CS769: Course Project

Optimization in Machine Learning

August 17, 2023

Instructor: Prof Ganesh Ramakrishnan

GROUP MEMBERS
Raavi Gupta, 200070064
Vedang Gupta, 200100166

Contents

1 Introduction

The focus of our project is to address the problem of function optimization under the constraint of having only a finite number of noisy evaluations. Naive repeated sampling is not the most efficient approach and is a wasteful technique. Instead, we aim to optimize the use of evaluations by obtaining precise estimates for the most promising options, while accepting rough estimates for the less attractive ones. Developing such an approach requires the design of sophisticated algorithms, which are dependent on the specific objective criterion and the budget constraint for the number of evaluations. Overall, our project seeks to address the challenge of function optimization with noisy evaluations using advanced algorithmic strategies based on the theory of multi-armed bandits. We will also extend these ideas to the continuous setting by partitioning the space appropriately.

2 Problem Statement

Formally, the multi-armed bandits can be described as follows: Consider a finite set of options $\{1, \ldots, K\}$, also called arms. To each option $i \in \{1, \ldots, K\}$ we associate a (reward) distribution ν_i on $[0, 1]$, with mean μ_i. Let i^* denote an optimal arm, that is, $\mu_{i^*} = \max_{1 \le j \le K} \mu_j$. We denote the suboptimality gap of option i by $\Delta_i = \mu_{i^*} - \mu_i$, and the minimal positive gap by $\Delta = \min_{i:\Delta_i > 0} \Delta_i$. We assume that when one evaluates an option i, one receives a random variable drawn from the underlying probability distribution ν_i (independently from the previous draws). We investigate strategies that perform sequential evaluations of the options to find the one with the highest mean. More precisely, at each time step $t \in \mathbb{N}$, a strategy chooses an option I_t to evaluate. We denote by $T_i(t)$ the number of times we evaluated option i up to time t, and by $\widehat{X}_{i,T_i(t)}$ the empirical mean estimate of option i at time t (based on $T_i(t)$ i.i.d. random variables). In this project, we consider two objectives for the strategy.

1. **Pure Exploration** - The learner possesses an evaluation budget, and once this budget is exhausted, he or she has to select an option S as the candidate for being the best option. The performance of the learner is evaluated only through the quality of option S.

2. **Online Optimization** - The result of an evaluation is associated to a reward, and the learner wants to maximize his or her cumulative rewards. This setting corresponds to the classical multi-armed bandit setting.

For discrete cases, we can directly use the bandit algorithms we will describe next, where we will take all discrete options as arms, then for any black box function with noisy evaluations, we will explore the space in an optimal way given a finite budget of evaluations.

We will also extend these methods to the continuous case under some local smoothness and Lipschitz assumptions. We will discuss the optimistic optimization technique extended to the continuous space using the Lipschitz assumption and the exploration and partitioning of the space using bandit algorithms.

3 Methodology

First, we explore the discrete case without the online setting, i.e., the pure exploration case. Here, we have two cases:

- **Fixed Budget Setting** - Number of evaluations is fixed. Here, we used the following algorithms:

  1. **Uniform Exploration** - Here, each arm is pulled equal number of times, and the arm with the highest empirical mean is chosen.
  2. **Successive Rejects** - Here, unpromising arms are eliminated in phases to incentivize exploration of the optimal arms. The algorithms is given as follows:

Let $A_1 = \{1, \ldots, K\}$, $\overline{\log}(K) = \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$, $n_0 = 0$ and for $k \in \{1, \ldots, K-1\}$,

$$n_k = \left\lceil \frac{1}{\overline{\log}(K)} \frac{n-K}{K+1-k} \right\rceil.$$

For each phase $k = 1, 2, \ldots, K-1$:

(1) For each $i \in A_k$, select option i for $n_k - n_{k-1}$ evaluations.

(2) Let $A_{k+1} = A_k \backslash \arg\min_{i \in A_k} \hat{X}_{i,n_k}$ (we remove only one element from A_k; if there is a tie, randomly select the option to dismiss among the worst options).

