected counts without including a_t. This is because n_{t-1} is computed using points up to time $t-1$. The equation (7) is different from (6) in the sense that $\{1 + n^0_{t-1}(a_t)\}_t$ are essentially the counts including a_t. While, with proper modification, both (6) and (7) can be used to derive Theorem 1, we shall not ignore the difference between (6) and (7).*

2.1.1 Proof of (6). We use a novel constructive trick to derive (6). This trick and the usefulness of the result (Remarks 2 and 3 and Section 2.3) mark our major technical contribution. The trick is to consider the incidence matrix of which points are within the same partition bin, and use this matrix as if it were a covariance matrix for a Gaussian process. Then, we use knowledge about Gaussian processes to bound the sum of the inverse of the number of points in each bin over time.

For each T, we construct a hypothetical noisy degenerate Gaussian process. We are not assuming our payoffs are drawn from these Gaussian processes. We only use these Gaussian processes as a proof tool. To construct these noisy degenerate Gaussian processes, we define the kernel functions $k_T : \mathcal{A} \times \mathcal{A} \to \mathbb{R}$,

$$k_T(a, a') = \begin{cases} 1, & \text{if } p_T(a) = p_T(a') \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

where p_T is the region selection function defined with respect to $\mathcal{P}_T$. The kernel k_T is positive semi-definite as shown in Proposition 1.

Proposition 1. *The kernel defined in (9) is positive semi-definite for any $T \ge 1$.*

Proof. For any $x_1, \ldots, x_n$ in where the kernel $k_T(\cdot,\cdot)$ is defined, the Gram matrix $K = \left[k_T(x_i, x_j)\right]_{n\times n}$ can be written into block diagonal form where diagonal blocks are all-one matrices and off-diagonal blocks are all zeros with proper permutations of rows and columns. Thus without loss of generality, for any vector $\boldsymbol{v} = [v_1, v_2, \ldots, v_n] \in \mathbb{R}^n$, $\boldsymbol{v}^\top K \boldsymbol{v} = \sum_{b=1}^{B} \left(\sum_{j: i_j \text{ in block } b} v_{i_j}\right)^2 \ge 0$ where the first summation is taken over all diagonal blocks and B is the total number of diagonal blocks in the Gram matrix. □

Now, at any time T, let us consider the model $\tilde{y}(a) = g(a) + e_T$ where g is drawn from a Gaussian process $g \sim \mathcal{GP}(0, k_T(\cdot,\cdot))$ and $e_T \sim \mathcal{N}(0, s_T^2)$. Suppose that the arms and hypothetical payoffs $\{(a_1, \tilde{y}_1), (a_2, \tilde{y}_2), \ldots, (a_t, \tilde{y}_t)\}$ are observed from this Gaussian process. The posterior variance for this Gaussian process after the observations at $a_1, a_2, \ldots, a_t$ is

$$\sigma^2_{T,t}(a) = k_T(a,a) - \boldsymbol{k}_a^T (K + s_T^2 I)^{-1} \boldsymbol{k}_a,$$

where $\boldsymbol{k}_a = [k_T(a,a_1), \ldots, k_T(a,a_t)]^\top$, $K = [k_T(a_i,a_j)]_{t\times t}$ and I is the identity matrix. In other words, $\sigma^2_{T,t}(a)$ is the posterior variance using points up to time t with the kernel defined by the partition at time T. After some matrix manipulation, we know that

$$\sigma^2_{T,t}(a) = 1 - \mathbf{1}_a [\mathbf{1}_a \mathbf{1}_a^\top + s_T^2 I]^{-1} \mathbf{1}_a,$$

where $\mathbf{1}_a = [1, \cdots, 1]^\top_{1\times n^0_{T,t}(a)}$. By the Sherman-Morrison formula, $[\mathbf{1}_a \mathbf{1}_a^\top + s_T^2 I]^{-1} = s_T^{-2} I - \frac{s_T^{-4} \mathbf{1}_a \mathbf{1}_a^\top}{1 + s_T^{-2} n^0_{T,t}(a)}$. Thus the posterior variance is

$$\sigma^2_{T,t}(a) = \frac{1}{1 + s_T^{-2} n^0_{T,t}(a)}. \tag{10}$$

Following the arguments in [25], we derive the following results. For any $t \le T$, and an arbitrary sequence $\boldsymbol{a}_t = \{a_1, a_2, \cdots, a_t\}$, we consider fixing this sequence and query the constructed Gaussian processes at these points. Since $\boldsymbol{a}_t$ is fixed, the entropy $H(\tilde{\boldsymbol{y}}_t, \boldsymbol{a}_t) = H(\tilde{\boldsymbol{y}}_t)$. Since, by definition of a Gaussian process, $\tilde{\boldsymbol{y}}_t$ follows a multivariate Gaussian distribution,

$$H(\tilde{\boldsymbol{y}}_t) = \frac{1}{2} \log\left[(2\pi e)^t \det\left(K + s_T^2 I\right)\right] \tag{11}$$

where $K = \left[k_T(a_i,a_j)\right]_{t\times t}$. We can then compute $H(\tilde{\boldsymbol{y}}_t)$ by

$$\begin{aligned} H(\tilde{\boldsymbol{y}}_t) &= H(\tilde{y}_t | \tilde{\boldsymbol{y}}_{t-1}) + H(\tilde{\boldsymbol{y}}_{t-1}) \\ &= H(\tilde{y}_t | a_t, \tilde{\boldsymbol{y}}_{t-1}, \boldsymbol{a}_{t-1}) + H(\tilde{\boldsymbol{y}}_{t-1}) \\ &= \frac{1}{2} \log\left(2\pi e \left(s_T^2 + \sigma^2_{T,t-1}(a_t)\right)\right) + H(\tilde{\boldsymbol{y}}_{t-1}) \\ &= \frac{1}{2} \sum_{\tau=1}^{t} \log\left(2\pi e \left(s_T^2 + \sigma^2_{T,\tau-1}(a_\tau)\right)\right), \end{aligned} \tag{12}$$

where (12) comes from recursively expanding $H(\tilde{\boldsymbol{y}}_\tau)$. By (11) and (12),

$$\sum_{\tau=1}^{t} \log\left(1 + s^{-2}\sigma^2_{T,\tau-1}(a_\tau)\right) = \log\left[\det\left(s^{-2}K + I\right)\right]. \tag{13}$$

