CSE599i: Online and Adaptive Machine Learning **Winter 2018**

Lecture 6: Non-stochastic best arm identification

Lecturer: Kevin Jamieson *Scribes: Anran Wang, Beibin Li, Brian Chan, Shiqing Yu, Zhijin Zhou*

1 Motivation

Before we define the non-stochastic best arm identification problem, we will start with a couple of motivating examples. Applications for this type of bandit problem often come with various assumptions that differ from previous bandit problems we have encountered, so starting with the examples is the easiest way to point out these differences.

1.1 Non-Convex Optimization

Non-convex optimization is one possible application for non-stochastic best arm identification. We begin by explaining the problem:

> **Example 1.** Imagine that we are solving a non-convex optimization problem on some (multivariate) function f using gradient descent. Recall that gradient descent converges to *local* minima. Because non-convex functions may have multiple minima, we cannot guarantee that gradient descent will converge to the global minimum. To resolve this issue, we will use *random restarts*, the process of starting multiple instances of gradient descent from random locations and outputting the least-valued minimum found. If an instance of gradient descent is obviously not going to converge quickly to a low-valued minimum, that instance should be stopped early in favor of other, more-promising ones.

We can pose this example as a non-stochastic pure exploration multi-armed problem: assume that we will run n different instances of gradient descent, each with a different starting location $\mathbf{x}_{i,0}$ for $i \in [n]$. We also define $\mathbf{x}_{i,t_i}$ to be the value of the i-th gradient descent instance after t_i iterations. Now we can make a bandit arm i for each instance and set its loss after being pulled t_i times to be $\ell_{i,t_i} \triangleq f(\mathbf{x}_{i,t_i})$.

We assume that arms are completely unrelated, so knowledge about one arm does not transfer at all to other arms. Although this may not be entirely true, in practice it is often difficult to determine relationships between arms (we wouldn't need gradient descent if we could analytically determine such information).

This bandit problem is non-stochastic since there is no randomness when incurring the loss for each arm—the loss function is deterministic. The process by which we choose arms (i.e. choosing starting locations for gradient descent instances) is random, but this randomness is not relevant since it occurs before the problem starts. Furthermore, our adversary is oblivious: the loss function does not depend on our previous sequence of arm pulls. This is good—adaptive adversaries are very difficult to reason about; remember that we made the same obliviousness assumption for the non-stochastic regret minimization.

In addition, the problem involves pure exploration because we just want to find the arm corresponding to the gradient descent instance which converges to the global minimum (i.e., the arm whose loss converges to the least value). We want to minimize the total number of arm pulls required to determine this best arm, and do not need to minimize the losses incurred along the way as we would in a regret minimization problem.

1.2 Hyperparameter Optimization

Best arm identification for non-stochastic infinitely-armed bandits can also be applied to hyperparameter optimization, where we attempt to find the best hyperparameters for some supervised machine learning model on some training data.

Example 2. In supervised learning, we are essentially trying to model some unknown bivariate distribution $\mathbb{P}_{X,Y}$ representing real-world data. We will call the variables X and Y, and their sample spaces $\mathcal{X}$ and $\mathcal{Y}$. In particular, we want to find a mapping function $f : \mathcal{X} \to \mathcal{Y}$ that, given the X value from a pair sampled from $\mathbb{P}_{X,Y}$, can predict the Y value. More formally, we want to find an f that minimizes $\mathbb{E}_{(X,Y)\sim\mathbb{P}_{X,Y}}[\mathcal{L}(f(X), Y)]$ for some arbitrary loss function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Here $\mathcal{Y} \times \mathcal{Y}$ denotes the Cartesian product of $\mathcal{Y}$ with itself. Since we do not know $\mathbb{P}_{X,Y}$, we approximate this expected loss by averaging the individual losses on a *validation dataset* (*validation set*) of k known pairs of data $\mathcal{V} \triangleq \{(x_j, y_j)\}_{j=1}^{k}$ sampled i.i.d. from the distribution. To help us find this mapping function, we sample a *training dataset* (*training set*) of another m pairs $\mathcal{T} \triangleq \{(x_j, y_j)\}_{j=k+1}^{k+m}$ from $\mathbb{P}_{X,Y}$. If the validation set is the data we will use to evaluate the mapping function, the training set is the data used to find the function—we keep these separate because using the same data to both create and evaluate a mapping function can result in a function with good performance (i.e., small loss) on that dataset but poor performance on other data.

