Online Stochastic Optimization under Correlated Bandit Feedback

Mohammad Gheslaghi Azar
Rehabilitation Institute of Chicago, Northwestern University

Alessandro Lazaric
Team SequeL, INRIA Nord Europe

Emma Brunskill
School of Computer Science, CMU

Abstract

In this paper we consider the problem of online stochastic optimization of a locally smooth function under bandit feedback. We introduce the high-confidence tree (HCT) algorithm, a novel anytime \( \mathcal{X} \)-armed bandit algorithm, and derive regret bounds matching the performance of state-of-the-art algorithms in terms of the dependency on number of steps and the near-optimality dimension. The main advantage of HCT is that it handles the challenging case of correlated bandit feedback (reward), whereas existing methods require rewards to be conditionally independent. HCT also improves on the state-of-the-art in terms of the memory requirement, as well as requiring a weaker smoothness assumption on the mean-reward function in comparison with the existing anytime algorithms. Finally, we discuss how HCT can be applied to the problem of policy search in reinforcement learning and we report preliminary empirical results.

1. Introduction

We consider the problem of maximizing the sum of the rewards obtained by sequentially evaluating an unknown stochastic function. This problem is known as stochastic optimization under bandit feedback or \( \mathcal{X} \)-armed bandit, since each function evaluation can be viewed as pulling one of the arms in a generic arm space \( \mathcal{X} \). Our objective is to minimize the cumulative regret relative to selecting at each step the global maximum of the function. In particular, we focus on the case where the reward obtained by pulling an arm (i.e., evaluating the function in a point) may depend on prior history of evaluations and outcomes. This implies that the reward, conditioned on its corresponding arm, is not an independent and identically distributed (iid) random variable, in contrast to prior work on \( \mathcal{X} \)-armed bandits (see e.g., Munos, 2013; Kleinberg et al., 2013; Bubeck et al., 2011a). \( \mathcal{X} \)-armed bandit with correlated reward is relevant to many real-world applications, including internet auctions, adaptive routing, and online games. As one important example, we show that the problem of policy search in an ergodic Markov Decision Process (MDP), a popular setting for learning in unknown MDPs, can be framed as an instance of the setting we consider in this paper (Sect. 5).

Our approach builds on recent advances in \( \mathcal{X} \)-armed bandits for iid settings (Bull, 2013; Djolonga et al., 2013; Bubeck et al., 2011a; Srinivas et al., 2009; Cope, 2009; Kleinberg et al., 2008; Auer et al., 2007). Under regularity assumptions on the mean-reward function (e.g., Lipschitz-smoothness), these methods provide formal guarantees on the cumulative regret, which is proved to scale sub-linearly w.r.t. the number of steps \( n \). To obtain this regret, these methods heavily rely on the iid assumption. To handle non-iid settings, we introduce a new anytime \( \mathcal{X} \)-armed bandit algorithm, called high-confidence tree (HCT) (Sect. 3).

Similar to the HOO algorithm of Bubeck et al. (2011a), HCT makes use of a covering binary tree to explore the arm space. The tree is constructed incrementally in an optimistic fashion, where the exploration of the arm space is guided by upper bounds on the largest reward of the arms covered by a particular node. Our key insight is that to achieve small regret it is enough to expand an optimistic node only when the estimate of its mean-reward has become sufficiently accurate. Under mild ergodicity and mixing assumptions, this allows us to obtain an accurate estimate of the reward of a particular arm even in the non-iid setting. Despite handling a more general case of non-iid feedback, our regret bounds matches (Sect. 4.1) that of HOO (Bubeck et al., 2011a) and zooming algorithm.

---

This assumption implies that, regardless of the history of prior observations, if arm $x$ is pulled infinitely many times from time $s$, then the time average reward converges in expectation to a fixed point which only depends on arm $x$ and is independent from the past history. We also make the following mixing assumption (see e.g., Levin et al., 2006).

**Assumption 2 (Finite mixing time).** There exists a constant $\Gamma \geq 0$ (mixing time) such that for any $x \in \mathcal{X}$, any $S > 0$, any $0 < s \leq S$ and any sequence of prior pulls $(x_1, x_2, \ldots, x_{s-1})$, the process $(z_t)_{t>0}$ is such that we have that $|\mathbb{E}[\sum_{s=s}^S (r_s(x_f) - f(x)) | \mathcal{F}_{s-1}]| \leq \Gamma$.

This assumption implies that the stochastic reward process induced by pulling arm $x$ can not substantially deviate from $f(x)$ in expectation for more than $\Gamma$ transient steps. Note that both assumptions trivially hold if each arm is an iid process: in this case $f(x)$ is the mean of $x$ and $\Gamma = 0$.

Given the mean-reward $f$, we assume that the maximizer $x^* = \arg\max_x f(x)$ exists and we denote the corresponding maximum $f(x^*)$ by $f^*$. We measure the performance of the learner over $n$ steps by its regret $R_n$ w.r.t. the $f^*$, defined as $R_n := n f^* - \sum_{t=1}^n r_t$. The goal of learner, at every $0 \leq t \leq n$, is to choose a strategy $\psi_t$ such that the regret $R_n$ is as small as possible.

