

# **Introduction to Multi-Armed Bandits**

(preliminary and incomplete draft)

Aleksandrs Slivkins

Microsoft Research NYC

<https://www.microsoft.com/en-us/research/people/slivkins/>

Last edits: Jan 25, 2017



# Preface

Multi-armed bandits is a rich area, multi-disciplinary area studied since (Thompson, 1933), with a big surge of activity in the past 10-15 years. An enormous body of work has accumulated over the years. While various subsets of this work have been covered in depth in several books and surveys (Berry and Fristedt, 1985; Cesa-Bianchi and Lugosi, 2006; Bergemann and Välimäki, 2006; Gittins et al., 2011; Bubeck and Cesa-Bianchi, 2012), this book provides a more textbook-like treatment of the subject.

The organizing principles for this book can be summarized as follows. The work on multi-armed bandits can be partitioned into a dozen or so lines of work. Each chapter tackles one line of work, providing a self-contained introduction and pointers for further reading. We favor fundamental ideas and elementary, teachable proofs over strongest possible results with very complicated proofs. We emphasize accessibility of the material: while exposure to machine learning and probability/statistics would certainly help, no advanced background is required beyond a standard undergraduate course on algorithms, *e.g.*, one based on (Kleinberg and Tardos, 2005).

The book is based on lecture notes from a graduate course at University of Maryland, College Park, taught by the author in Fall 2016. In fact, each chapter corresponds to a week of the course. The choice of topics and results to present is based on the author's understanding of what is important and teachable, and is necessarily subjective, to some extent. In particular, many important results has been deemed too technical/advanced to be presented in detail.

To keep the book manageable, and also more accessible, we chose not to dwell on the deep connections to online convex optimization. A modern treatment of this fascinating subject can be found, *e.g.*, in the recent textbook (Hazan, 2015). Likewise, we chose not venture into a much more general problem space of reinforcement learning, a subject of many graduate courses and textbooks such as (Sutton and Barto, 1998; Szepesvári, 2010). A course based on this book would be complementary to graduate-level courses on online convex optimization and reinforcement learning.

**Status of the manuscript.** The present draft needs some polishing, and, at places, a more detailed discussion of related work. (However, our goal is to provide pointers for further reading rather than a comprehensive discussion.) In addition to the nine chapters already in the manuscript, the author needs to add an introductory chapter on scope and motivations, and plans to add chapters covering connections to mechanism design and game theory (Chapters 11, 12, resp.), an “interlude” on practical aspects on bandit algorithms (Interlude B), and a section on contextual bandits as a system (Section 9.6). All the above is based on lectures from the course.

Several important topics have been omitted from the course, and therefore from the current draft, due to schedule constraints. Time permitting, the author hopes to include some of these topics in the near future: most notably, a chapter on the MDP formulation of bandits and the Gittins' algorithm. In the meantime, the author would be grateful for feedback and is open to suggestions.

**Acknowledgements.** The author is indebted to the students who scribed the initial versions of the lecture notes. Presentation of some of the fundamental results is heavily influenced by online lecture notes in (Kleinberg, 2007). The author is grateful to Alekh Agarwal, Bobby Kleinberg, Yishay Mansour, and Rob Schapire for discussions and advice.

# Contents

<b>1</b>	<b>Scope and motivations (to be added)</b>	<b>1</b>
<b>2</b>	<b>Bandits with IID Rewards</b>	<b>3</b>
2.1	Model and examples . . . . .	3
2.2	Simple algorithms . . . . .	4
2.3	Advanced algorithms . . . . .	6
2.4	Remarks . . . . .	11
<b>3</b>	<b>Lower Bounds</b>	<b>13</b>
3.1	KL-divergence (brief intro) . . . . .	13
3.2	A simple example: flipping one coin . . . . .	17
3.3	Flipping several coins: "bandits with prediction" . . . . .	17
3.4	From "bandits with predictions" to regret . . . . .	20
3.5	Per-instance lower bound (no proofs) . . . . .	21
3.6	Bibliographic notes . . . . .	22
	<b>Interlude A: Bandits with Initial Information</b>	<b>23</b>
<b>4</b>	<b>Thompson Sampling</b>	<b>25</b>
4.1	Bayesian bandits: preliminaries and notation . . . . .	25
4.2	Thompson Sampling: definition and characterizations . . . . .	26
4.3	Computational aspects . . . . .	27
4.4	Example: 0-1 rewards and Beta priors . . . . .	27
4.5	Example: Gaussian rewards and Gaussian priors . . . . .	28
4.6	Bayesian regret . . . . .	29
4.7	Thompson Sampling with no prior (and no proofs) . . . . .	31
	<b>Interlude B: Practical Aspects (to be added)</b>	<b>33</b>
<b>5</b>	<b>Lipschitz Bandits</b>	<b>35</b>
5.1	Continuum-armed bandits . . . . .	35
5.2	Lipschitz MAB . . . . .	38
5.3	Adaptive discretization: the Zooming Algorithm . . . . .	40
5.4	Remarks . . . . .	44
<b>6</b>	<b>Full Feedback and Adversarial Rewards</b>	<b>45</b>
6.1	Warm-up: full feedback and i.i.d. rewards . . . . .	46
6.2	Full feedback and adversarial rewards . . . . .	46
6.3	Majority vote algorithm . . . . .	48
6.4	Binary Prediction with Expert Advice: Weighted Majority Algorithm . . . . .	48
6.5	Hedge Algorithm . . . . .	50
6.6	Bibliographic remarks . . . . .	53