We consider mapping functions output by some chosen algorithm $\mathcal{A} : \{(\mathcal{X}, \mathcal{Y})\} \times \Theta \times \mathbb{N} \to \{f : \mathcal{X} \to \mathcal{Y}\}$, which takes as input the training set $\mathcal{T}$, some hyperparameters $\theta \in \Theta$, and a number of training iterations t_i to output a mapping function. (The algorithm $\mathcal{A}$ may also be randomized.) Our task, then, is to find the hyperparameters θ that minimize the mean validation loss: $\arg\min_{\theta\in\Theta} \frac{1}{k} \sum_{j=0}^{k} \mathcal{L}(f_{\theta,t_i}(x_j), y_j)$ where $f_{\theta,t_i} \triangleq \mathcal{A}(\mathcal{T}, \theta, t_i)$ [1].

To frame this as a bandit problem, we create an arm i for each configuration of n algorithm hyperparameters $\theta_i \in \Theta$ that we wish to explore and define its loss after t_i training iterations to be $\ell_{i,t_i} \triangleq \frac{1}{k} \sum_{j=1}^{k} \mathcal{L}(f_{\theta_i,t_i}(x_j), y_j)$ (i.e., the mean validation loss for the function output by $\mathcal{A}$ after t_i iterations of training).

Note that most common algorithms $\mathcal{A}$ have infinitely many possible hyperparameter configurations, or so many that we do not want to explore every single one, so we will only explore a subset of all possible configurations. It is usually too difficult to determine a relationship between the hyperparameter configuration of an arm and its losses, so we will again assume that the losses are independent of the configuration. Combined with the fact that we only explore a subset of all possible arms, this means that we cannot find the *best* possible configuration, and will have to settle for finding merely a very good one.

Generally, it is reasonable to assume that the loss for each arm will converge to some value over time, which we will refer to as the *limit loss*. We just want to find the arm with the best limit loss, and do not need to minimize losses incurred by other arms in the process, so this is again a pure exploration problem.

Again, the adversary is non-stochastic, since both the loss function $\mathcal{L}$ and the f_{θ,t_i} functions output by $\mathcal{A}$ are deterministic. The adversary is also oblivious: neither the loss function $\mathcal{L}$ nor the f_{θ,t_i} functions use any information about the player's actions.

1.3 Problem Statement

Now that we have some concrete examples of non-stochastic best arm identification problems, we can identify and justify the aforementioned differences from previous bandit problems we have encountered, and then we can outline this type of bandit problem.

First, although both problems are non-stochastic, our notion of a "best" arm assumes that the loss of each arm i converges to a limit loss value ν_i as it gets pulled more times (i.e., $\nu_i \triangleq \lim_{t_i\to\infty} \ell_{i,t_i}$); this means that the losses cannot be completely arbitrary as in previous non-stochastic bandit problems. This assumption is logical in the context of non-convex optimization, since gradient descent will converge to a local minimum; in hyperparameter learning, whether the losses converge will depend on the algorithm $\mathcal{A}$ used, but it generally makes sense to assume convergence.

Second, in both applications, there may be an infinite number of possible arms. In convex optimization, the arms are defined by the gradient descent starting location; since the domain of our function f is presumably continuous, there are an infinite number of possibilities. In hyperparameter optimization, the number of possible hyperparameter configurations can also be infinite, depending on the algorithm $\mathcal{A}$. Obviously, we cannot actually explore an infinite number of arms in a finite amount of time, so we must choose some finite subset of n arms to explore. The optimal number of arms is not known in general and is something we will have to find. The method of choosing n is described in section 1.4; first we will focus on the finitely-armed case, where we are given the n arms.

Additionally, there is a minor change that affects notation: losses for each arm are determined based on the number of times that specific arm has been pulled t_i, instead of the total number of arm pulls made t. In both examples, the loss of each arm i is determined by a function of some *state*, which does not change when other arms are pulled. For convex optimization, the function is f and the state is the current location $\mathbf{x}_{i,t_i}$. For hyperparameter optimization, the function is $\mathcal{A}(\mathcal{T}, \theta, \cdot)$ (i.e., a partial application of $\mathcal{A}$) and the state is simply t_i. Compare this to another example such as regret minimization when choosing the most accurate weather forecast. In that example, each weather forecast is a bandit arm, the loss is the difference from the real weather, and each day is an iteration of the bandit. When we choose an arm whose loss to observe and receive for one iteration, the losses for all arms change since both the weather and the weather forecasts will be different for the next day.

