“God Doesn’t Play Dice with the World”: Time to Move Beyond i.i.d. Assumption

Anonymous Author(s)
Affiliation
Address
email

1 Introduction

We consider the general problem of online optimization with the bandit feedback when, given an arm, its corresponding reward is not an i.i.d. random variable. The problem arises naturally in many interactive real-world settings such as online auctions, adaptive routing and online games, where the reward, at each time step, may depend not only on the latest arm pull but also on the entire history of previous observations.

Let $\mathcal{X}$ be a space of arms. We consider the optimization problem as an interaction between the decision maker and the environment: at each time step t, the decision maker pulls an arm X_t in $\mathcal{X}$. The environment in return provides the learner with a reward $Y_t \in [0,1]$ which depends on the history of previous rewards and pulls. We note the mean-payoff function $f(x)$ as the expected time-average of the received rewards while we pull arm x infinitely many times:

$$f(x, \mathcal{H}_0) = \lim_{n \to +\infty} \mathbb{E}\left[\frac{1}{n} \sum_{t=1}^{n} Y_t \middle| X_{1:n} = x, \mathcal{H}_0 \right],$$

where $X_{1:n}$ is the history of arm pulls from $t = 1$ to $t = n$ and $\mathcal{H}_0$ is the history of all observations prior to $t = 1$.[1] It is not difficult to prove that under the mixing assumption, which we introduce later in Sec. 2, the above limit always exists and it is independent of $\mathcal{H}_0$. So from now on we make use of the shorthand notation $f(x)$ instead of $f(x, \mathcal{H}_0)$. We also define the regret $\mathcal{R}_n$ w.r.t. the maximum payoff as follows:

$$\mathcal{R}_n = n \sup_{x \in \mathcal{X}} f(x) - \sum_{t=1}^{n} Y_t$$

The goal of decision maker is to choose the sequence of arms $X_1, X_2, \ldots, X_n$ such that the regret $\mathcal{R}_n$ becomes as small as possible. To solve this problem, we rely on the recent advances in the field of continuum-armed bandit (Valko et al., 2013; Bubeck et al., 2011a; Kleinberg et al., 2008; Auer et al., 2007). Those works address the problem of stochastic non-convex optimization under the assumption that given X_t the reward Y_t is independent of all other random events. Here we relax this assumption and introduce a new algorithm called *High Confidence Tree* (HCT) which also applies to the case of dependent Y_ts. Similar to the HOO algorithm of Bubeck et al. (2011a), *HCT* makes use of a covering binary tree for exploring $\mathcal{X}$. Furthermore, our algorithm relies on the celebrated optimism in the face of uncertainty principle to make balance between exploration and exploitation: It maintains upper bounds on the values of $f(x)$ for all regions of $\mathcal{X}$ and zoom into the region with the highest upper bound on $f(x)$ (optimistic node) by expanding its leaves. The main new idea, which allows *HCT* to handle non-i.i.d. rewards, is based on the observation that one only should expand an optimistic node when the algorithm achieves an accurate estimate of $f(x)$, with high confidence, for every leaf of the tree. Until that moment the algorithm may reside with the corresponding arm of the current optimistic node. In fact to achieve the optimal rate the accuracy of the estimates needs to grow exponentially with the depth of the tree, which also implies that

[1] Here we let negative values for time steps.

the number of pulls for the optimistic node should increase exponentially with the depth. The fact that the number of pulls increases exponentially with the depth prevents the algorithm from pulling too many different arms which is essential to achieve a sub-linear regret in the case of dependent rewards.[2]

We prove that under some mild mixing assumption *HCT* can achieve a regret of $\widetilde{O}(n^{(d+1)/(d+2)})$ where d is the near-optimality dimension of the mean-payoff function (see Bubeck et al., 2011a, for the definition of near-optimality dimension). This result matches those of HOO (Bubeck et al., 2011a) and zooming algorithm (Kleinberg et al., 2008) in terms of dependency on n and d. However our results covers a more general setting of dependent rewards as opposed to the bounds of HOO and zooming algorithm which only apply to i.i.d. setting. Also, one one show that due to exponential growth in the number of pulls the maximum depth of tree in *HCT* is no more than $O(\log(n))$ which also implies that the computational complexity of HCT is at maximum $O(n \log(n))$. This is an important observation since *HCT* achieves this linearithmic computational complexity without using any truncation or doubling trick which is required for the fast version of HOO algorithm. Finally, *HCT* also has a very favorable space requirement which makes it a suitable choice for online learning with *big data*. In fact one can show that in the case of **benign** mean-payoff function where the near-optimality dimension is 0 the space requirement of *HCT* is of order $O(\log(n))$. This is an improvement on HOO algorithm which, even with truncation, still may need $O(n)$ memory space regardless of the difficulty of optimization problem.