Recommend the unique element of A_K.

Figure 1: Successive Rejects Algorithm [1]

- **Fixed Confidence Setting** - Confidence in evaluations fixed (w.p. $1-\delta$, the algorithm predicted best arm is the optimal arm, δ fixed), we want to reduce number of evaluations. For this case, we have the following algorithms:

  1. **Hoeffding Race** -

Parameter: the confidence level δ.

Let $A = \{1, \ldots, K\}$ and $t = 1$

While $|A| > 1$

(1) sample every option in A for the t-th time.

(2) remove from A all the options having an empirical mean differing from the highest empirical mean by more than $\sqrt{2\log(nK/\delta)/t}$, that is,

$$A \leftarrow A \setminus \left\{ j \in A : \widehat{X}_{j,t} \leq \max_{1 \leq i \leq K} \widehat{X}_{i,t} - \sqrt{\frac{2\log(nK/\delta)}{t}} \right\}.$$

(3) $t \leftarrow t+1$.

Output the unique element of A.

Figure 2: Hoeffding Race Algorithm [1]

2. **Bernstein Race** -

Parameter: the confidence level δ.

Let $A = \{1, \dots, K\}$ and $t = 1$

While $|A| > 1$

(1) sample every option in A for the t-th time.

(2) remove suboptimal options from A:

$$A \leftarrow A \setminus \left\{ j \in A : \widehat{X}_{j,t} + \sqrt{\frac{2V_{j,t}\log(nK/\delta)}{t}} + 6\frac{\log(nK/\delta)}{t} \leq \max_{1 \leq i \leq K} \left(\widehat{X}_{i,t} - \sqrt{\frac{2V_{i,t}\log(nK/\delta)}{t}} \right) \right\},$$

where $V_{i,t} = \frac{1}{t}\sum_{s=1}^{t} \left(X_{i,s} - \widehat{X}_{i,t} \right)^2$ is the empirical variance of option i.

(3) $t \leftarrow t + 1$.

Output the unique element of A.

Figure 3: Bernstein Race Algorithm [1]

3. **Action Elimination** -

Input : $\delta > 0$
Output : An arm
Set $t = 1$ and $S = A$;
Set for every arm a: $\hat{p}_a^1 = 0$;
Sample every arm $a \in S$ once and let $\hat{p}_a^t$ be the average reward of arm a by time t;
repeat
 Let $\hat{p}_{max}^t = \max_{a \in S} \hat{p}_a^t$ and $\alpha_t = \sqrt{\ln(cnt^2/\delta)/t}$, where c is a constant;
 foreach arm $a \in S$ such that $\hat{p}_{max}^t - \hat{p}_a^t \geq 2\alpha_t$ **do**
  set $S = S \setminus \{a\}$;
 end
 $t = t + 1$;
until $|S| > 1$;

Figure 4: Action Elimination Algorithm [2]

Next, for the online optimization case we have the following algorithms that we explored:

1. **UCB:** UCB (Upper Confidence Bound) algorithm selects the arm with the highest upper confidence bound, which takes into account both the expected reward and the uncertainty of the estimate.

2. **UCB-V:** UCB-V (UCB with Variance) works by using the variance of the rewards obtained from each arm, in addition to the mean, to make decisions about which arm to choose.

3. **MOSS:** MOSS (Minimax Optimal Strategy for the Stochastic case) algorithm maintains an estimate of the maximum expected reward for each arm and selects the arm

that has the highest estimate, subject to a constraint on the minimum number of times each arm must be played.