Given these differences, here is the general structure of the non-stochastic finitely-armed best arm identification problem:

Non-Stochastic Finitely-Armed Best Arm Identification

Input:

total number of pulls allowed (i.e., budget) B

number of arms to explore n

losses ℓ_{i,t_i} for each arm $i \in [n]$ after $t_i \in \mathbb{N}$ pulls, where losses converge to $\nu_i \triangleq \lim_{t_i \to \infty} \ell_{i,t_i}$

For $t = 1, 2, \cdots, B$

Player chooses $I_t \in [n]$

Player observes loss $\ell_{I_t, t_{I_t}}$, where t_{I_t} is the number of times I_t has been chosen.

Output: Player's final choice I_B and its final loss $\ell_{I_B, t_{I_B}}$

1.4 Hyperband

Hyperband is an algorithm for solving the non-stochastic infinitely-armed best arm identification problem by using an algorithm for the finitely-armed case as a subroutine [2]. The idea is to repeatedly solve the finite case while doubling the budget B or number of arms n with each iteration, such that we will eventually find a good combination without wasting too much effort.

The algorithm has no set stopping condition, since once again we can never be certain that we have found the "best" arm. We might stop the algorithm at some point when the losses $\ell_{\hat{i},\lfloor \frac{2^k-1}{l} \rfloor}$ converge as k and l increase such that the arm it has found is "good enough" (k and l as in the Hyperband algorithm below), or perhaps when the values of k and l cause the algorithm to exceed our limits for computing time and memory.

1.5 Finitely-Armed Best Arm Identification

We can also apply the Hyperband technique, created for infinitely-armed bandits, to finitely-armed bandits. Hyperband involves choosing some subset of all possible arms to explore; this can be useful for finitely-armed bandits if the number of arms is finite, but very large. By picking a subset, we can make progress on the problem with a number of pulls independent of the number of arms. (Compare this to most multi-armed bandit algorithms, such as UCB, which first pull each arm once.) Note that we can also use this process for

```
Hyperband
Assume we have some algorithm FINITE CASE to solve the finitely-armed case of the
non-stochastic best arm identification problem.
Input: non-stochastic infinitely-armed best arm identification problem with predetermined
losses for each arm
For $k = 1, 2, \dots$
    For $l \in N$ s.t. $k - l \geq \log_2(l)$
        Set budget $B_{k,l} = 2^k$
        Set number of arms $n_{k,l} = 2^l$
        Randomly choose $n_{k,l}$ arms and denote the losses $\ell_{i,t_i}$ for the $i$th arm after $t_i$ pulls
        $\widehat{i}_{k,l}, \ell_{\widehat{i},\lfloor \frac{2^k-1}{l} \rfloor} \leftarrow$ FINITE CASE ($B_{k,l}$, $n_{k,l}$, all $\ell_{i,t_i}$ for $i \in [n]$ and $t_i \in \mathbb{N}$)
Output: $\widehat{i}_{k,l}, \ell_{\widehat{i},\lfloor \frac{2^k-1}{l} \rfloor}$ for some $k, l \in \mathbb{N}$ s.t. $k - l \geq \log_2(l)$ as determined by the user
```

stochastic problems, since stochastic problems fit into the non-stochastic framework. (By definition, non-stochastic problems are a superset of stochastic problems. We can imagine an oblivious adversary whose choice of losses is equivalent to the values sampled from some random distribution.)

2 Successive Halving

To solve the finite-case problem, we will be using Successive Halving, an algorithm originally proposed for the stochastic setting [3], which happens to generalize to the non-stochastic case. As its name suggests, the algorithm functions by successively discarding the worst-performing half of arms, such that the set of arms S_k being considered on each round k halves in size until only one arm is left. In the Hyperband paper [2], the algorithm was refined to find merely an ϵ-good arm, as we will show and prove in Theorem 1. However, the original proof is very complicated, so we give a much simpler proof focusing on the finitely-armed case where the total budget B and number of arms n are given [1].

```
Successive Halving
Input:
    total number of pulls allowed (i.e., budget) $B$
    number of arms to explore $n$
    losses $\ell_{i,t_i}$ for each arm $i \in [n]$ after $t_i \in \mathbb{N}$ pulls, where losses converge to $\nu_i \triangleq \lim_{t_i \to \infty} \ell_{i,t_i}$
Initialize: $S_0 \triangleq [n] = \{1, 2, \dots, n\}$.
For $k = 0, 1, \dots, \lceil \log_2(n) \rceil - 1$
    Pull each arm in $S_k$ for $r_k = \lfloor \frac{B}{|S_t| \lceil \log_2(n) \rceil} \rfloor$ times.
    Keep the best $\lfloor |S_k|/2 \rfloor$ arms in terms of the $r_k$th observed loss as $S_{k+1}$.
Output : $\widehat{i}$, $\ell_{\widehat{i},\lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor}$ where $\widehat{i} = S_{\lceil \log_2(n) \rceil}$
```

Before we jump into the analysis of the algorithm, we will define a monotonically non-increasing envelope function $\gamma : \mathbb{N} \to \mathbb{R}$ that satisfies

$$\gamma(t) > |\ell_{i,t} - \nu_i| \, , \quad \forall \, i \in [n], \; t \in \mathbb{N} \, . \tag{1}$$

The function $\gamma(t)$ is guaranteed to exist, since the losses must converge in order for the limit losses $\nu_1, \dots, \nu_n$ to exist. We define $\gamma(t)$ as a uniform measure of the behavior and rate of convergence of the sample losses to the limit losses, and it bounds the deviation from the limit value as t increases.