For the block diagonal matrix K of size $t \times t$, let n_i denote the size of block i and B' ($B' \le |\mathcal{P}_t|$) be the total number of diagonal blocks up to a time t ($t \le T$). Then we have

$$\begin{aligned} \det\left(s^{-2}K + I\right) &= \prod_{i=1}^{B'} \det\left(s^{-2}\mathbf{1}\mathbf{1}^\top + I_{n_i\times n_i}\right) \\ &= \prod_{i=1}^{B'} \left(1 + s^{-2} n_i\right) \le \left(1 + \frac{s^{-2}t}{B'}\right)^{B'}, \end{aligned}$$

where $\mathbf{1}$ is all-1 vector of proper length. In the above, (1) the equality on the first line uses the determinant of block-diagonal matrix equals to the product of determinant of diagonal blocks, 2) the equality on the last line is due to the matrix determinant lemma, and 3) the inequality on the last line is due to the AM-GM inequality and that $\sum_{i=1}^{B'} n_i = t$.

Next, since $|\mathcal{P}_t| \geq B'$ and $\left(1 + \frac{s^{-2}t}{x}\right)^x$ is increasing with x (on $[1, \infty)$),

$$\det\left(s^{-2}K + I\right) \leq \left(1 + \frac{s^{-2}t}{B'}\right)^{B'} \leq \left(1 + \frac{s^{-2}t}{|\mathcal{P}_t|}\right)^{|\mathcal{P}_t|}. \tag{14}$$

Therefore, from (13) and (14),

$$\sum_{\tau=1}^{T} \log\left(1 + s^{-2}\sigma^2_{T,\tau-1}(a_\tau)\right) \leq |\mathcal{P}_T| \log\left(1 + \frac{s^{-2}T}{|\mathcal{P}_T|}\right), \tag{15}$$

since arguments after (11) hold for all $t \leq T$.

Since the function $h(\lambda) = \frac{\lambda}{\log(1+\lambda)}$ is increasing for non-negative λ, $\lambda \leq \frac{s_T^{-2}}{\log(1+s_T^{-2})} \log(1+\lambda)$ for $\lambda \in [0, s_T^{-2}]$. Since $\sigma_{T,t}(a) \in [0,1]$ for all a,

$$\sigma^2_{T,t}(a) \leq \frac{1}{\log(1+s_T^{-2})} \log\left(1 + s_T^{-2}\sigma^2_{T,t}(a)\right) \tag{16}$$

for $t, T = 0, 1, 2, \cdots$. Since the partitions are nested, we have that for $T_1 \leq T_2$, $n_{T_1,t}(a) \geq n_{T_2,t}(a)$, and thus $\sigma^2_{T_1,t}(a) \leq \sigma^2_{T_2,t}(a)$. Suppose we query at points $a_1, \cdots, a_T$ in the Gaussian process $\mathcal{GP}(0, k_T(\cdot,\cdot))$. Then,

$$\begin{aligned}
&\sum_{t=1}^{T} \frac{1}{n_{t-1}(a_t)} \leq \sum_{t=1}^{T} \frac{1+s_T^{-2}}{1+s_T^{-2}n_{t-1}(a_t)} \\
&\leq \sum_{t=1}^{T} \frac{1+s_T^{-2}}{1+s_T^{-2}n^0_{T,t-1}(a_t)} \leq \left(1+s_T^{-2}\right)\sum_{t=1}^{T}\sigma^2_{T,t-1}(a_t) \qquad (17)\\
&\leq \frac{1+s_T^{-2}}{\log(1+s_T^{-2})}\sum_{t=1}^{T}\log\left(1+s_T^{-2}\sigma^2_{T,t-1}(a_t)\right) \\
&\leq \frac{1+s_T^{-2}}{\log(1+s_T^{-2})}|\mathcal{P}_T|\log\left(1+s_T^{-2}\frac{T}{|\mathcal{P}_T|}\right),
\end{aligned}$$

where (17) uses (10), the second last inequality uses (16), and the last inequality uses (15). Finally, we optimize over s_T. Since $s_T^{-2} = e - 1$ minimizes $\frac{1+s_T^{-2}}{\log(1+s_T^{-2})}$, we have

$$\sum_{t=1}^{T} \frac{1}{n_{t-1}(a_t)} \leq e|\mathcal{P}_T| \log\left(1 + (e-1)\frac{T}{|\mathcal{P}_T|}\right).$$

The above argument proves (6).

Remark 3. *One important insight of our analysis is that this allows us to link the Hoeffding-type concentration term to the posterior variance of the constructed Gaussian processes. This connection is directly shown in (10). As we will discuss in Section 2.3.2, we can use this connection to improve the entire learning process via "softening".*

Next, we sketch the proofs of (7) and (8).