<b>7</b>	<b>Adversarial Bandits</b>	<b>55</b>
7.1	Recap: best-expert problem . . . . .	55
7.2	Reduction: from bandit feedback to full feedback . . . . .	56
7.3	Adversarial bandits with expert advice . . . . .	56
7.4	Preliminary analysis: unbiased estimates . . . . .	57
7.5	Algorithm Exp4 and crude analysis . . . . .	58
7.6	Improved analysis . . . . .	59
7.7	Remarks on Exp4 . . . . .	60
7.8	Extensions of adversarial bandits . . . . .	61
7.9	Bibliographic notes . . . . .	61
<b>8</b>	<b>Bandits/Experts with Linear Costs</b>	<b>63</b>
8.1	Recap from last lecture . . . . .	63
8.2	The Online Routing Problem . . . . .	64
8.3	Combinatorial (semi-)bandits . . . . .	65
8.4	Best-expert problem with linear costs: Follow the Perturbed Leader . . . . .	67
<b>9</b>	<b>Contextual Bandits</b>	<b>71</b>
9.1	Problem statement and examples . . . . .	71
9.2	Small number of contexts . . . . .	72
9.3	Lipshitz Contextual bandits (a simple example) . . . . .	72
9.4	Linear Contextual Bandits (LinUCB algorithm without proofs) . . . . .	74
9.5	Contextual Bandits with a Known Policy Class . . . . .	74
9.6	Contextual bandits as a system (to be written, based on (Agarwal et al., 2016)) . . . . .	77
<b>10</b>	<b>Bandits with Knapsacks</b>	<b>79</b>
10.1	Motivating Example: Dynamic Pricing . . . . .	79
10.2	General Framework: Bandits with Knapsacks (BwK) . . . . .	79
10.3	Regret bounds . . . . .	81
10.4	Fractional relaxation . . . . .	82
10.5	“Clean event” and confidence regions . . . . .	83
10.6	Three algorithms for BwK (no proofs) . . . . .	83
10.7	Bibliographic notes . . . . .	85
<b>11</b>	<b>Bandits and Incentives (lecture to be written up)</b>	<b>87</b>
<b>12</b>	<b>Bandits and Games (lecture to be written up)</b>	<b>89</b>
<b>13</b>	<b>MDP bandits and Gittins Algorithm (to be written)</b>	<b>91</b>
	<b>Bibliography</b>	<b>93</b>



# Chapter 1

## Scope and motivations (to be added)

[as: To be written. For now, see slides at  
<http://www.cs.umd.edu/~slivkins/CMSC858G-fall16/Lecture1-intro.pdf>]





## Chapter 2

# Bandits with IID Rewards

Last lecture we had a probability theory recap, including concentration of measure inequalities. Today, we're going to cover the basic model of multi-arm bandits with i.i.d rewards. We will introduce several techniques for solving the problem, and analyze their performance in terms of regret guarantees.

### 2.1 Model and examples

In the multi-arm bandit problem with iid rewards, the learner selects an arm  $a \in \mathcal{A}$  at every time step  $t$ . Learning proceeds in rounds, and we assume that the number of rounds is fixed, and indexed by  $t = 1 \cdots T$ . At each round, the algorithm chooses one action  $a_t$  (we'll use the arms and actions interchangeably to mean the same thing). After taking the action, a reward for this action is realized and observed by the algorithm. The process is repeated until the end of time horizon  $T$  is reached. The goal of the algorithm is to gather as much commulative reward as possible.

It's important to emphasize that the algorithm observes only the reward for the selected action, not all the other actions that could have been selected, that is why it is called a bandit feedback setting.