**Related models.** Although the learner observes a context $y_t$ at each time $t$, this problem differs from the contextual bandit setting (see e.g., Slivkins, 2009). In contextual bandits, the reward $r_t$ is random realization of a function $r(x_t, y_t)$ of the selected arm and input context $y_t$. The contextual bandit objective is typically to minimize the regret against the optimal arm in the context provided at each step, $y_t$, i.e. $x_t^* = \arg\max_x r(x, y_t)$. A key difference is that in our model the reward, and next context, may depend on the current context and arm, and we define $f(x)$ only as the average reward obtained by pulling arm $x$. In this sense, our model is related to the reinforcement learning (RL) problem of trying to find a policy that maximizes the long run reward. Among prior work in RL our setting is similar to the general reinforcement learning model of Lattimore et al. (2013) which also considers arbitrary temporal dependence between rewards and observations. The main difference is that here we consider the regret in undiscounted reward scenario, whereas the focus of Lattimore et al. (2013) is on proving PAC-bounds in the discounted case. Another difference is that in our model, unlike that of Lattimore et al. (2013), the observation and action spaces need not to be finite (see further discussion in Sect. 5).

**The cover tree.** Similar to recent optimization methods (e.g., Bubeck et al., 2011a), our approach seeks to minimize the regret by building an estimate of $f$ using an infinite binary covering tree $T$, in which each node covers a subset of $\mathcal{X}$. We denote by $(h, i)$ the node at depth $h$ and index $i$ among the nodes at the same depth (e.g., the root index $i$).
The space $\mathcal{X}$ is equipped with a dissimilarity function $\ell : \mathcal{X}^2 \to \mathbb{R}$ such that $\ell(x, x') \geq 0$ for all $(x, x') \in \mathcal{X}^2$ and $\ell(x, x) = 0$.

Given a dissimilarity $\ell$, the diameter of a subset $A \subseteq \mathcal{X}$ is defined as $\text{diam}(A) := \sup_{x, y \in A} \ell(x, y)$, while an $\ell$-ball of radius $\varepsilon > 0$ and center $x \in \mathcal{X}$ is defined as $B(x, \varepsilon) := \{x' \in \mathcal{X} : \ell(x, x') \leq \varepsilon\}$.

Assumption 4 (Local smoothness). We assume that there exist constants $\nu_2, \nu_1 > 0$ and $0 < \rho < 1$ such that for all nodes $(h, i)$:

(a) $\text{diam}(\mathcal{P}_{h,i}) \leq \nu_1 \rho^h$ 
(b) $\exists x_{h,i}^0 \in \mathcal{P}_{h,i}$ s.t. $B_{h,i} := B(x_{h,i}^0, \nu_2 \rho^h) \subseteq \mathcal{P}_{h,i}$, 
(c) $B_{h,i} \cap B_{h,j} = \emptyset$, 
(d) For all $x \in \mathcal{X}$, $f^* - f(x) \leq \ell(x^*, x)$.

These assumptions coincide with those in (Bubeck et al., 2011a), except for the local smoothness (Assm. 4.d), where the function is assumed to be Lipschitz between any two arms $x, x'$ close to the maximum $x^*$ (i.e., $|f(x) - f(x')| \leq \ell(x, x')$), while here we only require the function to be Lipschitz w.r.t. the maximum. Finally, we characterize the complexity of the problem using the near-optimality dimension, which defines how large is the set of $\epsilon$-optimal arms in $\mathcal{X}$. For brevity, we consider a slightly simplified definition of near-optimality dimension w.r.t. (Bubeck et al., 2011a).

Assumption 5 (Near-optimality dimension). Let $\epsilon = 3\nu_2 \rho^h$ and $\epsilon' = \nu_2 \rho^h < \epsilon$, for any subset of $\epsilon$-optimal nodes $\mathcal{X}_\epsilon = \{x \in \mathcal{X} : f^* - f(x) \leq \epsilon\}$, there exists a constant $C$ such that $\mathcal{N}(\mathcal{X}_\epsilon, \ell, \epsilon') \leq C(\epsilon')^{-d}$, where $d$ is the near-optimality dimension of $f$ and $\mathcal{N}(\mathcal{X}_\epsilon, \ell, \epsilon')$ is the $\epsilon'$-cover number of $\mathcal{X}_\epsilon$ w.r.t. the dissimilarity measure $\ell$.

### 3. The High Confidence Tree algorithm

We now introduce the High Confidence Tree (HCT) algorithm. Throughout this discussion, a function evaluation corresponds to the reward received from pulling an arm.

**Algorithm 1 The HCT algorithm.**

**Require:** Parameters $\nu_1 > 0$, $\rho \in (0, 1)$, $c > 0$, tree structure $(P_{h,i})_{h \geq 0, 1 \leq i \leq 2^h}$, and confidence $\delta$.

