BAI with Limited Memory

Richard Xu, Moses Mayer
(raxu,mosesmayer)@college.harvard.edu

September 24, 2021

1 Introduction

Note. This note is designed to be a roadmap and to keep you company during the talk. They are not a full exposition of what we will talk about, and does not contain most of the mathematical detail for the algorithms.

We consider streaming algorithms for the Best-Arm Identification problem. That is,

1. Arms arrive one by one.
2. We need to store arms in memory, and once we throw out an arm we cannot use it again.
3. Aim to minimize memory use (streaming complexity) *and* number of times we pull the arms (sample complexity).

Exercise 1. What is the streaming / sample complexity of 1. the naive algorithm in lecture 1, and 2. the median elimination algorithm?

Exercise 2. What are some applications of streaming BAI?

1.1 Streaming vs. Space

We talk about "memory" when discussing streaming BAI algorithm, but this is not quite space complexity. Fortunately, many algorithms (and, algorithms we've studied) tend to need $\Theta(1)$ memory per arm it might pull in the future. So, in many cases "streaming complexity" = "space complexity".

1.2 BAI vs. ϵ-BAI vs. ϵ-KAI

The papers we present consider two different problem:

1. BAI, where you want to find the *best* arm with high probability, and that arm is known to be Δ better than the second best.
2. ϵ-BAI, where you get no guarantee and need an arm that's $\leq \epsilon$ worse than the best arm.

Exercise 3. One of these problems imply the other. Which?

2 BAI When Gap Exists

Note. The paper uses the phrase "coins", considering contexts where the arm is a weighted coin. The notes below will use arms instead, but the two scenarios are virtually identical for this algorithm.

We will review the goals above, and present the algorithm in AW. For notation sake, let $M = \frac{4}{\Delta^2}\ln(1/\delta)$. The algorithm is as follows:

1. Choose the first arm as "king". The remaining arm will "challenge" and perhaps dethrone the king.

2. When a challenger comes, we increase the king's budget by CM where $C > 100$ is some constant.

3. Use each arm $3M$ times (level 1). If king loses, do it for $9M$ times (level 2), then $27M$ times, etc. until the king cannot afford the number of uses. In that case the challenger dethroned the king.

4. If challenger loses, we throw it away.

Exercise 4. What does it mean for a king/challenger to lose?

Efficiency of algorithm: streaming complexity is 1, use amortized argument to see that the sample complexity is optimal.

2.1 Correctness Proof: Two Parts

It suffices to show that

1. The best arm will become king with probability $\geq 1 - \delta/2$,

2. The best arm will stay king with probability $\geq 1 - \delta/2$.

Recall from class (Hoeffding) that if we compare the best arm against another K/Δ^2 times, then $P(\text{worse arm wins}) \leq 2\exp(-K/4)$.

Another inequality we will use is Bernstein's inequality. We say that X is sub-exponential with parameter κ if $P(X \geq t) \leq 2\exp(-t/\kappa)$. Bernstein's inequality says that if $X_1 \dots X_m$ are all sub-exponential with parameter κ, then

$$P(\sum X_i \geq t) \leq 2\exp(-c\min(t^2/(\kappa^2 m), t/\kappa)).$$

We will write the details on the board if time permits.

Proof that best arm will become king. Note that the best arm loses the level l challenge with probability $2\delta\exp(-3^l)$. Summing over all l to find that the series converges to $< \delta/2$.

Proof that best arm will stay king. Harder: need to keep track of budget.

Consider the challengers, and let X_i' be the number of arm use in challenger i, divided by M, and let $Y_i' = \sum_{j=2}^{i} X_i'$.

Note that most challengers lose after round 1, so $E(X_i') < 10$. Then, we notice that X_i' is a subexponential random variable and apply Bernstein's inequality.

This helps us determine that the best arm runs out of budget on challenger i with exponentially small probability, and another union bound finishes the problem.

3 ϵ-BAI in General

We will first review the setup of the general ϵ-BAI problem.

3.1 Why does our previous approach not work?

Notice that the best arm is likely to be dethroned when challenged by 10 arms that are say $\epsilon/10$ worse than it.

So, we can use progressively worse challengers until we end up with a king that is more than ϵ weaker.

3.2 JHTX algorithm

We will draw the algorithm on the board. Here are the main ideas:

1. Add a gap of α, randomized between $\epsilon/4$ and $\epsilon/2$.
2. No more "budget" for the king.
3. We also don't resample the king.
4. If a challenger wins by less than α, we throw it away. Otherwise, we keep going until we go for $\log j^2 \cdot M$ challenges.

3.3 Why do we need a gap?

Exercise 5. Intuitively, how does having a gap resolve the bad example we have in the earlier section?

We will go through the analysis for the bad example above, and give some general intuition for the correctness proof.

3.4 Runtime: why we need to randomize α?

Exercise 6. Suppose we pick $\alpha = \epsilon/4$ always. Can you think of a situation where the sample complexity can be too big?

Having a random α solves the issue. This is a little subtle, so bear with us.

Runtime analysis. Suppose the current arm has empirical mean $\hat{\mu}$, we call arms whose means are at most $3\epsilon/8$ better "weak arms", and arms whose means are at least $3\epsilon/8$ better "strong arms".

When might we end up rolling the arm $\log(j^2)M$ times?

1. When the arm is weak, and we set $\alpha = \epsilon/4$. Note that this happens with probability $1/\log(j) = 2/\log(j^2)$ times, so this will not change the expectation.
2. When the arm is strong, and we set $\alpha = \epsilon/2$.

   Note that $\sum 1/\log(j)$ diverges, so after $polylog(j)$ strong arms we will set $\alpha = \epsilon/4$ once with high probability (at which point that arm will replace the old arm).

   Therefore, this case should not happen too often!

4 Summary

In this talk, we saw

1. The idea of a streaming BAI algorithm and some of its applications,
2. An algorithm that, through a multi-level challenging scheme, achieves the optimal streaming and sample complexity,
3. How that algorithm fails in the more general case of the ϵ-BAI problem,
4. How, by having a randomized "gap" parameter, we can create an optimal algorithm in the general case.

4.1 Potential Research Topics

If you are interested in looking more into this topic, here are some potential research areas:

1. Context-dependent complexity. There are some "easy" cases of the BAI problem where we know about offline algorithms with lower sample complexity. Can we find streaming algorithms that match those complexities?
2. Regret minimization: you can also study streaming bandits for regret minimization.
3. Practical analysis of the algorithm above. The second algorithm, in particular, makes many design choices to simplify their analysis. What happens when we change the design of the algorithm?