Note that although our notion of "best" is based on and thus requires the existence of limit losses, we can extend the algorithm to other loss sequences by using a different metrics for "best." In essence, this involves mapping the original loss sequences $\{\{\ell_{i,t_i}\}_{i=1}^n\}_{t_i=1}^\infty$ to corresponding loss sequences $\{\{\ell'_{i,t_i}\}_{i=1}^n\}_{t_i=1}^\infty$ that *do* have limit losses. For example, if each arm has losses bounded from below, we can define the best arm to be the one with the lowest lower bound and use new losses $\ell'_{i,t_i} \triangleq \inf_{1\le j\le t_i} \ell_{i,j}$ in the algorithm.

To assess the theoretical guarantees of the proposed Successive Halving algorithm, the following theorem guarantees that if the algorithm is given enough budget and if we know the limit losses of each arm, the limit loss of the returned arm is at most ϵ away from the optimum.

Theorem 1. *Assume without loss of generality that $\nu_1 \le \cdots \le \nu_n$. For any $\epsilon > 0$ let the sufficient budget for Successive Halving z_{SH} be*

$$\begin{aligned} z_{SH} &\triangleq 2\lceil \log_2(n) \rceil \max_{i=2,\dots,n} i \left(1 + \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_1}{2}\right\}\right)\right) \\ &\le 2\lceil \log_2(n)\rceil \big(n + \sum_{i=1,\dots,n} \gamma^{-1}\left(\max\left\{\tfrac{\epsilon}{4}, \tfrac{\nu_i-\nu_1}{2}\right\}\right)\big), \end{aligned}$$

where $\gamma^{-1}(y) \triangleq \min\{t \in \mathbb{N} : \gamma(t) \le y\}$ is an inverse for the γ function.

If the Successive Halving algorithm is run with any budget $B > z_{SH}$ then an arm $\widehat{i}$ is returned that satisfies $\nu_{\widehat{i}} - \nu_1 \le \epsilon/2$. Moreover, $|\ell_{\widehat{i}, \lfloor \frac{B/2}{\lceil \log_2(n)\rceil} \rfloor} - \nu_1| \le \epsilon$.

Proof. First assume $\nu_1 < \nu_2$ so that 1 is the single best arm; the case where $\nu_1 = \nu_2$ will be covered later. We first find a lower bound for the number of pulls r_k at each stage k as a function of γ^{-1}. Then, we prove that if r_1 is sufficiently large at each stage, we never discard arm 1. Finally, we show that even if r_1 is not sufficiently large and we end up discarding arm 1, the lower bound guarantees that the arms remaining in S_{k+1} all have losses at most $\epsilon/2$ away from ν_1. Assume that $B \ge z_{SH}$. Then we have for each round k

$$\begin{aligned} r_k &= \left\lfloor \frac{B}{|S_k|\lceil \log_2(n)\rceil} \right\rfloor \\ &\ge \frac{B}{|S_k|\lceil \log_2(n)\rceil} - 1 \\ &\ge \frac{z_{SH}}{|S_k|\lceil \log_2(n)\rceil} - 1 \\ &\ge \frac{2}{|S_k|} \max_{i=2,\dots,n} i \left(1 + \gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2}\right\}\right)\right) - 1 \\ &\ge \frac{2}{|S_k|} \left(\lfloor |S_k|/2 \rfloor + 1\right) \left(1 + \gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2\rfloor+1} - \nu_1}{2}\right\}\right)\right) - 1 \\ &\ge \left(1 + \gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2\rfloor+1} - \nu_1}{2}\right\}\right)\right) - 1 \qquad (\text{since } \lfloor |S_k|/2\rfloor \ge |S_k|/2 - 1) \\ &= \gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2\rfloor+1} - \nu_1}{2}\right\}\right). \end{aligned}$$