**Initialize:** $t = 1$, $T_t = \{(0, 1), (1, 1), (1, 2)\}$, $H(t) = 1$, $U_{h,i}(t) = U_{1,2}(t) = +\infty$.

**Loop:**

1. If $t = t^+$ then 
   - $U_{h,i}(t) \leftarrow \hat{\mu}_{h,i}(t) + \nu_1 \rho^h + \sqrt{\frac{2 \log(1/\delta(t^+))}{\tau_{h,i}(t)}}$
   - **Refresh phase**
   - for all $(h, i) \in T_t$ do 
     - if $(h, i) \notin \text{leaf}(T_t)$ then 
       - $B_{h,i}(t) \leftarrow U_{h,i}(t)$
     - else 
       - $B_{h,i}(t) \leftarrow \min \{U_{h,i}(t), \max_{j \in \{2^h-1, 2^h\}} B_{h+1,j}(t)\}$
   - end for
   - end for
   - end if
   - if Algorithm HCT-iid then 
     - Pull arm $x_{h,i}$ and observe $r_t$
     - $t = t + 1$
   - else if Algorithm HCT-1 then 
     - $T_{\text{cur}} = T_{h,i,t}(t)$
     - while $T_{h,i,t}(t) < 2T_{\text{cur}}$ AND $t < t^+$ do 
       - Pull arm $x_{h,i}$ and observe $r_t$
       - $(h_{t+1}, i_{t+1}) = (h_t, i_t)$
     - end while
   - end if
   - Update counter $T_{h,i,t}(t)$ and empirical average $\hat{\mu}_{h,i,t}(t)$
   - $U_{h,i,t}(t) \leftarrow \hat{\mu}_{h,i,t}(t) + \nu_1 \rho^h + \sqrt{\frac{2 \log(1/\delta(t^+))}{\tau_{h,i,t}(t)}}$
   - UpdateB($T_t$, $P_{h,i}$, $(h_t, i_t)$)
   - $\tau_t(h) = \frac{c^2 \log(1/\delta(t^+))}{\nu_1^2}$
   - if $T_{h,i,t}(t) \geq \tau_t(h)$ AND $(h_t, i_t) = \text{leaf}(T)$ then 
     - $T_t = \{(h_t + 1, 2i_t - 1), (h_t + 1, 2i_t)\}$
     - $T_{\text{cur}} = T_{\text{cur}} \cup T_t$
     - $U_{h_t+1,2i_t-1}(t) = U_{h_t+1,2i_t}(t) = +\infty$
   - end if
   - end loop

We first describe the general structure of HCT, before discussing two particular variants: HCT-iid, designed for the case when arm rewards are iid, and HCT-1 which handles the correlated feedback case, where the reward from pulling an arm may depend on all prior arms pulled and resulting outcomes. Alg. 1 shows the structure of the algorithm for HCT-iid and HCT-1 and their minor differences.

**The general structure.** The HCT algorithm relies on a binary covering tree $T$ provided as input to construct a hierarchical approximation of the mean-reward function $f$. At each node $(h, i)$ of the tree, the algorithm keeps track of some statistics regarding the arm $x_{h,i}$ corresponding to node $(h, i)$. These include the empirical estimate $\hat{\mu}_{h,i}(t)$ of...
Online Stochastic Optimization under Correlated Bandit Feedback

Algorithm 2 The OptTraverse function.

Require: Tree $T$

$(h, i) \leftarrow (0, 1), P \leftarrow (0, 1)$

$T_{0, 1} = \tau_0(t) = 1$;

while $(h, i) \not\in \text{Leaf}(T) \text{ AND } T_{h, i}(t) \geq \tau_h(t)$ do

if $B_{h+1, 2i-1} \geq B_{h+1, 2i}$ then

$(h, i) \leftarrow (h + 1, 2i - 1)$

else

$(h, i) \leftarrow (h + 1, 2i)$

end if

$P \leftarrow P \cup \{(h, i)\}$

end while

return $(h, i)$ and $P$

Algorithm 3 The Update$B$ function.

Require: Tree $T$, the path $P_t$, selected node $(h_t, i_t)$

if $(h_t, i_t) \in \text{Leaf}(T)$ then

$B_{ht, i_t}(t) = U_{ht, i_t}(t)$

else

$B_{ht, i_t}(t) = \min\{U_{ht, i}(t), \max_{j \in \{2i_t-1, 2i_t\}} B_{ht+1, j}(t)\}$

end if

for all $(h, i) \in P_t - (h_t, i_t)$ backward do

$B_{h, i}(t) = \min\{U_{h, i}(t), \max_{j \in \{2i-1, 2i\}} B_{h+1, j}(t)\}$

end for

the mean-reward of $x_{h, i}$ computed as

$$\tilde{\mu}_{h, i}(t) := (1/T_{h, i}(t)) \sum_{s=1}^{T_{h, i}(t)} r^s(x_{h, i}),$$  \hspace{1cm} (1)