Proof of (7). Consider the partition $\mathcal{P}_T$ at time T. We label the regions of the partitions by $j = 1, 2, \cdots, |\mathcal{P}_T|$. Let $t_{j,i}$ be the time when the i-th point in the j-th region in $\mathcal{P}_T$ being selected. Let b_j be the number of points in region j. Since the partitions are nested, we have $1 + n^0_{t_{j,i}-1}(x_{t_{j,i}}) \geq i$ for all i, j. We have, for $T \geq 1$,

$$\begin{aligned}
\sum_{t=1}^{T} \frac{1}{1+n^0_{t-1}(x_t)} &= \sum_{j=1}^{|\mathcal{P}_T|}\sum_{i=1}^{b_j} \frac{1}{1+n^0_{t_{j,i}-1}(x^0_{t_{j,i}})} \leq \sum_{j=1}^{|\mathcal{P}_T|}\sum_{i=1}^{b_j}\frac{1}{i} \qquad (18)\\
&\leq \sum_{j=1}^{|\mathcal{P}_T|}\left(1+\log b_j\right) = |\mathcal{P}_T| + \sum_{j=1}^{|\mathcal{P}_T|}\log b_j \\
&= |\mathcal{P}_T| + \log\prod_{j=1}^{|\mathcal{P}_T|} b_j \leq |\mathcal{P}_T| + |\mathcal{P}_T|\log\frac{T}{|\mathcal{P}_T|}, \qquad (19)
\end{aligned}$$

where (18) uses $1+n^0_{t_{j,i}-1}(x_{t_{j,i}}) \geq i$ and (19) uses AM-GM inequality and that $\sum_{j=1}^{|\mathcal{P}_T|} b_j = T$.

Proof of (8). The idea is similar to that of (7). For $0 < \alpha < 1$,

$$\begin{aligned}
\sum_{t=1}^{T}\left(\frac{1}{1+n^0_{t-1}(x_t)}\right)^\alpha &= \sum_{j=1}^{|\mathcal{P}_T|}\sum_{i=1}^{b_j}\left(\frac{1}{1+n^0_{t_{j,i}}(x^0_{t_{j,i}})}\right)^\alpha \\
&\leq \sum_{j=1}^{|\mathcal{P}_T|}\sum_{i=1}^{b_j}\frac{1}{i^\alpha} \leq \sum_{j=1}^{|\mathcal{P}_T|}\frac{1}{1-\alpha}b_j^{1-\alpha} \\
&\leq \frac{1}{1-\alpha}|\mathcal{P}_T|^\alpha T^{1-\alpha}, \qquad (20)
\end{aligned}$$

where (20) is due to the Hölder's inequality and that $\sum_{j=1}^{|\mathcal{P}_T|} b_j = T$.

Proof of Theorem 1. Now we are ready to prove Theorem 1. We can split the sum of regrets by

$$\sum_{t=1}^{T}\left(f(a^*) - f(a_t)\right) = \sum_{t=1}^{\left\lfloor\sqrt{T}\right\rfloor}\left(f(a^*) - f(a_t)\right) + \sum_{\left\lfloor\sqrt{T}\right\rfloor+1}^{T}\left(f(a^*) - f(a_t)\right).$$

Also, by Claim 2, with probability at least $1 - \frac{1}{3\left\lfloor\sqrt{T}\right\rfloor^3}$, (5) holds simultaneously for all $t = \left\lfloor\sqrt{T}\right\rfloor + 1, \cdots, T$ ($T \geq 2$). Thus for $T \geq 2$, the event

$$E_T = \left\{\frac{R_T}{T} > \frac{1}{T}\left(\sqrt{T} + \sum_{t=\left\lfloor\sqrt{T}\right\rfloor+1}^{T} B_t\right)\right\}, \quad \textit{where}$$

$$B_t := \left(2L \cdot D(p_{t-1}(a_t)) + 2C\sqrt{\frac{4\log t}{n_{t-1}(a_t)}}\right)$$

occurs with probability at most $\frac{1}{3\left\lfloor\sqrt{T}\right\rfloor^3}$. Since $\frac{1}{3\left\lfloor\sqrt{T}\right\rfloor^3} \sim \frac{1}{3T^{3/2}}$, we know $\sum_{T=2}^{\infty}\mathbb{P}(E_T) < \infty$. By the Borel-Cantelli lemma, we know $\mathbb{P}\left(\limsup_{T\to\infty} E_T\right) = 0$. In other words, with probability 1, E_T occurs finitely many times. Thus, with probability 1, there exists a constant T_0, such that the event $\overline{E}_T$ (negation of E_T) occurs for all $T > T_0$. Also, from the Cauchy-Schwarz inequality (used below in

the second line) and (6) (used below in the last line), we know that

$$\sum_{t=\left\lfloor\sqrt{T}\right\rfloor+1}^{T}\sqrt{\frac{\log t}{n_{t-1}(a_t)}} \le \sum_{t=1}^{T}\sqrt{\frac{\log t}{n_{t-1}(a_t)}} \le \sqrt{T\log T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_{t-1}(a_t)}}$$

$$\le \sqrt{T\log T}\sqrt{e|\mathcal{P}_T|\log\left(1+(e-1)\frac{T}{|\mathcal{P}_T|}\right)} = \widetilde{O}\left(T^{\frac{1+\gamma}{2}}\right),$$

where the last equality is from the assumption that $|\mathcal{P}_t| = o(t^\gamma)$ for some $\gamma < 1$. This means

$$\lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sqrt{\frac{4\log t}{n_{t-1}(a_t)}} = 0.$$

In addition, by the assumption that $D(p_t(a)) = o(1)$, we know $\limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T} D(p_{t-1}(a)) = 0$. The above two limits give us

$$\lim_{T\to\infty}\frac{1}{T}\left(\sqrt{T} + \sum_{\left\lfloor\sqrt{T}\right\rfloor+1}^{T} B_t\right) = 0, \quad \textit{where} \tag{21}$$

$$B_t := \left(2L\cdot D(p_{t-1}(a_t)) + 2C\sqrt{\frac{4\log t}{n_{t-1}(a_t)}}\right). \tag{22}$$

Combining all the facts above, we have $\lim_{T\to\infty}\frac{R_T}{T} = 0$ with probability 1.

Adaptive partitioning: TUCB shall be implemented using regression trees or incremental regression trees. This naturally leverages the practical advantages of regression trees. Leaves in a regression tree form a partition of the space. Also, a regression tree is designed to fit an underlying function. This leads to an adaptive partitioning where the underlying function values within each region should be relatively similar to each other. We defer the discussion on the implementation we use in our experiments to Section 3. Please refer to [7] for more details about regression tree fitting.