2 Background

In this section we briefly describe the assumptions needed for *HCT*.

2.1 Statistical Assumption

In this extended abstract we make no restrictive statistical assumption, such as Markov property, on the of dependency of observations on each other. In fact our results apply to a rather general setting where the reward Y_t may depend on the entire history of all previous observations. In that sense our approach can be used to solve any optimization problem with dependent observations as long as the following mixing assumption holds,

Assumption 1 (Mixing sequence of rewards)**.** *Let $Y_1, Y_2, \dots, Y_n$ be a sequence of rewards induced by pulling arm x, n times in a row. Define $\mathcal{H}_0$ as the history of all observations prior to Y_1. We assume that there exists some universal constant $\tau > 0$ for which following inequality holds for every integer $n > 0$, $x \in \mathcal{X}$ a:*

$$\left| \mathbb{E}\left[\sum_{t=1}^{n} Y_t \middle| \mathcal{H}_0 \right] - f(x) \right| \leq \tau$$

The above mixing assumption is only slightly stronger than the ergodicity assumption, which arguably is the most common assumption for weakly dependent sequences of random variables. In fact one can show that if Y_t belongs to a finite set this two assumptions are equivalent. Moreover one can easily prove that for any *fast mixing* ergodic sequence of random variables Assumption 1 always hold.

2.2 Geometrical Assumptions

We begin by a brief description of the binary tree we use for exploring $\mathcal{X}$:[3] The covering decision tree is used to estimate the mean-payoff function over the space $\mathcal{X}$. The main idea is to build an accurate estimate of f around its maximum $f_{\sup} \triangleq \sup_{x\in\mathcal{X}} f(x)$ while avoiding the low reward regions of $\mathcal{X}$ as much as possible. To achieve this goal we approximate the mean-payoff function with an infinite binary *tree of covering* $\mathcal{T}$. The tree consists of the set of nodes each corresponds with a a subset of $\mathcal{X}$. Each node is indexed by a pair $\{(h, i)\}$ where h is the depth of the node and i is its index among the nodes in depth h (the root node which covers the entire $\mathcal{X}$ is indexed by $(0, 1)$). By convention $(h + 1, 2i - 1)$ and $(h + 1, 2i)$ is used to refer to the two children of the node

[2]Note that, unlike i.i.d. setting which we can switch between arms at any time, in the case of dependent observations we need a long trajectories of rewards to estimate the expected time-average accurately.

[3]The reader is referred to Bubeck et al. (2011a) for a more detailed description of the covering tree.

(h,i). Also the corresponding area of each (h,i) is denoted by $P_{h,i} \subset \mathcal{X}$. These regions must be measurable and satisfy the following constraints:

$$\mathcal{P}_{0,1} = \mathcal{X}$$
$$\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h,2i} \qquad \text{for all} h \geq 0 \text{ and } 1 \leq i \leq 2^h.$$

In words the sum of the areas of all nodes at any depth h accumulates to the space $\mathcal{X}$. Also there should be no overlap between the areas of the nodes at any depth h.