where $T_{h, i}(t)$ is the number of times node $(h, i)$ has been selected in the past and $r^s(x_{h, i})$ denotes the $s$-th reward observed after pulling $x_{h, i}$ (while we previously used $r_t$ to denote the $t$-th sample of the overall process). As explained in Sect. 2, although a node is associated to a single arm $x_{h, i}$, it also covers a full portion of the input space $\mathcal{X}$, i.e., the subset $\mathcal{P}_{h, i}$. Thus, similar to the HOO algorithm (Bubeck et al., 2011a), HCT also maintains two upper-bounds, $U_{h, i}$ and $B_{h, i}$, which are meant to bound the mean-reward $f(x)$ of all the arms $x \in \mathcal{P}_{h, i}$. For any node $(h, i)$, the upper-bound $U_{h, i}$ is computed as

$$U_{h, i}(t) := \tilde{\mu}_{h, i}(t) + \nu_t \rho^h + c\sqrt{(\log(1/\delta(t^+) + T_{h, i}(t))},$$  \hspace{1cm} (2)

where $t^+ = 2[\log(t^+)] + 1$ and $\delta(t) := \min\{c \delta(t/1)\}$. Intuitively speaking, the second term is related to the resolution of node $(h, i)$ and the third term accounts for the uncertainty of $\tilde{\mu}_{h, i}(t)$ in estimating $f(x_{h, i})$. The $B$-values are designed to have a tighter upper bound on $f(x)$ by taking the minimum between $U_{h, i}$ for the current node, and the maximum upper bound of the node’s two child nodes, if present.\(^1\) More precisely,

\(^1\)Since the node’s children together contain the same input space as the node (i.e., $\mathcal{P}_{h, i} = \mathcal{P}_{h+1, 2i-1} \cup \mathcal{P}_{h+2, i}$), the node’s maximum cannot be greater than the maximum of its children.

To identify which arm to pull, the algorithm traverses the tree along a path $P_t$ obtained by selecting nodes with maximum $B_{h, i}$ until it reaches an optimistic node $(h_t, i_t)$, which is either a leaf or a node which is not pulled enough w.r.t. to a given threshold $\tau_h(t)$, i.e., $T_{h, i}(t) \leq \tau_h(t)$ (see function OptTraverse in Alg. 2). Then the arm $x_{h_t, i_t} \in \mathcal{P}_{h_t, i_t}$ corresponding to selected node $(h_t, i_t)$ is pulled.

The key element of HCT is the condition to decide when to expand the tree. We expand a leaf node only if we have pulled its corresponding arm a sufficient number of times such that the uncertainty over the maximum value of the arms contained within that node is dominated by the size of the subset of $\mathcal{X}$ it covers. Recall from Eq. 2 that the upper bound $U_{h, i}$ is composed of two terms beside the empirical average reward. The first ($\nu_t \rho^h$) is a constant that depends only on the node depth and from assumptions 3 and 4 it follows that it bounds the possible difference in the mean-reward function between the representative arm for this node and all other arms also contained in this node, i.e., the difference between $f(x_{h, i})$ and $f(x)$ for any other $x \in \mathcal{P}_{h, i}$. The second term depends only on $t$ and decreases with the number of pulls. At some point, the second term will become smaller than the first term, implying that the uncertainty over the rewards in $\mathcal{P}_{h, i}$ becomes dominated by the potential difference in the mean-reward of the arms in the node. This means that the domain $\mathcal{P}_{h, i}$ is too large, and thus the resolution of the current approximation of $f$ in that region needs to be increased. Therefore HCT waits until these two terms become of the same magnitude before expanding a node. This happens when the number of pulls $T_{h_t, i_t}(t)$ exceeds a threshold

$$\tau_h(t) := c^2 \log(1/\delta(t^+)) \rho^{-2h_t}/n^2.$$

(see Sect. A of the supplement for further discussion). It is at this point that expanding the node to two children can increase the accuracy of the approximation of $f(x)$, since $\nu_t \rho^h + c\sqrt{(\log(1/\delta(t^+) + T_{h, i}(t))} \leq 2[\log(t^+)] + 1$ and $\delta(t) := \min\{c \delta(t/1)\}$. Intuitively speaking, the second term is related to the resolution of node $(h, i)$ and the third term accounts for the uncertainty of $\tilde{\mu}_{h, i}(t)$ in estimating $f(x_{h, i})$. The $B$-values are designed to have a tighter upper bound on $f(x)$ by taking the minimum between $U_{h, i}$ for the current node, and the maximum upper bound of the node’s two child nodes, if present.\(^1\) More precisely,