2.2 The Contextual TreeUCB algorithm

Algorithm 2 Contextual TreeUCB (CTUCB)

1: Parameter: $M > 0$, $C > 0$, and tree fitting rule $\mathcal{R}$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Observe context z_t.
4: Fit a regression tree f_{t-1} (using rule $\mathcal{R}$) on observations $\{(z_t, a_t), y_t\}_{t=1}^{T}$, .
5: With respect to the partition $\mathcal{P}_{t-1}$ defined by leaves of f_{t-1}, define m_{t-1} and n_{t-1} in (1) and (2) (over the joint space $\mathcal{Z}\times\mathcal{A}$). Play

$$a_t \in \arg\max_{a\in\mathcal{A}}\{U_t((z_t, a))\},$$

where $U_t(\cdot)$ is defined in (3). Ties are broken at random.
6: Observe the reward y_t.

In this section, we present an extension of Algorithm 1 for the contextual stochastic bandit problem. The contextual stochastic bandit problem is an extension to the stochastic bandit problem. In this problem, at each time, context information is revealed, and the agent chooses an arm based on past experience as well as the contextual information. Formally, the expected payoff function f is defined over the product of the context space $\mathcal{Z}$ and the arm space $\mathcal{A}$ and takes values from $[0, 1]$. Similar to the previous discussions, compactness of the product space and Lipschitzness of the payoff function are assumed. In addition, a mean zero, almost surely bounded noise that is independent of the expected reward function is added to the observed rewards. At each time t, a contextual vector $z_t \in \mathcal{Z}$ is revealed and the agent plays an arm $a_t \in \mathcal{A}$. The performance of the agent following algorithm `Alg` is measured by the cumulative contextual regret

$$R_T^c(\texttt{Alg}) = \sum_{t=1}^{T} f(z_t, a_t^*) - f(z_t, a_t), \tag{23}$$

where $f(z_t, a_t^*)$ is the maximal value of f given contextual information z_t. A simple extension of Algorithm 1 can solve the contextual version problem. *In particular, in the contextual case, we partition the joint space $\mathcal{Z}\times\mathcal{A}$ instead of the arm space $\mathcal{A}$.* As an analog to (2) and (1), we define the corrected count n_t and the corrected average m_t over the joint space $\mathcal{Z}\times\mathcal{A}$ with respect to the partition $\mathcal{P}_t$ of the joint space $\mathcal{Z}\times\mathcal{A}$, and observations in the joint space $((z_1, a_1), y_1, \cdots, (z_t, a_t), y_t)$. The guarantee of Algorithm 2 is in Theorem 2.

Theorem 2. *Suppose that the payoff function f defined on a compact doubling metric space $(\mathcal{Z}\times\mathcal{A}, d)$ satisfies $f(z, a) \in [0, 1]$ for all (z, a) and is Lipschitz. If the tree growing rule satisfies requirements 1-4 listed in Theorem 1, then $\lim_{T\to\infty}\frac{R_T^c(CTUCB)}{T} = 0$ with probability 1.*

Theorem 2 follows from Theorem 1. Since the point scattering inequality holds for any sequence of (context-)arms, we can replace regret with contextual regret and alter Claims 1 and 2 accordingly to prove Theorem 2.

In particular, Claims 1 and 2 extend to the contextual setting, as stated and proved below.

Claim 3. *For any context z, arm a, and time t, with probability at most $\frac{1}{t^4}$, we have:*

$$|m_{t-1}(z, a) - f(z, a)| > L\cdot D(p_{t-1}(z, a)) + C\sqrt{\frac{4\log t}{n_{t-1}(z, a)}} \tag{24}$$

for a constant C.

Proof. First of all, when $t = 1$, this is trivially true by Lipschitzness. Now let us consider the case when $t \ge 2$. Let us use $A_1, A_2, \cdots, A_t$ to denote the random variables of arms selected up to time t, $Z_1, Z_2, \cdots, Z_t$ to denote the random context up to time t and $Y_1, Y_2, \cdots, Y_t$ to denote random variables of rewards received up to time t. Then the random variables $\left\{\sum_{t=1}^{T}(f(Z_t, A_t) - Y_t)\right\}$ is a martingale sequence. This is easy to verify since the noise is mean zero and independent. In addition, since there is no randomness in the partition formation (given a sequence of observations), for a fixed a, we have the times $\mathbb{I}[(Z_t, A_i) \in p_{t-1}(z, a)]$ $(i \le t)$ is measureable with respect to $\sigma(Z_1, A_1, Y_1, \cdots, Z_t, A_t, Y_t)$. Therefore, the sequence $\left\{\sum_{i=1}^{t}(f(Z_i, A_i) - Y_i)\mathbb{I}[(Z_i, A_i) \in p_{t-1}(z, a)]\right\}_{t=1}^{T}$ is a skipped martingale. Since skipped martingale is also a martingale,

we apply the Azuma-Hoeffding inequality (with sub-Gaussian tails) [23]. For simplicity, we write

$$B_t(z,a) := C\sqrt{\frac{4\log t}{n_{t-1}(z,a)}} + L\cdot D(p_{t-1}(z,a)), \tag{25}$$

$$\mathcal{E}_t^i(z,a) := (Z_i, A_i) \in p_{t-1}(z,a). \tag{26}$$

Combining this with Lipschitzness, we get there is a constant C (depends on the a.s. bound of the reward, as a result of Hoeffding inequality), such that

$$\begin{aligned}
&\mathbb{P}\left\{|m_{t-1}(z,a) - f(z,a)| > B_t(z,a)\right\} \\
\le\ &\mathbb{P}\left\{\left|\frac{1}{n_{t-1}(z,a)}\sum_{i=1}^{t-1}(f(Z_i,A_i)-Y_i)\mathbb{I}[\mathcal{E}_t^i(z,a)]\right|\right. \\
&+\left|f(z,a) - \frac{1}{n_{t-1}(z,a)}\sum_{i=1}^{t-1} f(Z_i,A_i)\mathbb{I}[\mathcal{E}_t^i(z,a)]\right| \\
&\left.> C\sqrt{\frac{4\log t}{n_{t-1}(z,a)}} + L\cdot D(p_{t-1}(z,a))\right\} \le \frac{1}{t^4},
\end{aligned} \tag{27}$$

where (27) uses both the Lipschitzness and the Azuma-Hoeffding's inequality. □

Claim 4. *At any t, with probability at least $1-\frac{1}{t^4}$, the single step contextual regret satisfies:*

$$f(z_t,a_t^*) - f(z_t,a_t) \le 2L\cdot D(p_{t-1}(z_t,a_t)) + 2C\sqrt{\frac{4\log t}{n_{t-1}(z_t,a_t)}}$$

for a constant C. Here a_t^* is the optimal arm for the context z_t.

