BANDIT VIEW IN NOISY OPTIMIZATION

Asif Shaikh 20D070017
Kushal Pawaskar 20D070059

ABSTRACT

Optimization of an unknown function from a finite number of its noisy evaluations can be modeled as a bandit problem. We are concerned with the case where the function is (weakly) Lipschitz continuous and defined over a bounded and continuous set. We consider the online optimization framework, where the result of an evaluation is associated to a reward, and the goal is to maximize the sum of obtained rewards. To this end, we use the Hierarchical Optimistic Optimization (HOO) strategy, and also propose a modification to the strategy which can help in achieving the objective faster.

1 INTRODUCTION

There are several strategies for optimization of a function. Having a convex function makes the task a lot easier, but doing the same for a non-convex function is difficult. When given an unknown function, one does not have any initial idea about its convexity. Thus, one of the methods left for optimizing the function is through querying the function at different parts of the domain, and obtaining an estimate of the function value in that domain. Now, this estimate won't be an exact value but will come with some error depending on the part of domain chosen for querying.

Here, we try to optimize an unknown function from a finite number of noisy evaluations. To make the best use of evaluations, one may want to estimate the seemingly best options more precisely, while for bad options, a rough estimate might be enough.

Online Optimization is a setting where the result of an evaluation (query) is associated to a reward, and the objective is to maximize the sum of obtained rewards. Thus, at each time step (i.e., at each evaluation), there is a trade-off induced between exploitation and exploration. Exploration means trying to obtain more information about the options, in this case, it means evaluating the function on different parts of the domain to obtain corresponding rewards. Here, we will be considering that the reward given by the function is simply the value of the function at that point. Since we are evaluating on a range of values, the function value could be considered to be $avg\_value \pm error$. Now, at each part of the domain, we could again evaluate on some subpart of that part, which would give a more accurate estimation. This phase is called the exploration phase.

Thus, the algorithm has to balance between trying to obtain more information about the options and selecting the option which seem to yield, in expectation, the highest rewards. When there are a large number of options (possibly infinite), for the purpose of analysis, we can consider them to form a continuous set and map them to expected reward function, which can be assumed to be Lipschitz continuous.

In this work we have modified the HOO strategy from Audibert (2010) to achieve the objective.

2 PROBLEM SETUP

Before heading on to the case of optimization of a function on a continuous set, let us first build up a bit on the discrete optimization case. In this case, there is only a finite set of options. Here, we assign an upper confidence bound (UCB) on the mean reward of each option, and then select the option with the highest bound.

To assess a strategy we will use expected cumulative regret, defined as

$$R_n = n \max_{1 \le i \le K} \mu_i - \sum_{t=1}^{n} \mathbb{E}\mu_{I_t}.$$

Here, n is the total number of evaluations, μ_i is the mean reward of the option i in n evaluations, I_t denotes the option evaluated at time t, $\mathbb{E}\mu_{I_t}$ is the expected reward of option I_t in the t evaluations done. Thus, R_n represents the difference in expected reward between the optimal strategy and the strategy used by us.

Upper Confidence Bound (UCB)
Let exploration rate be $\alpha > 0$. For an option (arm) i, define its upper confidence bound index $B_{i,s,t}$ by

$$B_{i,s,t} = \hat{X}_{i,s} + \sqrt{\frac{\alpha log(t)}{s}},$$

for $s, t \ge 1$, and $B_{i,0,t} = \infty$. Here, s is the number of evaluations of option i in t time steps (one evaluation per time step), and $\hat{X}_{i,s}$ is the expected reward for option i evaluated s times in t time steps.
At time t, evaluate an option I_t maximizing $B_{i,T_i(t-1),t}$, where $T_i(t-1)$ denotes the number of times we evaluate option i during the first $t-1$ time steps.

The UCB index is defined in the above way to aid the exploration-exploitation process. Notice the $log(t)/s$ term. Here, the denominator increases linearly but the numerator increases logarithmically. Hence, if the same option is evaluated repeatedly, this term would become smaller after each evaluation for that option. But when this option is being evaluated, every other option isn't being evaluated. Hence, s remains constant for every other option wheres $log(t)$ increasing. Hence, the upper bound of the evaluated option comes down and that of the other options goes up. Thus, we can choose other options for exploration, and we won't get stuck in exploiting only one option.