We now state our main geometrical assumption regarding the space $\mathcal{X}$ and mean-payoff function f:

Assumption 2 (One sided Lipschitzness). *Given a dissimilarity l,* [4] *the diameter of a subset A of $\mathcal{X}$ is defined by diam$(A) \triangleq \sup_{x,y\in A} l(x,y)$. Also the l–open ball of $\mathcal{X}$ with radius $\varepsilon > 0$ and center $x \in \mathcal{X}$ is defined by $\mathcal{B}(\mathcal{X},\varepsilon) \triangleq \{y \in \mathcal{Y} : l(x,y) \leq \varepsilon\}$. We then assume that there exists $\nu_2, \nu_1 > 0$ and $0 < \rho < 1$ such that for all integers $h \geq 0$:*

(a) diam$(\mathcal{P}_{h,i}) \leq \nu_1 \rho^h$

(b) there exists $x^o_{h,i} \in \mathcal{P}_{h,i}$ such that $\mathcal{B}_{h,i} \triangleq \mathcal{B}(x^o_{h,i}, \nu_2\rho^h) \subset \mathcal{P}_{h,i}$ for all $i = 1, \ldots, 2^h$.

(c) $\mathcal{B}_{h,i} \cap \mathcal{B}_{h,j} = \emptyset$ for all $1 \geq i < j \leq 2^h$.

(c) Then for all $x, y \in \mathcal{X}$ the mean-payoff function satisfies

$$f_{\sup} - f(x) \leq \max\{f_{sup} - f(y)\}, l(x,y)\}$$

3 Algorithm

Similar to HOO algorithm, in HCT the binary tree $\mathcal{T}$ keeps tracks of some statistics regarding every arm $x_{h,i}$ (corresponding arm of node (h,i)). In particular we save the values of empirical mean-payoff $\widehat{\mu}_{h,i}$ defined as follows:

$$\widehat{\mu}_{h,i} = 1/T_{h,i} \sum_{t=1}^{T_{h,i}} Y_t, \tag{1}$$

in which $T_{h,i}$ is the number of updates of arm $x_{h,i}$. The algorithm also saves the upper-bounds $U_{h,i}$ which is defined as follows:

$$\begin{cases} U_{h,i} = \widehat{\mu}_{h,i} + ((2\sqrt{2} + \rho^h)\tau + \nu_1)\rho^h & (h,i) \text{ is a leaf} \\ U_{h,i} = \max(U_{h+1,2i-1}, U_{h+1,2i}) & \text{otherwise.} \end{cases} \tag{2}$$

The HCT algo. proceeds in phases (see Algo. 1). At each phase the algorithm works as follows: the algorithm finds the leaf with the highest upper confidence $U_{h,i}$ and expands it. It then selects an arm randomly in the corresponding area of each of new nodes and pulls that arm for $(1/\rho)^{2h} \log(9(2/\rho^2)^{2H_{\max}})$ times,[5] that is, the total number of pulls required to achieve a confidence interval of order $O(\rho^h)$ with high probability. In the case that $H_{\max}$ increases form the previous phase the algorithm also pulls the corresponding arms of all other leaves until their $T_{h,i} \geq (1/\rho)^{2h} \log(9(2/\rho^2)^{2H_{\max}})$. This is to guarantee that $U_{h,i}$ is uniform bound on $f(x)$ with a same high probability for every (h,i).

3.1 Main Result

In this section, we state our main theoretical result which is in the form of bound on the expected regret of *HCT*. The result matches the previous result of Kleinberg et al. (2008) and Bubeck et al. (2011a). Though here we do not require the i.i.d. assumption.

Theorem 1. *Define $4(3\tau + \nu_1)/\nu_2$ near-optimality dimension d of function f with respect to the dissimilarity $l(.,.)$ as in (Bubeck et al., 2011a). Then under Assumption 1 and Assumption 2 the following bound holds on the expected regret:*[6]

$$\mathbb{E}(\mathcal{R}_n) = O((\log(n))^{1/(d+2)} n^{(d+1)/(d+2)} + \log(n)).$$

[4]See (Bubeck et al., 2011a) for the formal definition of dissimilarity function.

[5]$H_{\max}$ is the maximum depth of $\mathcal{T}$.

[6]We will include the proof in a longer version.

Algorithm 1 A phase of *HCT* algorithm.