Proof. By Claim 3, with probability at least $1-\frac{1}{t^4}$, the following ((28) and (29)) hold simultaneously,

$$\begin{aligned}
&m_{t-1}(z_t,a_t) + C\sqrt{\frac{4\log t}{n_{t-1}(z_t,a_t)}} + L\cdot D(p_{t-1}(z_t,a_t)) \\
\ge\ &m_{t-1}(z_t,a_t^*) + \sqrt{\frac{4\log t}{n_{t-1}(z_t,a_t^*)}} + L\cdot D(p_{t-1}(z_t,a_t^*)) \\
\ge\ &f(z_t,a_t^*),
\end{aligned} \tag{28}$$

$$f(z_t,a_t) \ge m_{t-1}(z_t,a_t) - C\sqrt{\frac{4\log t}{n_{t-1}(z_t,a_t)}} - L\cdot D(p_{t-1}(z_t,a_t)). \tag{29}$$

This is true since we first take a one-sided version of Hoeffding-type tail bound in (24), and then take a union bound over the two points (z_t,a_t) and (z_t,a_t^*). This first halves the probability bound and then doubles it. Then we take the complementary event to get (28) and (29) simultaneously hold with probability at least $1-\frac{1}{t^4}$. We then take another union bound over time t, as discussed in the main text. Note that throughout the proof, we do not need to take union bounds over all arms or all regions in the partition.

(a) CNN architecture for SVHN. A value with * means that this parameter is tuned, and the batch-normalization layer uses all Tensorflow's default settings.

Layer	Hyperparameters	values
Conv1	conv1-kernel-size	*
	conv1-number-of-channels	200
	conv1-stride-size	(1,1)
MaxPooling1	pooling1-size	(3,3)
	pooling1-stride	(1,1)
Conv2	conv2-kernel-size	*
	conv2-number-of-channels	200
	conv2-stride-size	(1,1)
MaxPooling2	pooling2-size	(3,3)
	pooling2-stride	(2,2)
Conv3	conv3-kernel-size	(3,3)
	conv3-number-of-channels	200
	conv3-stride-size	(1,1)
AvgPooling3	pooling3-size	(3,3)
	pooling3-stride	(1,1)
Dense	batch-normalization	default
	number-of-hidden-units	512
	dropout-rate	0.5

(b) Hyperparameter search space. β_1 and β_2 are parameters for the AdamOptimizer [14]. The learning rate is discretized in the following way: from 1e-6 to is 1 (including the end points), we log-space the learning rate into 50 points, and from 1.08 to 5 (including the end points) we linear-space the learning rate into 49 points.

Hyperparameters	Range
conv1-kernel-size	$\{1,2,\cdots,7\}$
conv2-kernel-size	$\{1,2,\cdots,7\}$
β_1 & β_2	$\{0, 0.05, \cdots, 1\}$
learning-rate	1e-6 to 5
training-iteration	$\{300, 400, \cdots, 1500\}$

Table 1: Settings for the SVHN experiments.

Equation 28 holds by algorithm definition. Otherwise we will not select a_t at time t. Combine (28) and (29), and we get

$$\begin{aligned}
&f(z_t,a_t^*) - f(z_t,a_t) \\
=\ &f(z_t,a_t^*) - m_{t-1}(z_t,a_t) + m_{t-1}(z_t,a_t) - f(z_t,a_t) \\
\le\ &2C\sqrt{\frac{4\log t}{n_{t-1}(z_t,a_t)}} + 2L\cdot D(p_{t-1}(z_t,a_t)).
\end{aligned}$$

□

2.3 Use Cases of Point Scattering Inequalities

2.3.1 Recover Previous Bounds. In this section, we give examples of using the point scattering inequalities to derive regret bounds for other algorithms. For our purpose of illustrating the point scattering inequalities, the discussed algorithms are simplified. We also assume that the reward and the sub-Gaussianity are properly scaled so that the parameter before the Hoeffding-type concentration term is 1.

(a) CNN architecture for CIFAR-10. A value with * means that this parameter is tuned, and the batch-normalization layer uses all Tensorflow's default setting.

Layer	Hyperparameters	values
Conv1	conv1-kernel-size	*
	conv1-no.-of-channels	200
	conv1-stride-size	(1,1)
MaxPooling1	pooling1-size	*
	pooling1-stride	(1,1)
Conv2	conv2-kernel-size	*
	conv2-no.-of-channels	200
	conv2-stride-size	(1,1)
MaxPooling2	pooling2-size	*
	pooling2-stride	(2,2)
Conv3	conv3-kernel-size	*
	conv3-no.-of-channels	200
	conv3-stride-size	(1,1)
AvgPooling3	pooling3-size	*
	pooling3-stride	(1,1)
	pooling3-padding	"same"
Dense	batch-normalization	default
	no.-of-hidden-units	512
	dropout-rate	0.5

(b) Hyperparameter search space. β_1 and β_2 are parameters for the Adamoptimizer. The learning rate is discretized in the following way: from 1e-6 to is 1 (including the end points), we log-space the learning rate into 50 points, and from 1.08 to 5 (including the end points) we linear-space the learning rate into 49 points. The learning-rate-reduction parameter is how many times the learning rate is going to be reduced by a factor of 10. For example, if the total training iteration is 200, the learning-rate is 1e-6, and the learning-rate-reduction is 1, then for the first 100 iteration the learning rate is 1e-6, and the for last 100 iterations the learning rate is 1e-7.