Now, let us come to the continuous case. In this case, we make a smoothness assumption on the mapping from options to expected reward (the mean payoff function to be optimized). Let $\mathcal{X}$ denote the set of options, f the mean payoff function, and f^* the supremum of f over $\mathcal{X}$. Consider the sequence $(\mathcal{P}_{h,i})_{h \ge 0, 1 \le i \le 2^h}$ of subsets of $\mathcal{X}$ satisfying

- $\mathcal{P}_{0,1} = \mathcal{X}$, and for all $h \ge 0, 1 \le i \le 2^h, \mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}$.
- There exists $v_1, v_2 > 0$ and $\rho \in (0,1)$ such that each $\mathcal{P}_{h,i}$ is included in a ball of radius $v_1\rho^h$ and contains a ball of radius $v_2\rho^h$. Moreover, for a given h, the balls of radius $v_2\rho^h$ are all disjoint.

Intuitively, for a given h, the sets $(\mathcal{P}_{h,i})_{1 \le i \le 2^h}$ represent a covering of $\mathcal{X}$ at "scale" h.

The proposed algorithm takes this sequence of subsets and the real numbers v_1, ρ as inputs. Moreover, the sequence $(\mathcal{P}_{h,i})$ will be represented as an infinite binary tree, where the nodes are indexed by pairs of integers (h, i), such that the nodes $(h+1, 2i-1)$ and $(h+1, 2i)$ are the children of (h, i). The subset $\mathcal{P}_{h,i}$ is associated with node (h, i).

3 THE HIERARCHICAL OPTIMISTIC OPTIMIZATION (HOO) STRATEGY

The core idea of HOO is to estimate f precisely around its maxima, while estimating it loosely in other parts of the space $\mathcal{X}$. To implement this idea, HOO maintains the binary tree described in previous section, whose nodes are associated with subsets of $\mathcal{X}$ such that the regions associated with nodes deeper in the tree represent increasingly smaller subsets of $\mathcal{X}$. At each node, HOO stores some statistics based on the information received in the previous evaluations. In particular, HOO keeps track of the number of times a node was traversed up to round n and the corresponding empirical average of the rewards received so far. Based on these, HOO assigns an optimistic estimate (denoted by B) to the maximum mean payoff associated with each node. These estimates are then used to select the next node to evaluate. This is done by traversing the tree, beginning from the root and always following the node with the highest B. Once a node is selected, a point in the region associated with it is chosen and is evaluated. Based on the point selected and the reward received, the tree is updated.

Parameters: Two real numbers $v_1 > 0$ and $\rho \in (0, 1)$, a sequence $(\mathcal{P}_{\langle,\rangle})_{h\geq 0, 1\leq i\leq 2^h}$ of subsets of $\mathcal{X}$.
Auxiliary function: $LEAF(T)$: outputs a leaf of T.
Initialization: $\mathcal{T} = \{(0, 1)\}$ and $B_{1,1} = B_{1,2} = \infty$.

Algorithm 1 The HOO Strategy

1: **for** $n = 1, 2, \ldots$ **do**
2: $(h, i) \leftarrow (0, 1)$
3: $P \leftarrow \{(h, i)\}$
4: **while** $(h, i) \in \mathcal{T}$ **do**
5: **if** $B_{h+1,2i-1} > B_{h+1,2i}$ **then**
6: $(h, i) \leftarrow (h + 1, 2i - 1)$
7: **else if** $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
8: $(h, i) \leftarrow (h + 1, 2i)$
9: **else**
10: $Z \sim Ber(0.5)$
11: $(h, i) \leftarrow (h + 1, 2i - Z)$
12: **end if**
13: $P \leftarrow P \cup \{(h, i)\}$
14: **end while**
15: $(H, I) \leftarrow (h, i)$
16: Choose option x in $\mathcal{P}_{H,I}$ and evaluate it
17: Receive corresponding reward Y
18: $\mathcal{T} \leftarrow \mathcal{T} \cup \{(H, I)\}$
19: **for all** $(h, i) \in \mathcal{P}$ **do**
20: $T_{h,i} \leftarrow T_{h,i} + 1$
21: $\hat{\mu}_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}_{h,i} + Y/T_{h,i}$
22: **end for**
23: **for all** $(h, i) \in \mathcal{T}$ **do**
24: $U_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{(2logn)/T_{h,i}} + v_1\rho^h$
25: **end for**
26: $B_{H+1,2I-1} \leftarrow \infty$
27: $B_{H+1,2I} \leftarrow \infty$
28: $\mathcal{T}' \leftarrow \mathcal{T}$
29: **while** $\mathcal{T}' \neq \{(0, 1)\}$ **do**
30: $(h, i) \leftarrow LEAF(\mathcal{T}')$
31: $B_{h,i} \leftarrow \min\{U_{h,i}, \max\{B_{h+1,2i-1}, B_{h+1,2i}\}\}$
32: $\mathcal{T}' \leftarrow \mathcal{T}'\backslash\{(h, i)\}$
33: **end while**
34: **end for**