\(^1\)Since the node’s children together contain the same input space as the node (i.e., $\mathcal{P}_{h, i} = \mathcal{P}_{h+1, 2i-1} \cup \mathcal{P}_{h+2, i}$), the node’s maximum cannot be greater than the maximum of its children.
As HCT is an anytime algorithm, we periodically need to recalculcate the node upper bounds to guarantee their validity with enough probability (see supplementary material for a more precise discussion). To do so, at the beginning of each step $t$, the algorithm verifies whether the $B$ and $U$ values need to be refreshed or not. In fact, in the definition of $U$ in Eq. 2, the uncertainty term depends on the confidence $\tilde{\delta}^+(t)$, which changes at $t = 1, 2, 4, 8, \ldots$. Refreshing the $U$ and $B$ values triggers a "refreshing phase" of the internal nodes of the tree $T_{\tau}$ along the optimistic path. In fact, the second condition in the OptTraverse function (Alg. 2) forces HCT to pull arms that belong to the current optimistic path $P_t$ until the number of pulls $T_{h,i}(t)$ becomes greater than $\tau_h(t)$ again. Notice that the choice of the confidence term $\tilde{\delta}$ is particularly critical. For instance, choosing a more natural $\tilde{\delta}(t)$ would tend to trigger the refresh (and the resampling) phase too often thus increasing the computational complexity of the algorithm and seriously affecting its theoretical properties. On the other hand, the choice of $\tilde{\delta}(t^+)$ limits the need to refresh the $U$ and $B$ values to only $O(\log(n))$ times over $n$ rounds and guarantees that $U$ and $B$ are valid upper bounds with high probability.

**HCT-iid and HCT-Γ.** The main difference between the two implementations of HCT is that, while HCT-iid pulls the selected arm for only one step before re-traversing the tree from the root to again find another optimistic node, HCT-Γ pulls the the representative arm of the optimistic node for an episode of $T_{\text{cur}}$ steps, where $T_{\text{cur}}$ is the number of pulls of arm $x_{h,i}$ at the beginning of episode. In other words, the algorithm doubles the number of pulls of each arm throughout the episode. Notice that a similar approach has been used before in other methods working with ergodic processes, such as the UCRL algorithm for ergodic MDPs (Jaksch et al., 2010). The additional stopping condition in the loop is such that not all the episodes may actually finish after $T_{\text{cur}}$ steps and double the number of pulls: The algorithm may interrupt the episode when the confidence bounds of $B$ and $U$ are not valid anymore (i.e., $t \geq t^+$) and perform a refresh phase. The reason for this change is that in order to accurately estimate the mean-reward given correlated bandit feedback, it is necessary to pull an arm for a series of pulls rather than a single pull. Due to our assumption on the mixing time (Asm. 2), pulling an arm for a sufficiently long consecutive number of steps will provide an accurate estimate of the mean-reward even in the correlated setting, thus ensuring that the empirical average $\mu_{h,i}$ actually concentrates towards their mean value (see Lem. 2). It is this mechanism, coupled with only expanding the nodes after obtaining a good estimate of their mean reward, that allows us to handle the correlated feedback setting. Although in this sense HCT-Γ is more general, we do however include the HCT-iid variant because whenever the rewards are iid it performs better than HCT-Γ. This is due to the fact that, unlike HCT-iid, HCT-Γ has to keep pulling an arm for a full episode even when there is evidence that another arm could be better. We also notice that there is a small difference in the constants $c_1$ and $c$: in HCT-iid $c_1 := \sqrt[4]{\nu}/3\nu_1$ and $c := 2\sqrt{1/(1-\rho)}$, whereas HCT-Γ uses $c_1 := \sqrt[4]{\nu}/(4\nu_1)$ and $c := 3(3G + 1)\sqrt{1/(1-\rho)}$.

### 4. Theoretical Analysis

In this section we analyze the regret and the complexity of HCT. All the proofs are reported in the supplement.

#### 4.1. Regret Analysis

We start by reporting a bound on the maximum depth of the trees generated by HCT.

**Lemma 1.** Given the threshold $\tau_h(t)$ in Eq. 4, the depth $H(n)$ of the tree $T_{\tau}$, is bounded as

$$H(n) \leq H_{\text{max}}(n) = 1/(1-\rho) \log(n\nu^2/(2(c\rho)^2)).$$

This bound guarantees that HCT never expands trees beyond depth $O(\log n)$. This is ensured by the fact the HCT waits until the mean-reward of a node is sufficiently well estimated before expanding it and this implies that the number of pulls exponentially grows with the depth of tree, thus preventing the depth to grow linearly as in HOO.

We report regret bounds in high probability, bounds in expectation can be obtained using standard techniques.

**Theorem 1 (Regret bound of HCT-iid).** Let Assumptions 3–5 hold and at each step $t$, the reward $r_{h,i}$ is independent of all prior random events. Then the regret of HCT-iid in $n$ steps is, with probability $1-\delta$:

$$R_n \leq O\left( (\log(n/\delta))^{1/(d+2)} n^{(d+1)/(d+2)} \right).$$

**Remark (the bound).** We notice that the bound perfectly matches the bound for HOO up to constants (see Thm. 6 in (Bubeck et al., 2011a)). This represents a first sanity check w.r.t. the structure of HCT, since it shows that changing the structure of HOO and expanding nodes only when they are pulled enough, preserves the regret properties of the algorithm. Furthermore, this result holds under milder assumptions than HOO. In fact, Asm. 4-(d) only requires $f$ to be Lipschitz w.r.t. to the maximum $x^*$. Other advantages of HCT-iid are discussed in the Sect. 4.2 and 6.