Hyperparameters	Range
conv1-kernel-size	$\{1, 2, \cdots, 7\}$
conv2-kernel-size	$\{1, 2, \cdots, 7\}$
conv3-kernel-size	$\{1, 2, 3\}$
pooling1-size & pooling2-size	$\{1, 2, 3\}$
pooling3-size	$\{1, 2, \cdots, 6\}$
β_1 & β_2	$\{0, 0.05, \cdots, 1\}$
learning-rate	1e-6 to 5
learning-rate-redeuction	{1,2,3}
training-iteration	$\{200, 400, \cdots, 3000\}$

Table 2: Settings for CIFAR-10 experiments.

The UCB1 algorithm The classic UCB1 algorithm [5] assumes a finite set of arms, each having a different reward distribution. Following our notation, at time t, the UCB1 algorithm plays

$$a_t \in \arg\max_a \left\{ m_{t-1}(a) + \sqrt{\frac{2\log T}{n_{t-1}(a)}} \right\}. \tag{30}$$

Indeed, this equation can be interpreted as (4) under the discrete 0-1 metric: two points are distance zero if they coincide and distance 1 otherwise. Then from the point scattering inequality (6), we get for UCB1

$$\begin{aligned}\mathbb{E}[R_T(UCB1)] &= O\left(\sum_{t=1}^{T}\sqrt{\frac{\log T}{n_{t-1}(a_t)}}\right) \\ &= O\left(\sqrt{T\log T}\sqrt{\sum_{t=1}^{T}\frac{1}{n_{t-1}(a_t)}}\right) = \widetilde{O}\left(\sqrt{K\cdot T}\right),\end{aligned}$$

where K is number of arms in the problem. This matches the gap-independent (independent of the reward gap between an arm and the optimal arm) bound derived using traditional methods in UCB1 algorithm [5, 8]. In this analysis, we apply the point scattering inequality with the partition $\mathcal{P}_t$ being the set of arms at all t.

Finite Time Bound for Lipschitz Bandits and Lipschitz RL. As shown in Claim 2, the single step regret is bounded by a Hoeffding-type concentration and the diameter of selected region (due to Lipschitzness). Since the point scattering inequalities provide a bound of the overall summation of the Hoeffding terms, we can design and analyze many partition-based Lipschitz algorithms using point scattering inequalities We can do this since the partitioning is up to our choice. Examples include the `UniformMesh` algorithm discussed by [15], and parition-based Lipschitz reinforcement learning algorithm recently studied [e.g. 21].

2.3.2 Hierarchical Bayesian Method for Lipschitz Bandits. Existing Lipschitz bandit algorithms [e.g., 15] partition the arm space into disjoint bins. Based on this partition, arms in two different bin do not give information about each other, and all arms within the same bins are viewed as the same. This implicit assumption, however, is obviously untrue. On the other hand, imposing a strong prior on the reward function would break the Lipschitzness assumption. To simultaneously address the above two difficulties, we link the learned tree (or partition) to a Bayesian model in light of our analysis of (6). This new viewpoint allows us to "soften" the entire model using a hierarchical Bayesian method.

Formally, at each time t, we consider the following hierarchical Bayesian problem with respect to the learned partition $\mathcal{P}_t$. Note that this hierarchical Bayesian model is updated whenever we update the partition. This is roughly the same as make a finite partition and treat each bin as an arm, and do not impose extra structures on the reward function. Let $\mathcal{P}_t$ be the learnt partition such that each bin is a rectangle. Then the kernel function is defined as

$$\widetilde{k}_T(\cdot,\cdot) = \sum_{p\in\mathcal{P}_T} \widetilde{k}_T^{(p)}(\cdot,\cdot), \tag{31}$$

where p are regions in $\mathcal{P}_T$, and $\widetilde{k}_T^{(p)}(\cdot,\cdot)$ is defined as follows. For a partition $p = \prod_{i=1}^{d}[a_i, b_i]$, define

$$\widetilde{k}_T^{(p)}(\cdot,\cdot) = \prod_{i=1}^{d} \widetilde{k}_T^{(p,i)}(\cdot,\cdot), \quad \text{where} \tag{32}$$

$$\widetilde{k}_T^{(p,i)}(\boldsymbol{x},\boldsymbol{x}') = \left[1 + \exp\left(-\alpha_T\left(\Delta_i - \frac{b_i - a_i}{2}\right)\right)\right]^{-1}, \tag{33}$$

$$\Delta_i = \max\left\{\left|\boldsymbol{x}_i - \frac{a_i+b_i}{2}\right|, \left|\boldsymbol{x}_i' - \frac{a_i+b_i}{2}\right|\right\}, \tag{34}$$

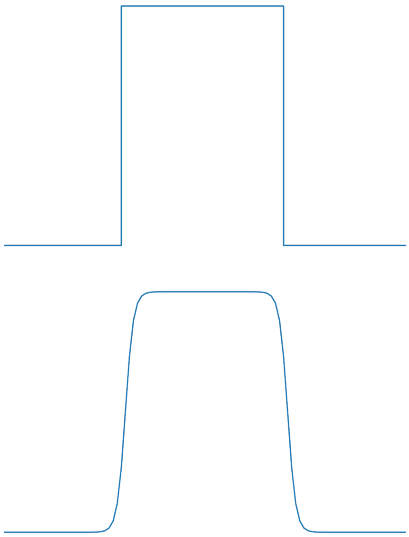


Figure 2: The left subfigure is the metric learned by Algorithm 1 (9). The right subfigure is the smoothed version of this learned metric.