First we show that $\ell_{i,r} - \ell_{1,r} > 0$ for $i > 1$ and any number of pulls number of pulls $r \ge \tau_i \triangleq \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right)$. By the definition of γ, we have for all $j \in [n]$ that $|\ell_{j,t_j} - \nu_j| < \gamma(t_j) \le \frac{\nu_j - \nu_1}{2}$ where the last inequality holds for $t_j \ge \tau_j$. Thus, for $r \ge \tau_i$ we have

$$\begin{aligned} \ell_{i,r} - \ell_{1,r} &= \ell_{i,r} - \nu_i + \nu_i - \nu_1 + \nu_1 - \ell_{1,r} \\ &= \ell_{i,r} - \nu_i - (\ell_{1,r} - \nu_1) + \nu_i - \nu_1 \\ &> -2\gamma(r) + \nu_i - \nu_1 \\ &\ge -2\frac{\nu_i - \nu_1}{2} + \nu_i - \nu_1 \\ &= 0. \end{aligned}$$

Note that by the assumption the ν_i limits are non-decreasing in i so that the τ_i values are non-increasing in i.

Fix a round k and assume $1 \in S_k$ (note, $1 \in S_0$). The above calculation shows that

$$r_k \geq \tau_i \implies \ell_{i,r_k} > \ell_{1,r_k} . \tag{2}$$

This means that if we pull all arms enough times, the best arm will achieve the best loss. Consequently, if the best arm is discarded in round k, we can draw the following conclusion ($\mathbb{1}\{\cdot\}$ is the indicator function):

$$\begin{aligned}
\{1 \in S_k \wedge 1 \notin S_{k+1}\} &\iff \left\{\sum_{i \in S_k} \mathbb{1}\{\ell_{i,r_k} < \ell_{1,r_k}\} \geq \lfloor |S_k|/2 \rfloor\right\} && \text{(by the definition of the algorithm)} \\
&\implies \left\{\sum_{i \in S_k \setminus \{1\}} \mathbb{1}\{r_k < \tau_i\} \geq \lfloor |S_k|/2 \rfloor\right\} && \text{(by Equation 2)} \\
&\implies \left\{\sum_{i=2}^{\lfloor |S_k|/2 \rfloor + 1} \mathbb{1}\{r_k < \tau_i\} \geq \lfloor |S_k|/2 \rfloor\right\} && \text{(for } i < j,\ \tau_i \geq \tau_j, \text{ so } \mathbb{1}\{r_k < \tau_i\} \geq \mathbb{1}\{r_k < \tau_j\}) \\
&\iff \left\{r_k < \tau_{\lfloor |S_k|/2 \rfloor + 1}\right\} .
\end{aligned}$$

This implies

$$\{1 \in S_k \wedge r_k \geq \tau_{\lfloor |S_k|/2 \rfloor + 1}\} \implies \{1 \in S_{k+1}\}. \tag{3}$$

Recalling that $r_k \geq \gamma^{-1}\big(\max\big\{\frac{\epsilon}{4}, \frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2}\big\}\big)$ and $\tau_{\lfloor |S_k|/2 \rfloor + 1} = \gamma^{-1}\big(\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2}\big)$, we examine the following three exhaustive cases:

- **Case 1:** $1 \in S_k$ and $\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2} \geq \frac{\epsilon}{4}$.

  In this case, $r_k \geq \gamma^{-1}\big(\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2}\big) = \tau_{\lfloor |S_k|/2 \rfloor+1}$. By Equation 3 we have that $1 \in S_{k+1}$ since $1 \in S_k$.

- **Case 2:** $1 \in S_k$ and $\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2} < \frac{\epsilon}{4}$.

  In this case $r_k \geq \gamma^{-1}\big(\frac{\epsilon}{4}\big)$ but $\gamma^{-1}\big(\frac{\epsilon}{4}\big) < \tau_{\lfloor |S_k|/2 \rfloor+1}$. Equation 3 suggests that it may be possible for $1 \in S_k$ but $1 \notin S_{k+1}$. In the ideal case that $1 \in S_{k+1}$, the algorithm continues, and on the next round either case 1 or case 2 could be true.