Although the proof is mostly based on standard techniques and tools from bandit literature, HCT has a different structure from HOO (and similar algorithms) and moving from iid to correlated arms calls for the development of a significantly different proof technique. The main technical issue

---

2 Constants are provided in Sect. A of the supplement.
is to show that the empirical average $\hat{\mu}_{h,i}$ computed by averaging rewards obtained across different episodes actually converges to $f(x_{h,i})$. In particular, we prove the following high-probability concentration inequality (see Lem. 6 in the supplement for further details).

**Lemma 2.** Under Assumptions 1 and 2, for any fixed node $(h, i)$ and step $t$, we have that, w.p. $1 - \delta$,

$$|\hat{\mu}_{h,i}(t) - f(x_{h,i})| \leq (3 \Gamma + 1) \sqrt{\frac{2 \log(5/\delta)}{T_{h,i}(t)}} + \Gamma \frac{\log(t)}{T_{h,i}(t)}.$$

Furthermore $K_{h,i}(t)$, the number of episodes in which $(h, i)$ is selected, is bounded by $\log_{2}(4T_{h,i}(t)) + \log_{2}(t)$.

This technical lemma is at the basis of the derivation of the following regret bound for $HCT-\Gamma$.

**Theorem 2** (Regret bound of $HCT-\Gamma$). Let Assumptions 1–5 hold and that rewards are generated according to the general model defined in Sect. 2. Then the regret of $HCT-\Gamma$ after $n$ steps is, w.p. $1 - \delta$, $R_n \leq O((\log(n/\delta))^{1/(d+2)} n^{d/(d+2)})$.

**Remark (the bound).** The most interesting aspect of this bound is that $HCT-\Gamma$ achieves the same regret as $HCT$-iid when samples are non-iid. This represents a major step forward w.r.t. the existing algorithms, since it shows that the very general case of correlated rewards can be managed as well as the simpler iid case. In Sect. 5 we discuss how this result can be used in policy search for MDPs.

4.2. Complexity

**Time complexity.** The run time complexity of both versions of $HCT$ is $O(n \log(n))$. This is due to the boundedness of the depth $H(n)$ and by the structure of the refresh phase. By Lem. 1, we have that the maximum depth is $O(\log(n))$. As a result, at each step $t$, the cost of traversing the tree to select a node is at most $O(\log(n))$, which also coincides with the cost of updating the $B$ and $U$ values of the nodes in the optimistic path $P_t$. Thus, the total cost of selecting, pulling, and updating nodes is no larger than $O(n \log(n))$. Notice that in case of $HCT-\Gamma$, once a node is selected it is pulled for an entire episode, which further reduces the total selection cost. Another computational cost is represented by the refresh phase where all the nodes in the tree are actually updated. Since the refresh is performed only when $t = t^+$, then the number of times all the nodes are refreshed is of order of $O(\log(n))$ and the boundedness of the depth guarantees that the number of nodes to update cannot be larger than $O(2^{\log n})$, which still corresponds to a total cost of $O(n \log(n))$. This implies that $HCT$ achieves the same run time as $T$-HOO (Bubeck et al., 2011a). Though unlike $T$-HOO, our algorithm is fully anytime and it does not suffer from the extra regret incurred due to the truncation and the doubling trick.

**Space complexity.** The following theorem provides bound on space complexity of the $HCT$ algorithm.

**Theorem 3.** Under the same conditions of Thm. 2, let $N_n$ denote the space complexity of $HCT-\Gamma$, then we have that $\mathbb{E}(N_n) = O((\log(n))^{2/(d+2)} n^{d/(d+2)})$.

This result guarantees that the space complexity of $HCT-\Gamma$ scales sub-linearly w.r.t. $n$. An important observation is that the space complexity of $HCT-\Gamma$ increases slower, by a factor of $O(n^{1/(d+2)})$, than its regret. This implies that, for small values of $d$, HCT does not require to use a large memory space to achieve a good performance. An interesting special case is the class of problem with near-optimality dimension $d = 0$. For this class of problems the bound translates to a space complexity of $O(\log(n))$, whereas the space complexity of alternative algorithms may be as large as $n$ (see e.g., HOO). The fact that $HCT-\Gamma$ solves the optimization problem using only a relatively small memory space makes it a suitable choice for big-data applications, where the algorithms with linear space complexity can not be used due to very large size of the dataset.

**Switching frequency.** Finally, we also remark another interesting feature of $HCT-\Gamma$. Since an arm is pulled for an entire episode before another arm could be selected, this drastically reduces the number of switches between arms. In many applications, notably in reinforcement learning (see next section), this can be a significant advantage since pulling an arm may correspond to the actual implementation of a complex solution (e.g., a position in a portfolio management problem) and continuously switch between different arms might not be feasible. More formally, since each node has a number of episodes bounded by $O(\log(n))$ (Lem. 2), then the number of switches can be derived from the number of nodes in Thm. 3 multiplied by $O(\log(n))$, which leads to $O((\log(n))^{(d+4)/(d+2)} n^{d/(d+2)})$.