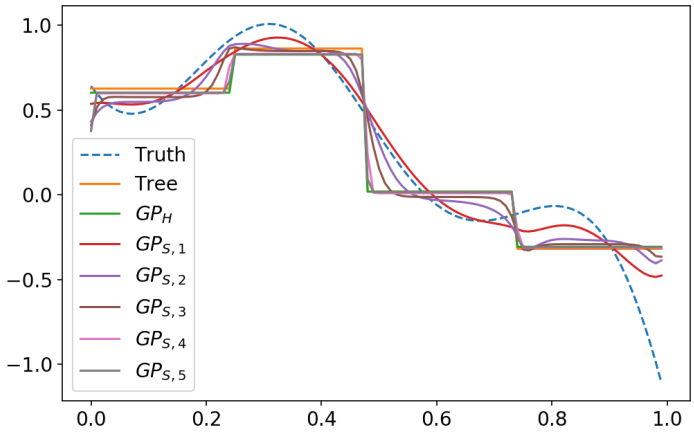


Figure 3: The estimates for a function with respect to a given partition. The "Tree" line is directly averaging within each partition. The "GP_H" line is the learned posterior GP mean function using the "hard metric." The lines "$GP_{S,1}$ - $GP_{S,5}$" are 5 learned posterior GP mean functions using the "soft metric" (Eq. (32) - (34)).

where $\boldsymbol{x}_i$ (resp. $\boldsymbol{x}'_i$) are the i-th entry of $\boldsymbol{x}_i$ (resp. $\boldsymbol{x}'$), and $\alpha_T > 0$ are parameters that controls how smooth are the smoothed tree metrics. Given a learned partition $\mathcal{P}_T = \{p_1, p_2, \cdots, p_K\}$, where $p_j = \prod_{i=1}^{d}[a_j^{(i)}, b_j^{(i)}]$, we construct the following hierarchical Bayesian model

$$\widetilde{a}_j^{(i)} \sim \mathcal{N}(a_j^{(i)}, \sigma^2), \text{ for all } i, j; \quad \widetilde{b}_j^{(i)} \sim \mathcal{N}(b_j^{(i)}, \sigma^2), \text{ for all } i, j \quad (35)$$

$$\widetilde{k}_T = \sum_{j=1}^{K} \widetilde{k}_T^{(p_j)}, \text{ where } \widetilde{k}_T^{(p_j)} \text{ is defined respect to } \prod_{i=1}^{d}[\widetilde{a}_j^{(i)}, \widetilde{b}_j^{(i)}]$$

$$f \sim \mathcal{GP}\left(0, \widetilde{k}_T(\cdot,\cdot)\right) \quad (36)$$

$$y = f + \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, \sigma_y^2). \quad (37)$$

This hierarchical model has several advantages: (1) It respect Lipschitzness. As we collect more observations, the partition can grow arbitrarily fine, and the approximation can be arbitrarily close to an extract indicator function. Because of this, the no prior smoothness assumption on the true (unknown) reward function is needed. (2) It treats arms within the same bin differently, and can use information across bins.

Going back to bandit learning process, we can replace the mean and/or confidence intervals of UCB index with the posteriors of this hierarchical bayesian model. As we discussed in Remark 3, a key insight of our analysis is the link between the Hoeffding-type concentration interval to the posterior variance of the Gaussian processes, which allows us to do this principled substitute. In Section 3.1, we empirically study this hierarchical Bayesian model.

3 EMPIRICAL STUDY

Since the TreeUCB algorithm imposes only mild constraints on tree formation, we use greedy decision tree splitting to fit the reward function, using the following splitting rule: we find the split that maximizes the reduction in the Mean Absolute Error (MAE), and we stop growing the tree once the maximal possible reduction is below 0.001.

3.1 Gaussian Processes with Learned Kernel

In this section, we compare several baselines, including piecewise constant estimates (within each bin), a Gaussian process regression with box kernel (left subfigure in Figure 2) and Gaussian process regression with softened box kernel (right subfigure in Figure 2). The splitting procedure is the same for all methods, so the partitions are the same for the methods. Our results, shown in Figure 3, demonstrates a transition from the hardness of the piecewise constant estimate to the softness of the Gaussian process regression with the softened kernel. This justifies the "softening" discussed in Section 2.3.2. The Gaussian process kernel parameters for $GP_{S,1}, GP_{S,2}, GP_{S,3}, GP_{S,4}, GP_{S,5}$, namely α_T in Eq. (33), were set to 10, 50, 100, 500, 1000 respectively.

3.2 Application to Neural Network Tuning

One application of stochastic bandit algorithms is zeroth order optimization. In this section, we apply TUCB to tuning neural networks. In this setting, we treat the hyperparameter configurations (e.g., learning rate, network architecture) as the arms of the bandit, and use validation accuracy as reward. The task is to select a hyperparameter configuration and train the network to observe the validation accuracies, and find the best hyperparameter configuration rapidly. This experiment shows that TUCB can compete with the state-of-the-art tuning methods on such hard real-world tasks.

The architecture and the hyperparameter space for the simple Multi-Layer Perceptron (MLP) for the MNIST dataset are: in the feed-forward direction, there are the *input layer*, the *fully connected*

hidden layer with dropout ensemble, and then the *output layer*. The hyperparameter search space is five dimensional, including *number of hidden neurons* (range $[10, 784]$), *learning rate* ($[0.0001, 4)$), *dropout rate* ($[0.1, 0.9)$), *batch size* ($[10, 500]$), and *number of iterations* ($[30, 243]$).

The details of the CNN setting for SVHN and CIFAR-10 can be found in Tables 1 and 2. The results are found in Figure 4, indicating that TUCB outperforms existing state-of-the-art software packages for tuning neural network methods.

4 CONCLUSION

We propose the TreeUCB and the Contextual TreeUCB frameworks that use decision trees (regression trees) to flexibly partition the arm space and the context-arm space as an Upper Confidence Bound strategy is played across the partition regions. We also provide regret analysis via the point scattering inequalities. We provide implementations using decision trees that learn the partition. TUCB is competitive with the state-of-the-art hyperparameter optimization methods in hard tasks like neural-net tuning, and could save substantial computing resources. This suggests that, in addition to random search and Bayesian optimization methods, more bandit algorithms should be considered as benchmarks for difficult real-world problems such as neural network tuning.