Require: The exploration rate $\alpha > 0$

UCB Index: $B_{i,s,t} = \hat{X}_{i,s} + \sqrt{\frac{\alpha \log(t)}{2s}}$

UCB-V Index: $B_{i,s,t} = \hat{X}_{i,s} + \sqrt{\frac{2\alpha V_{i,s} \log(t)}{2s}} + + \frac{3\alpha \log(t)}{s}$

MOSS Index: $B_{i,s,t} = \hat{X}_{i,s} + \sqrt{\frac{\max(\log(\frac{n}{Ks}),0)}{s}}$

for $s, t \geq 1$, and $B_{i,0,t} = +\infty$. At time t, evaluate an option I_t, maximising $B_{i,T_i(t-1),t}$ where $T_i(t-1)$ denotes the number of times we evaluated option i during $t-1$ first steps

Next, we discuss the case of continuous optimization. Here, under Lipschitz assumptions, having evaluated the function at some points, one can upper bound the function as follows:

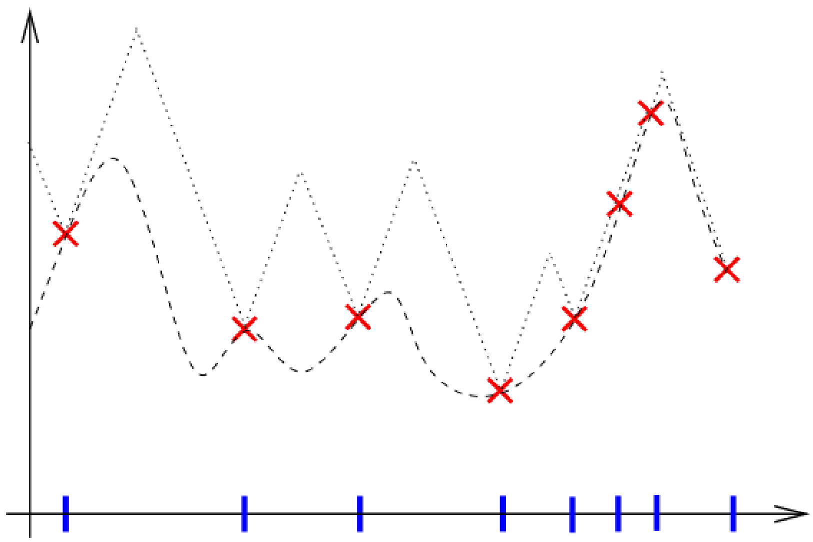

Figure 5: Upper Bound due to Lipschitz Assumption

Now, the optimistic optimization strategies chooses most promising upper bound as the next point to evaluate, similar to UCB. This can be extended for noisy evaluations using the algorithm called HOO - Hierarchical Optimistic Optimization, which is as follows:

Parameters: Two real numbers $\nu_1 > 0$ and $\rho \in (0,1)$, a sequence $(\mathcal{P}_{h,i})_{h \geq 0, 1 \leq i \leq 2^h}$ of subsets of $\mathcal{X}$.

Auxiliary function $\textsc{Leaf}(\mathcal{T})$: outputs a leaf of $\mathcal{T}$.

Initialization: $\mathcal{T} = \{(0,1)\}$ and $B_{1,2} = B_{2,2} = +\infty$.