  Now consider when $1 \notin S_{k+1}$. Here we show that $\{1 \in S_k \wedge 1 \notin S_{k+1}\} \implies \forall i \in S_{k+1},\ \nu_i \leq \nu_1 + \epsilon/2$. Let $p \triangleq \min\{i \in [n] : \frac{\nu_i - \nu_1}{2} \geq \frac{\epsilon}{4}\} = \min\{i \in [n] : \nu_i \geq \nu_1 + \epsilon/2\}$, so p is the arm with the least limit loss that is not ϵ-good. Note that $p > \lfloor |S_k|/2 \rfloor + 1$ by the criteria of the case, and that

  $$r_k \geq \gamma^{-1}\left(\frac{\epsilon}{4}\right) \geq \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right) = \tau_i, \quad \forall i \geq p.$$

  Thus, by Equation 2 ($r_k \geq \tau_i \implies \ell_{i,r_k} > \ell_{1,r_k}$) we have that arms $i \geq p$ would always have $\ell_{i,r_k} > \ell_{1,r_k}$. We can see from the definition of the algorithm that if we discard an arm, we must also discard all arms with greater losses. So, if we discard arm 1 in round k, we must also discard all arms $i \geq p$, meaning that only arms $j < p$ can remain in S_{k+1}. Then, by the definition of p, for all remaining arms $i \in S_{k+1}$, $\nu_i \leq \max\limits_{i \in S_{k+1}} \nu_i \leq \max\limits_{j<p} \nu_j < \nu_1 + \epsilon/2$.

- **Case 3:** $1 \notin S_k$

  Since $1 \in S_0$, there exists some $j < k$ such that $1 \in S_j$ and $1 \notin S_{j+1}$. For this j, only case 2 is possible since case 1 would proliferate $1 \in S_{j+1}$. However, for case 2, if $1 \notin S_{j+1}$ then $\forall\, i \in\ S_{j+1},\ \nu_i \leq\ \nu_1 + \epsilon/2$.

Because $1 \in S_0$, we either have that 1 remains in S_k (possibly alternating between cases 1 and 2) for all k until the algorithm exits with the best arm 1, or there exists some k such that case 3 is true and the algorithm exits with an arm $\widehat{i}$ such that $\nu_{\widehat{i}} \leq \nu_1 + \epsilon/2$.

If $\nu_1 < \nu_2$ does not hold, i.e., if there are ties with the optimal loss, arbitrarily breaking the ties the proof above would still go through with $\nu_{\widehat{i}} \leq \nu_1 + \epsilon/2$ since the optimal loss itself is unique.

Thus we have shown that $\nu_{\widehat{i}} - \nu_1 \leq \epsilon/2$. The proof is complete by noting that

$$|\ell_{\widehat{i},\lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor} - \nu_1| \leq |\ell_{\widehat{i},\lfloor \frac{B/2}{\lceil \log_2(n) \rceil} \rfloor} - \nu_{\widehat{i}}| + |\nu_{\widehat{i}} - \nu_1| \leq \epsilon/4 + \epsilon/2 \leq \epsilon$$

by the triangle inequality and because $B \geq 2\lceil \log_2(n) \rceil \gamma^{-1}(\epsilon/4)$ by assumption. □

Remark 1. *It is apparent from the proof that in the case where $\nu_1 < \nu_2$, if we define z_{SH} without the $\frac{\epsilon}{4}$ then by definition we have $r_k \geq \gamma^{-1}\left(\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2}\right) = \tau_{\lfloor |S_k|/2 \rfloor+1}$ and we rule out Case 2 and thus Case 3, so that $B > z_{SH}$ assures that $\hat{i} = 1$.*

In order to show the Successive Halving algorithm is competitive, we prove an analogous sufficient condition for the budget for the naïve uniform budget allocation strategy, in which we pull each arm in $[n]$ for B/n times, assuming B/n is an integer for simplicity.

Theorem 2. *Assume $\nu_1 \leq \cdots \leq \nu_n$. For any $\epsilon > 0$, let the sufficient budget of uniform budget allocation strategy be*

$$\begin{aligned} z_{UB} &\triangleq \max_{i=2,\ldots,n} n\gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_i - \nu_1}{2}\right\}\right) \\ &= n\gamma^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_2 - \nu_1}{2}\right\}\right). \end{aligned}$$

If the uniform budget strategy is run with budget $B > z_{UB}$ then an arm $\hat{i}$ is returned that satisfies $\nu_{\hat{i}} - \nu_1 \leq \epsilon/2$. Moreover, $|\ell_{\hat{i},B/n} - \nu_1| \leq \epsilon$.

Proof. Assume for simplicity that B is a multiple of n. We consider two cases as in the proof for Theorem 1, and similarly we consider the case where $\nu_1 < \nu_2$ without loss of generality.