ACKNOWLEDGEMENT

The authors are grateful to Aaron J Fisher and Tiancheng Liu for insightful discussions. The authors thank anonymous reviewers for valuable feedback. The project is partially supported by the Alfred P. Sloan Foundation through the Duke Energy Data Analytics fellowship.

REFERENCES

[1] Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved Algorithms for Linear Stochastic Bandits. In *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2312–2320.

[2] Shipra Agrawal and Navin Goyal. 2012. Analysis of Thompson Sampling for the Multi-armed Bandit Problem *(Proceedings of Machine Learning Research, Vol. 23)*. JMLR Workshop and Conference Proceedings, Edinburgh, Scotland, 39.1–39.26.

[3] Shipra Agrawal and Navin Goyal. 2013. Thompson Sampling for Contextual Bandits with Linear Payoffs *(Proceedings of Machine Learning Research, Vol. 28)*. PMLR, Atlanta, Georgia, USA, 127–135.

[4] Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research* 3, Nov (2002), 397–422.

[5] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning* 47, 2-3 (2002), 235–256.

[6] Peter Auer, Ronald Ortner, and Csaba Szepesvári. 2007. Improved rates for the stochastic continuum-armed bandit problem. In *International Conference on Computational Learning Theory*. Springer.

[7] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees*. CRC press.

[8] Sébastien Bubeck and Nicolo Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning* (2012).

[9] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. 2011. X-armed bandits. *Journal of Machine Learning Research* 12, May (2011), 1655–1695.

[10] Emile Contal, Vianney Perchet, and Nicolas Vayatis. 2014. Gaussian process optimization with mutual information. In *Proceedings of International Conference on Machine Learning*. 253–261.

[11] Varsha Dani, Thomas P Hayes, and Sham M Kakade. 2008. Stochastic Linear Optimization under Bandit Feedback.. In *Annual Conference on Learning Theory*. 355–366.

[12] Nando de Freitas, Alex Smola, and Masrour Zoghi. 2012. Exponential Regret Bounds for Gaussian Process Bandits with Deterministic Observations. In *Proceedings of International Conference on Machine Learning*.

[13] John C Gittins. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)* (1979), 148–177.

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*.

[15] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. 2008. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 681–690.

[16] Tze Leung Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics* 6, 1 (1985), 4–22.

[17] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World Wide Web*. ACM, 661–670.

[18] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. *The Journal of Machine Learning Research* (2016).

[19] Stefan Magureanu, Richard Combes, and Alexandre Proutiere. 2014. Lipschitz bandits: Regret lower bound and optimal algorithms. In *Annual Conference on Learning Theory*. 975–999.

[20] Ruben Martinez-Cantin. 2014. Bayesopt: A Bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research* (2014).

[21] Chengzhuo Ni, Lin F Yang, and Mengdi Wang. 2019. Learning to control in metric space with optimal regret. In *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, 726–733.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[23] Ohad Shamir. 2011. A variant of Azuma's inequality for martingales with sub-Gaussian tails. *arXiv preprint arXiv:1110.2392* (2011).

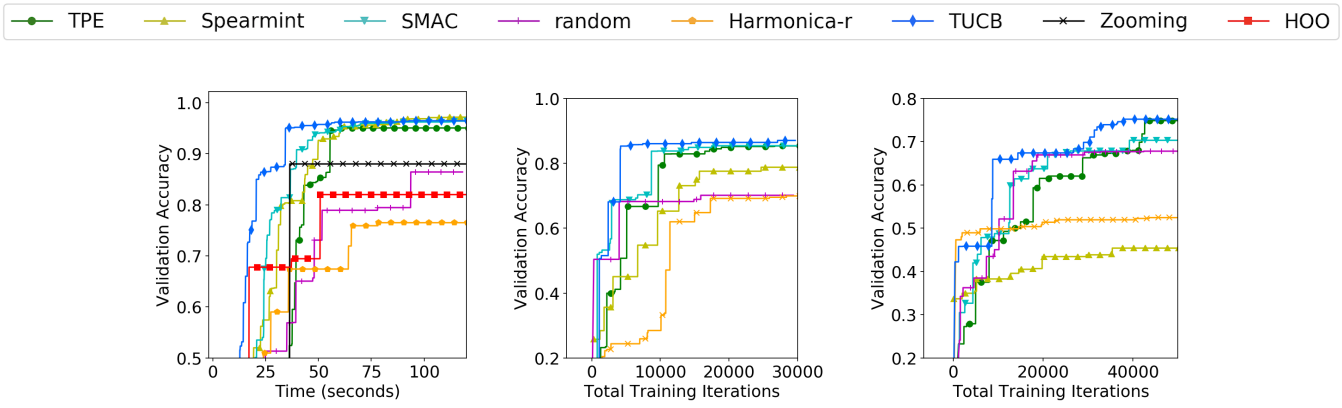
[24] Aleksandrs Slivkins. 2014. Contextual bandits with similarity information. *The Journal of Machine Learning Research* 15, 1 (2014), 2533–2568.

[25] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. 2010. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning*. Omnipress, Haifa, Israel, 1015–1022.

[26] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[27] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* (1933).

[28] Paul E Utgoff. 1989. Incremental induction of decision trees. *Machine learning* 4, 2 (1989), 161–186.



(a) MLP for MNIST (b) CNN for SVHN (c) CNN for CIFAR-10

Figure 4: For MNIST, each plot is averaged over 10 runs. For SVHN and CIFAR-10, each plot is averaged over 5 runs. The implementation of TUCB here uses the scikit-learn package [22]. In the left-most subplot, x-axis is time (in seconds). This shows TUCB's scalability, since TUCB's curve goes up the fastest. In (a), we use clock time as cost measure.