for $n = 1, 2, \ldots$ **do** ▷ Strategy HOO in round $n \geq 1$
  $(h,i) \leftarrow (0,1)$ ▷ Start at the root
  $P \leftarrow \{(h,i)\}$ ▷ P stores the path traversed in the tree
  while $(h,i) \in \mathcal{T}$ **do** ▷ Search the tree $\mathcal{T}$
    if $B_{h+1,2i-1} > B_{h+1,2i}$ **then** ▷ Select the "more promising" child
      $(h,i) \leftarrow (h+1, 2i-1)$
    else if $B_{h+1,2i-1} < B_{h+1,2i}$ **then**
      $(h,i) \leftarrow (h+1, 2i)$
    else ▷ Tie-breaking rule
      $Z \sim \mathrm{Ber}(0.5)$ ▷ e.g., choose a child at random
      $(h,i) \leftarrow (h+1, 2i-Z)$
    end if
    $P \leftarrow P \cup \{(h,i)\}$
  end while
  $(H,I) \leftarrow (h,i)$ ▷ The selected node
  Choose option x in $\mathcal{P}_{H,I}$ and evaluate it ▷ Arbitrary selection of an option
  Receive corresponding reward Y
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(H,I)\}$ ▷ Extend the tree
  for all $(h,i) \in P$ **do** ▷ Update the statistics T and $\hat{\mu}$ stored in the path
    $T_{h,i} \leftarrow T_{h,i} + 1$ ▷ Increment the counter of node (h,i)
    $\hat{\mu}_{h,i} \leftarrow \big(1 - 1/T_{h,i}\big)\hat{\mu}_{h,i} + Y/T_{h,i}$ ▷ Update the mean $\hat{\mu}_{h,i}$ of node (h,i)
  end for
  for all $(h,i) \in \mathcal{T}$ **do** ▷ Update the statistics U stored in the tree
    $U_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{(2\log n)/T_{h,i}} + \nu_1 \rho^h$ ▷ Update the U–value of node (h,i)
  end for
  $B_{H+1,2I-1} \leftarrow +\infty$ ▷ B–values of the children of the new leaf
  $B_{H+1,2I} \leftarrow +\infty$
  $\mathcal{T}' \leftarrow \mathcal{T}$ ▷ Local copy of the current tree $\mathcal{T}$
  while $\mathcal{T}' \neq \{(0,1)\}$ **do** ▷ Backward computation of the B–values
    $(h,i) \leftarrow \textsc{Leaf}(\mathcal{T}')$ ▷ Take any remaining leaf
    $B_{h,i} \leftarrow \min\Big\{U_{h,i}, \max\big\{B_{h+1,2i-1}, B_{h+1,2i}\big\}\Big\}$ ▷ Backward computation
    $\mathcal{T}' \leftarrow \mathcal{T}' \setminus \big\{(h,i)\big\}$ ▷ Drop updated leaf (h,i)
  end while
end for

Figure 6: HOO Strategy [1]

One drawback of HOO is that the regret guarantees depend on knowing the parameter of local smoothness. But, [3], extend this and give guarantees for the unknown parameter using a new algorithm called Simultaneous Optimistic Optimization (SOO), described below:

Algorithm 1 Sketch of algorithm SOO to minimize function f in a given domain using maxeval calls to the objective function. Parameters are the maximum depth of the tree h_{max}, the number of function evalautions maxeval, and the split factor k.

```
t ← 1 // number of objective function evaluation having been made so far
// C denotes the set of cells:
// initially, there is one cell that comprises the whole domain of definition of the objective function, at depth 0 in the tree
C ← {(whole domain, 0)}
while t ≤ maxeval do
  B ← ∅ // set of cells that have to be splitted
  v_min ← +∞ // current best evaluation of the function
  for h = 0; h < min(maxdepth(C), h_max) do
    among all leaves of C at depth h, select the one associated to the best point: x*_h
    if f(x*_h) ≤ v_min then
      add this cell to B
      v_min ← f(x*_h)
    end if
  end for
  split each cell in B into k sub-cells and add these sub-cells to C; evaluate their center point; update t accordingly (exit
  the loop if maxeval is reached)
end while
Return (the best found point)
```

Figure 7: SOO Strategy [3]

4 Results

4.1 Discrete Optimization

	ACCURACY					TIME STEPS		
Sampling Method	**SR**	**UE**	**UCB**	**UCB-V**	**MOSS**	**AE**	**Hoeffding**	**Bernstein**
Uniform	0.67	0.63	0.68	0.7	0.67	18661	5609	1369
Systematic	0.86	0.76	0.81	0.86	0.91	3807	1498	743
Stratified	0.71	0.57	0.76	0.79	0.65	2996	725	545
Halton	**0.9**	**0.79**	**0.85**	**0.88**	**0.91**	**2309**	**569**	**492**