- **Case 1:** $1 \in S_k$ and $\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2} \geq \frac{\epsilon}{4}$.
  In this case, $B/n \geq \gamma^{-1}\left(\frac{\nu_2 - \nu_1}{2}\right) = \gamma^{-1} \max_{i=2,\ldots,n} \left(\frac{\nu_i - \nu_1}{2}\right)$, then by Equation 2 we have $\ell_{i,B/n} \geq \ell_{1,B/n}$ for all $i = 2, \ldots, n$ and we are done.
- **Case 2:** $1 \in S_k$ and $\frac{\nu_{\lfloor |S_k|/2 \rfloor+1} - \nu_1}{2} < \frac{\epsilon}{4}$.
  In this case, define $p \triangleq \min\{i \in [n] : \frac{\nu_i - \nu_1}{4}\} = \min\{i \in [n] : \nu_i - \nu_1 \geq \epsilon/2\}$, then
  $$B/n \geq \gamma^{-1}\left(\frac{\epsilon}{4}\right) \geq \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right), \quad \forall i \geq p.$$
  Thus, by Equation 2 again we have that arms $i \geq p$ would always have $\ell_{i,B/n} \geq \ell_{1,B/n}$. For $i < p$, $\nu_i < \nu_1 + \epsilon/2$ by definition of p.

Thus we have shown that $\nu_{\hat{i}} - \nu_1 \leq \epsilon/2$. The rest of the proof is similar to that for Theorem 1. □

Remark 2. *Again, similar to Remark 1, in the case where $\nu_1 < \nu_2$ if we define z_{UB} without the $\frac{\epsilon}{4}$ term then we can rule out Case 2, so that $B > z_{UB}$ assures that $\hat{i} = 1$.*

Since Theorem 2 a sufficiency statement, it is unclear how it actually compares to the performance of the Successive Halving algorithm. We now prove a fact which shows that the above result is tight in a worst-case sense, i.e., if the budget for uniform allocation is insufficient we can always find a setting where the arm returned has limit loss more than $\epsilon/2$ away from the optimum.

Theorem 3. *Let $\hat{i}$ be the arm returned by the uniform budget allocation strategy. Given $\{\nu_i\}_{i=1}^n$ and $\{\{\ell_{i,t_i}\}_{i=1}^n\}_{t_i=1}^\infty$, let $\gamma_0 : \mathbb{N} \to \mathbb{R}_+$ be defined as $\gamma_0(t) \triangleq \max_i |\ell_{i,t} - \nu_i|$. Then for any $\nu_1 \le \nu_2 \le \dots \nu_n$ there exist sequences of losses $\{\{\ell_{i,t_i}\}_{i=1}^n\}_{t_i=1}^\infty$ such that whenever*

$$B < z_{UB} = n\gamma_0^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_2 - \nu_1}{2}\right\}\right),$$

we always have $\lim_{t\to\infty} \ell_{i,t_i} = \nu_i$ and $\hat{i} \neq 1$; furthermore, $\nu_{\hat{i}} - \nu_1 > \epsilon/2$ if $\epsilon < 2(\nu_2 - \nu_1)$.

Proof. Let $\beta(t) : \mathbb{N} \to \mathbb{R}_+$ be an arbitrary monotonically decreasing function of t with $\lim_{t\to\infty} \beta(t) = 0$, and define $\ell_{1,t_1} \triangleq \nu_1 + \beta(t_1)$ and $\ell_{i,t_i} \triangleq \nu_i - \beta(t_i)$ for all $i = 2, \dots, n$, so $\gamma_0(t) = \beta(t)$. Thus, for any $\epsilon > 0$ we have

$$\begin{aligned}
\hat{i} = 1 \quad &\Leftrightarrow \quad \ell_{1,B/n} < \min_{i=2,\dots,n} \ell_{i,B/n} \\
&\Leftrightarrow \quad \nu_1 + \beta(B/n) < \min_{i=2,\dots,n} \nu_i - \beta(B/n) \\
&\Leftrightarrow \quad \nu_1 + \beta(B/n) < \nu_2 - \beta(B/n) \\
&\Leftrightarrow \quad \beta(B/n) < \frac{\nu_2 - \nu_1}{2} \\
&\Rightarrow \quad B \ge n\gamma_0^{-1}\left(\frac{\nu_2 - \nu_1}{2}\right) \ge n\gamma_0^{-1}\left(\max\left\{\frac{\epsilon}{4}, \frac{\nu_2 - \nu_1}{2}\right\}\right) = z_{UB}. \qquad (4)
\end{aligned}$$

Since this contradicts the assumption, $\hat{i} \neq 1$. Since at any time stamp t, for $i = 2, \dots, n$ the loss $\ell_{i,t}$ is just ν_i translated by a common constant, we always have $\ell_{2,t} \le \dots \le \ell_{n,t}$, and thus $\hat{i} \in \{j : \nu_j = \nu_2\}$. We then have

$$\nu_{\hat{i}} - \nu_1 \le \epsilon/2 \quad \Leftrightarrow \quad \nu_2 - \nu_1 \le \epsilon/2.$$