5. Application to Policy Search in MDPs

As discussed in Sect. 2, $HCT$ is designed to handle the very general case of optimization in problems with strong correlation among the rewards, arm pulls, and contexts, at different time steps. An important subset of this general class is represented by the problem of policy search in infinite-horizon *ergodic* Markov decision processes.

A MDP $M$ is defined as a tuple $\langle S, A, P \rangle$ where $S$ is the set of states, $A$ is the set of actions, $P : S \times A \rightarrow M(S \times [0, 1])$ is the transition kernel mapping each state-action pair to a distribution over states and rewards. A (stochastic) policy $\pi : S \rightarrow M(A)$ is a mapping from states to distribution over actions. Policy search algorithms (Scherrer & Geist, 2013; Azar et al., 2013; Kober & Peters, 2011) aim at finding the policy in a given policy set which maximizes the long-term performance. Formally, a policy search algorithm receives as input a set of policies $G = \{\pi_\theta; \theta \in \Theta\}$,
each of them parameterized by a parameter vector $\theta$ in a given set $\Theta \subset \mathbb{R}^d$. Any policy $\pi_\theta \in \mathcal{G}$ induces a state-reward transition kernel $T : \mathcal{S} \times \Theta \to \mathcal{M}(\mathcal{S} \times [0, 1])$. $T$ relates to the state-reward-action transition kernel $P$ and the policy kernel $\pi_\theta$ as follows $T(ds', dr[s, \theta]) := \int_{\theta \in \Theta} P(ds', dr[s, \theta]) \pi_\theta ds$. For any $\pi_\theta \in \mathcal{G}$ and initial state $s_0 \in \mathcal{S}$, the time-averaged reward over $n$ steps is $\mu_\pi^n(s_0, n) := 1/n \mathbb{E}[\sum_{t=1}^n r_t]$, where $r_1, r_2, \ldots, r_n$ is the sequence of rewards observed by running $\pi_\theta$ for $n$ steps staring at $s_0$. If the Markov reward process induced by $\pi_\theta$ is ergodic, $\mu_\pi^n(s_0, n)$ converges to a fixed point independent of the initial state $s_0$. The average reward of $\pi_\theta$ is thus defined as $\mu(\theta) := \lim_{n \to \infty} \mu_\pi^n(s_0, n)$. The goal of policy search is to find the best $\theta^* = \arg \max_{\theta \in \Theta} \mu(\theta)$.

It is straightforward now to match the MDP scenario to the general setting in Sect. 2, notably mapping $\Theta$ to $\mathcal{X}$ and $\mu(\theta)$ to $f(x)$ (further details are provided in Sect. D of the supplement). This allows us to directly apply HCT-$\Gamma$ to the problem of policy search. The advantage of HCT-$\Gamma$ algorithm w.r.t. prior work is that, to the best of our knowledge, it is the first policy search algorithm which provides finite sample guarantees in the form of regret bounds on the performance loss of policy search in MDPs (see Thm. 2), which guarantee that HCT-$\Gamma$ suffers from a small sub-linear regret w.r.t. $\pi_\theta$. Also, it is possible to prove that the policy induced by HCT-$\Gamma$ has a simple regret, that is, the average reward of the policy chosen by HCT-$\Gamma$ converges to $\mu(\pi^*)$ with a polynomial rate. Another interesting feature of HCT-$\Gamma$ is that it can be used in large (continuous) state-action problems since it does not make any restrictive assumption on the size of state-action space.

**Related work.** A related work to HCT-$\Gamma$ is the UCRL algorithm by Ortner & Ryabko (2012), which extends the original UCRL algorithm (Jaksch et al., 2010) to continuous state spaces. Although a direct comparison between the two methods is not possible, it is interesting to notice that the assumptions used in UCRL are stronger than for HCT-$\Gamma$, since they require both the dynamics and the reward function to be globally Lipschitz. Furthermore, UCRL requires the action space to be finite, while HCT-$\Gamma$ can deal with any continuous policy space. Finally, while HCT-$\Gamma$ is guaranteed to minimize the regret against the best policy in the policy class $\mathcal{G}$, UCRL targets the performance of the actual optimal policy of the MDP at hand. Another relevant work is the OMDP algorithm of Abbasi et al. (2013) which deals with the problem of RL in continuous state-action MDPs with adversarial rewards. OMDP achieves a sub-linear regret under the assumption that the space of policies is finite.

**6. Numerical Results.**

While our primary contribution is the definition of HCT and its technical analysis, we also give preliminary simulation results to demonstrate some of its properties.

**Setup.** We focus on minimizing the regret across repeated noisy evaluations of the garland function $f(x) = x(1 - x)(4 - \sqrt{\sin(60x)})$ relative to repeatedly selecting its global optima.\(^5\) We evaluate the performance of each algorithm in terms of the per-step regret, $R_n = R_n / n$. Each run is $n = 10^5$ steps and we average the performance on 10 runs. For all the algorithms compared in the following, parameters\(^6\) are optimized to maximize their performance.