Here, P denotes the path taken in an evaluation from root node, $T_{h,i}$ denotes the number of times the (h, i) node is evaluated, $\hat{\mu}_{h,i}$ is the corresponding mean reward, and $U_{h,i}$ then denotes the updated mean payoff upper bound. Notice line 24 of algorithm 1. Here, the square root term is similar to the term in UCB discussed in the discrete optimization part of section 2. It helps in making sure that other paths are also explored.

4 Our modification to HOO

Our modification to HOO involves adding a max depth parameter (d) so that the tree starting from the root cannot extend beyond max depth. The reasoning behind this is that as the depth increases, and the subset size decreases, the accuracy of estimation would improve; but this would happen in a submodular way, at least after some large value of depth, i.e., the improvements would diminish after every unit increase in depth. This can be seen from the ρ^h term on line 24 of algorithm 1. Hence, to save time, it could be done that no further nodes are added to the tree that result in increasing depth beyond some limit.

Parameters: Three real numbers $d > 0$, $v_1 > 0$, and $\rho \in (0, 1)$, a sequence $(\mathcal{P}_{\langle,\rangle})_{h\geq 0, 1\leq i\leq 2^h}$ of subsets of $\mathcal{X}$.
Auxiliary function: $LEAF(T)$: outputs a leaf of T.
Initialization: $\mathcal{T} = \{(0, 1)\}$ and $B_{1,1} = B_{1,2} = \infty$.

Algorithm 2 The Modified HOO Strategy

for $n = 1, 2, \ldots$ **do**
  $(h, i) \leftarrow (0, 1)$
  $P \leftarrow \{(h, i)\}$
  while $(h, i) \in \mathcal{T}$ **do**
    if $B_{h+1,2i-1} > B_{h+1,2i}$ **then**
      $(h, i) \leftarrow (h + 1, 2i - 1)$
    else if $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
      $(h, i) \leftarrow (h + 1, 2i)$
    else
      $Z \sim Ber(0.5)$
      $(h, i) \leftarrow (h + 1, 2i - Z)$
    end if
    $P \leftarrow P \cup \{(h, i)\}$
  end while
  $(H, I) \leftarrow (h, i)$
  Choose option x in $\mathcal{P}_{H,I}$ and evaluate it
  Receive corresponding reward Y
  if $H \leq d$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{(H, I)\}$
  else
    continue
  end if
  for all $(h, i) \in \mathcal{P}$ **do**
    $T_{h,i} \leftarrow T_{h,i} + 1$
    $\hat{\mu}_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}_{h,i} + Y/T_{h,i}$
  end for
  for all $(h, i) \in \mathcal{T}$ **do**
    $U_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{(2logn)/T_{h,i}} + v_1\rho^h$
  end for
  $B_{H+1,2I-1} \leftarrow \infty$
  $B_{H+1,2I} \leftarrow \infty$
  $\mathcal{T}' \leftarrow \mathcal{T}$
  while $\mathcal{T}' \neq \{(0, 1)\}$ **do**
    $(h, i) \leftarrow LEAF(\mathcal{T}')$
    $B_{h,i} \leftarrow \min\{U_{h,i}, \max\{B_{h+1,2i-1}, B_{h+1,2i}\}\}$
    $\mathcal{T}' \leftarrow \mathcal{T}' \backslash \{(h, i)\}$
  end while
end for

REFERENCES

Jean-Yves Audibert. 1 bandit view on noisy optimization. 2010. URL https://api.semanticscholar.org/CorpusID:1246139.