For $\epsilon < 2(\nu_2 - \nu_1)$ the last display does not hold and the proof is complete. □

Remark 3. *By inverting (4) we see that under the same setting, we always have $\hat{i} \neq 1$ whenever $B < n\gamma_0^{-1}((\nu_2 - \nu_1)/2)$, as a partial inverse to Remark 2.*

Remark 4. *Here we proved the theorem using γ_0; but by definition, $\gamma(t) > \max_i |\ell_{i,t} - \nu_i|$ while $\gamma_0(t) = \max_i |\ell_{i,t} - \nu_i|$, so for any $\epsilon > 0$ we can find some γ that is at most ϵ larger than γ_0 at all t, making γ and γ_0 virtually no different in practice.*

From the looser bound of z_{SH} in Theorem 1 we see that the sufficient number of pulls for the Successive Halving algorithm behaves roughly like $(n-1)\log_2(n)$ times the average $\frac{1}{n-1}\sum_{i=2}^n \gamma^{-1}\left(\frac{\nu_i - \nu_1}{2}\right)$, while the necessary condition in Theorem 3 behaves like n times the maximum $\max_{i=2,\dots,n} \gamma_0^{-1}\left(\frac{\nu_i - \nu_1}{2}\right)$. While γ and γ_0 can be arbitrarily close as per Remark 4, the difference between this average and max can be very significant, e.g. in the case of least squares problems solved with stochastic gradient descent.

In practice, it can be difficult to determine the γ function, and without knowledge of the ν_i's, there is no way to determine the sufficient budget in Theorem 1 a priori. But on the other hand, since the Successive Halving algorithm is agnostic to the γ function, it is adaptive to it, in the sense that the best arm can be discovered faster if the losses converge faster. It also has the advantage that γ is not explicitly needed as an input to the algorithm.

Without knowledge of γ and the ν_i's we cannot determine the relationship between $\nu_{\hat{i}} - \nu_1$ and B beforehand, but the next Theorem guarantees that the Successive Halving algorithm is not much worse than the uniform allocation strategy.

Theorem 4. *For a budget B and set of n arms, define $\hat{\imath}_{SH}$ and $\hat{\imath}_{UB}$ as the unique output arms of the Successive Halving algorithm and the uniform strategy, respectively. Then*

$$\begin{aligned}\nu_{\hat{\imath}_{SH}} - \nu_1 &\leq \lfloor \log_2(n) \rfloor 2\gamma \left(\left\lfloor \frac{B}{n\lceil \log_2(n) \rceil} \right\rfloor \right),\\ \nu_{\hat{\imath}_{UB}} - \nu_1 &\leq \ell_{\hat{\imath}_{UB},B/n} - \ell_{1,B/n} + 2\gamma(B/n) \leq 2\gamma(B/n).\end{aligned}$$

Proof. Since $\arg\min_{i\in S_0} \nu_i = 1$ and $S_{\lceil \log_2(n)\rceil} = \{\hat{\imath}_{SH}\}$ (since half of the arms are pruned in each round),

$$\begin{aligned}\nu_{\hat{\imath}_{SH}} - \nu_1 &= \min_{i\in S_{\lceil \log_2(n)\rceil}} \nu_i - \min_{i\in S_0} \nu_i\\ &= \sum_{k=0}^{\lceil \log_2(n)\rceil - 1} \left(\min_{i\in S_{k+1}} \nu_i - \min_{i\in S_k} \nu_i \right)\\ &\leq \sum_{k=0}^{\lceil \log_2(n)\rceil - 1} \left(\min_{i\in S_{k+1}} \ell_{i,r_k} - \min_{i\in S_k} \ell_{i,r_k} + 2\gamma(r_k) \right) \qquad (5)\\ &= \sum_{k=0}^{\lceil \log_2(n)\rceil - 1} 2\gamma(r_k)\\ &\leq \lceil \log_2(n) \rceil 2\gamma \left(\left\lfloor \frac{B}{n\lceil \log_2(n) \rceil} \right\rfloor \right). \qquad (6)\end{aligned}$$

Note that (5) is obvious by definitions of ν and γ, and the fact that we always keep the arm with the smallest current loss, and that (6) follows immediately from a trivial lower bound for r_k in each step. □

The above theorem is a fall-back guarantee for the Successive Halving algorithm and concludes our analysis of its performance.

References

[1] Kevin G. Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. *CoRR*, abs/1502.07943, 2015.

[2] Lisha Li, Kevin G. Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Efficient hyperparameter optimization and infinitely many armed bandits. *CoRR*, abs/1603.06560, 2016.

[3] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1238–III–1246. JMLR.org, 2013.