**I.i.d. setting.** In the first experiment we compare HCT-iid to the truncated hierarchical optimistic optimization (T-HOO) algorithm (Bubeck et al., 2011a). T-HOO is a state-of-the-art $\mathcal{X}$-armed bandit algorithm, developed as a computationally-efficient alternative of HOO. In Fig. 1 we show the per-step regret, the runtime, and the space requirements of each approach. As predicted by the theoretical bounds, the per-step regret $R_n$ of both HCT-iid and T-HOO decreases rapidly with number of steps. Though the big-O bounds are identical for both approaches, empirically we observe that in this setting HCT-iid outperforms T-HOO by a large margin. Similarly, though the computational complexity of both approaches matches in the dependence on the number of time steps, empirically we observe that our approach outperforms T-HOO (Fig. 1). Perhaps the most significant expected advantage of HCT-iid over T-HOO for iid settings is in the space requirements. HCT-iid has a space requirement for this domain that scales logarithmically with the time step $n$, as predicted by Thm. 3. In contrast, in this domain we observe a polynomial growth of memory usage for T-HOO. These patterns mean that HCT-iid can achieve a very small regret using a sparse cover tree with only few hundred nodes, whereas T-HOO requires orders of magnitude more nodes than HCT-iid.

**Correlated setting.** In this setting, we compare HCT-$\Gamma$ to PoWER, a standard RL policy search algorithm (Kober & Peters, 2011), on a continuous-state-action MDP constructed out of the garland function.\(^7\) PoWER uses an Expectation Maximization approach to optimize the policy parameters. We also compare our algorithm with T-HOO, although this algorithm is designed for the iid setting and it may fail to converge to the global optimum under correlated bandit feedback. Fig. 2 shows per-step regret of the 3 approaches in the MDP. Only HCT-$\Gamma$ suc-

---

\(^{5}\)We discuss some properties of the garland function in Sect. C of the supplement where the function is illustrated in Fig. 3.

\(^{6}\)For both HCT and T-HOO we introduce a tuning parameter used to multiply the upper bounds, while for PoWER we optimize the window for computing the weighted average.

\(^{7}\)See Sect. C of the supplement for details.
ceeds in finding the globally optimal policy, since only for HCT-$\Gamma$ the average regret tends to converge to zero (as predicted by Thm. 2). The PoWER method finds worse solutions than both stochastic optimization approaches for the same amount of computational time, likely due to using EM which is known to be susceptible to local optima. On the other hand, its primary advantage is that it has a very small memory requirement. Overall this illustrates the benefit of HCT for online MDP policy search, since it can quickly (as a function of samples and runtime) find a global optima, and is, to our knowledge, one of the only policy search methods guaranteed to do so.

7. Discussion and Future Work

In the current version of HCT we assume that the learner has access to the information regarding the smoothness of function $f(x)$ and the mixing time $\Gamma$. In many problems those information are not available to the learner. In the future it would be interesting to build on prior work that handles unknown smoothness in iid settings and extend it to correlated feedback. For example, Bubeck et al. (2011b) require a stronger global Lipschitz assumption and propose an algorithm to estimate the Lipschitz constant. Other work on the iid setting include Valko et al. (2013) and Munos (2011), which are limited to the simple regret scenario, but who only use the mild local smoothness assumption we define in Asm. 4, and do not require knowledge of the dissimilarity measure $\ell$. On the other hand, Slivkins (2011) and Bull (2013) study the cumulative regret but consider a different definition of smoothness related to the zooming concept introduced by Kleinberg et al. (2008). Finally, we notice that to deal with unknown mixing time, one may rely on data-dependent tail’s inequalities, such as empirical Bernstein inequality (Tolstikhin & Seldin, 2013; Maurer & Pontil, 2009), replacing the mixing time with the empirical variance of the rewards.

In the future we also wish to explore using HCT in other problems that can be modeled as optimization with correlated bandit feedback. For example, HCT may be used for policy search in partially observable MDPs (Vlassis & Toussaint, 2009; Baxter & Bartlett, 2000), as long as the POMDP is ergodic.

To conclude, in this paper we introduce a new $X$-armed bandit algorithm, called HCT, for optimization under bandit feedback and prove regret bounds and simulation results for it. Our approach improves on existing results to handle the important case of correlated bandit feedback. This allows HCT to be applied to a broader range of problems than prior $X$-armed bandit algorithms, such as policy search in continuous MDPs.

Acknowledgements

This work was supported in part by the NSF Award SBE-0836012 to the Pittsburgh Sciences of Learning Center. We would like to thank R. Munos, A. Farahmand and A. Slivkins for valuable discussions. A. Lazaric would like to acknowledge the support of the Ministry of Higher Education and Research, Nord-Pas- de-Calais Regional Council and FEDER through the Contrat de Projets Etat Region (CPER) 2007-2013, and the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement 231495 (project CompLACS).
Online Stochastic Optimization under Correlated Bandit Feedback

References