Table 1: Performance of different algorithms with $f(x) = x/10$ as objective

	ACCURACY					TIME STEPS		
Sampling Method	**SR**	**UE**	**UCB**	**UCB-V**	**MOSS**	**AE**	**Hoeffding**	**Bernstein**
Uniform	0.81	0.59	0.86	0.78	0.74	1443	543	488
Systematic	0.97	0.88	0.94	0.98	**0.98**	**725**	476	401
Stratified	0.95	0.87	0.94	0.93	0.93	4830	957	594
Halton	**0.99**	**0.88**	**0.94**	**0.98**	0.96	886	**383**	**335**

Table 2: Performance of different algorithms with $f(x) = -(x - 0.5)^2$ as objective

	ACCURACY					TIME STEPS		
Sampling Method	**SR**	**UE**	**UCB**	**UCB-V**	**MOSS**	**AE**	**Hoeffding**	**Bernstein**
Uniform	0.51	0.49	0.45	0.6	0.6	1187251	98414	1910
Systematic	0.86	0.67	0.82	0.8	0.81	183888	37265	**1224**
Stratified	0.75	0.69	0.71	0.72	0.79	132604	32486	1883
Halton	**0.88**	**0.69**	**0.85**	**0.81**	**0.84**	**31264**	**13464**	1542

Table 3: Performance of different algorithms with $f(x) = \frac{\sin(10(x+0.3))}{10(x+0.3)}$ as objective

4.2 Continuous Optimization

Functions	Time(s)	Best Solution	Est.	Actual
$\sin(x)$	2.52	[(-1.5712890625,-1.5703125)]	-0.99	$-\pi/2$, -1
$(sin(13x)sin(27x)+1)/2$	2.42	[(-1.33642578125, -1.3359375)]	0.997	-1.336 , 1
$cos(x)+cos(y)$	2.52	[(1.99609375, 2), (-2, -1.9921875)]	-0.82	[± 2 ± 2], -0.82
$sin(x)sin(y)sin(3x)$ $cos(2y)/x^2y$	2.51	[(0.0, 0.0078125), (1.4296875, 1.4375)]	-1.99	-2

Table 4: HOO Results

Functions	Time(s)	Best Solution	Est. Value	Actual Value
$sin(x)$	0.41	-1.5708	-0.99	$-\pi/2$, -1
$(sin(13x)sin(27x)+1)/2$	0.82	-1.5707	$1.29 * 10^{-9}$	$-\pi/2$, 0
$cos(x)+cos(y)$	0.74	[3.1491 3.1405]	-1.99	[π π], -2
$sin(x)sin(y)sin(3x)cos(2y)/x^2y$	0.59	[-0.0021 -1.4308]	-1.99	-2

Table 5: SOO Results

5 Observations and Conclusions

For a fixed budget of 1000 steps or a fixed confidence of 90% we have the following observations:

- As expected, for the fixed confidence case, Bernstein Race easily outperforms the other algorithms.
- Halton Sampling gives the best results in almsot all cases.
- MOSS seems to be the best performing online optimization algorithm.
- Successive Rejects is the best performing algorithm for the fixed budget case.

An avenue of future work in the discrete case is assuming some structure on the function and thereby adapting a particular better sampling strategy.

For the continuous case, SOO seems to be faster than HOO but that maybe due to specific architecture or implementation. Both seem to converge quite quick to the appropriate solution. In the future we would like to benchmark these against a wide variety of black box functions and optimize the code and tree traversals for both cases.

References

[1] J.-Y. Audibert, S. Bubeck, and R. Munos, *Bandit View on Noisy Optimization*, ch. 1. MIT Press, optimization for machine learning ed., January 2010.

[2] E. Even-Dar, S. Mannor, and Y. Mansour, "Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems," *Journal of Machine Learning Research*, vol. 7, no. 39, pp. 1079–1105, 2006.

[3] P. Preux, R. Munos, and M. Valko, "Bandits attack function optimization," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2245–2252, 2014.

[4] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari, "X-armed bandits," 2011.