Require: Decision tree $\mathcal{T}$, $U_{h,i}$ and $T_{h,i}$ for all nodes in the tree, maximum depth $H_{\max}$ and a real number $\rho \in (0, 1)$

Find the optimistic leaf $(h^+, i^+) = \arg\max_{(h,i)=\text{leaf}(\mathcal{T})} U_{h,i}$ and add its children to the tree

if $h^+ = H_{\max}$ **then** $H_{\max} \leftarrow H_{\max} + 1$

end if

for all $(h, i) = \text{leaf}(\mathcal{T})$ **do**

  repeat pulling (h, i) and updating $T_{h,i}$

  until $T_{h,i} \geq (1/\rho)^{2h} \log(9(2/\rho^2)^{2H_{\max}})$

  Update $U_{h,i}$ From Eq. 1 and Eq. 2

end for

4 Discussion

In this section we discuss some of the outstanding issues regarding *HCT* algorithm.

Run time of *HCT* As we mentioned earlier, the run time of *HCT* is at maximum $O(n \log(n))$. This is due to the fact that the maximum depth of the covering tree in *HCT* can not become larger than $O(\log(n))$: the number of pulls exponentially grows with the depth of $\mathcal{T}$, which implies that the depth of tree is at maximum $O(\log(n))$. The run time of algorithm is, therefore, no more than $O(n \log(n))$ since , except for $O(\log(n))$ steps, *HCT* only traverses one leaf per each step, which requires no more that $O(h) \leq O(\log(n))$ computation. This implies that *HCT* achieves the same run time as that of *truncated HOO*(Bubeck et al., 2011a). Though unlike truncated HOO, we need not to suffer the extra regret incurred due to truncation and doubling trick.

Space complexity *HCT* is also very efficient in terms of its memory usage. As we argued earlier the depth of tree in *HCT* is bounded by $O(\log(n))$. So *HCT* needs at most $O(|I_{\max}|(\log(n))$ memory space to represent the tree, where $I_{\max}$ is the maximum number of nodes per depth. Since we only expand those optimistic nodes which have reached the confidence intervals of $O(\rho^h)$, from the definition of near-optimality dimension d (see Bubeck et al., 2011a), the total number of nodes per depth is at maximum $I_{\max} = O(\rho^{-dH_{max}})$ with a high probability. In the case of benign optimization problems, where $d = 0$, $I_{\max}$ is a constant which leads to the space complexity of $O(\log(n))$.[7] To the best of our knowledge *HCT* is the only optimistic optimization algorithm which can represent the mean-payoff function using only $O(\log(n))$ memory space (e.g., for the same setting the space complexity of HOO can be as large as n).

Unknown smoothness and mixing time In the current version of HCT we assume that the decision maker has access to the information regarding the smoothness of function $f(x)$ as well as the mixing time τ. In many problems those information are not available to the decision maker. The case of unknown smoothness has been relatively well studied. In the absence of the knowledge of dissimilarity function, one may estimate the smoothness in an online manner and then use the estimated metric as it is the true metric (Bubeck et al., 2011b). Another solution is to rely on simultaneous approaches for optimistic optimization (Munos, 2011; Valko et al., 2013). Those methods require not the knowledge of smoothness (dissimilarity function) nor they need to estimate from the data, though they only provide guarantees in terms of simple regret as opposed to the more common notion of accumulated regret, which we consider in this extended abstract. On the issue of unknown mixing time τ one may rely on more powerful tails inequalities such as empirical Bernstein which can replace the dependency on the mixing time with some notion of empirical variance of the rewards. However, to the best of our knowledge there is no previous work on the extension of empirical tail's inequalities to the case of weakly dependent random variables, which we consider in this paper.

[7]More generally depending on the value of d a sub-linear space complexity can be achieved.

References

Auer, P., Ortner, R., and Szepesvári, C. (2007). Improved rates for the stochastic continuum-armed bandit problem. In *COLT*, pages 454–468.

Bubeck, S., Munos, R., Stoltz, G., and Szepesvári, C. (2011a). X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695.

Bubeck, S., Stoltz, G., and Yu, J. Y. (2011b). Lipschitz bandits without the lipschitz constant. In *ALT*, pages 144–158.

Kleinberg, R., Slivkins, A., and Upfal, E. (2008). Multi-armed bandits in metric spaces. In *STOC*, pages 681–690.

Munos, R. (2011). Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *NIPS*, pages 783–791.

Valko, M., Carpentier, A., and Munos, R. (2013). Stochastic simultaneous optimistic optimization. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 19–27.