There is also the iid assumption, where the reward for each action is assumed to be i.i.d (independent and identically distributed). More precisely, for each action  $a$ , there is a distribution  $\mathcal{D}_a$  over real numbers (called "reward distribution"); for simplicity, all rewards will be in the interval  $[0, 1]$ . Every time this action is chosen, the reward is sampled independently from this distribution. It is crucial that this distribution is unknown to the algorithm, and does not change over time.

Notation: the mean (expected) reward of action  $a$  is denoted  $\mu(a)$ ; the time horizon is  $T$ ; the number of arms is  $K$ . Actions are denoted  $a$ . Let's use these conventions throughout the course.

Perhaps the simplest reward distribution is the Bernoulli distribution, when the reward of each arm  $a$  can be either 0 or 1. ("Success or failure", "heads or tails".) This distribution is parameterized by the probability of success  $\mu(a)$ , which also defines the mean of the distribution. Note that the problem instance is then completely described by the time horizon  $T$  and the vector  $(\mu(a) : a \in \mathcal{A})$ .

#### 2.1.1 Some (stylized) examples for MAB with IID rewards

1. **News:** in a very stylized news application, a user visits a news site, the site presents it with a header for an article, and a user either clicks on this header or not. The goal of the website is to maximize #clicks. So each possible header is an arm in a bandit problem, and clicks are the rewards. Note that the rewards are 0-1.

A typical modeling assumption is that each user is drawn independently from a fixed distribution over users, so that in each round the click happens independently with a probability that depends only on the chosen header.

2. **Selecting Ads:** In website advertising, a user visits a webpage, and a learning algorithm has to select one of many possible ads to display. In this setting, each ad can be considered an arm. The ad is displayed, it is observed whether or not the user clicks on the ad. If the ad  $a$  is clicked, the advertiser pays some amount  $v_a \in [0, 1]$  which is fixed in advance and known to the algorithm. (This amount depends only on the chosen ad, but does not change over time.) So the paid amount is considered as the observed reward.

Note that here rewards can be arbitrary numbers, but for each arm  $a$ , the reward can take only two values: 0 or  $v_a$ .

3. **Medical Trials:** a patient visits a doctor and the doctor can proscribe one of several possible treatments, and observes the treatment effectiveness. Then the next patient arrives, and so forth. For simplicity of this example, the effectiveness of a treatment is quantified as a number in  $[0, 1]$ . So here each treatment can be considered as an arm, and the reward is defined as the treatment effectiveness.

Note that for each arm, the reward can, in principle, take arbitrarily many different values.

### 2.1.2 Regret

While algorithm’s goal is to maximize reward, we have to be more precise about defining what it means. There are several notions for defining reward maximization. One standard notion is **regret**; we will use this notion (or versions thereof) throughout most of this course.

To define regret, we will look at two quantities: the reward accumulated by the best arm, and the average reward accumulated by the algorithm, we define the difference between the two quantities to be the regret:

$$R(t) = \mu^* \times t - \sum_{s=1}^t \mu(a_s), \tag{2.1}$$

where  $R(t)$  is the regret after  $t$ -time steps, and  $\mu^* = \max_{a \in \mathcal{A}} \mu(a)$  is the expected reward for the best arm. Note that the arm  $a_t$  chosen by the algorithm is a random quantity, as it may depend on the (random) rewards and also possibly on the internal randomness of the algorithm. Hence, we will typically talk about “expected regret”  $\mathbb{E}[R(T)]$ .

So why is it called regret? Because it is how much the algorithm “regrets” not knowing what is the best arm!

*Remark 2.1.* One can view the  $\mu^* \times t$  term as a ‘benchmark’ to which the performance of the algorithm is compared. Hence, it is sometimes called the “best arm” benchmark. While it is the standard benchmark in most work on bandits, in some settings other benchmarks make sense (and sometimes make *more* sense).

*Remark 2.2 (Terminology).* Since our definition of regret sums over all rounds, we will sometimes call it *cumulative* regret. When/if we need to highlight the distinction between  $R(T)$  and  $\mathbb{E}[R(T)]$ , we will say *realized regret* and *expected regret*; but most of the time, we will just say “regret” and the meaning will be clear from the context. The quantity  $\mathbb{E}[R(T)]$  is sometimes called *pseudo-regret* in the literature.

Regret can depend on many different parameters, we care mainly about the dependence on the time horizon, number of arms, and the average reward for each arm ( $\mu$ ). We will usually use big-O notation to focus on the growth rate dependence on the different parameters instead of keeping track of all the constants.

## 2.2 Simple algorithms

We cover several algorithms for solving the multi-arm bandit problem. For the simplicity of presentation, we start with the simple case where we have only two arms with zero/one rewards. Later, we extend the algorithms for more than two arms with bounded rewards.

### 2.2.1 Explore-First

- 1 Exploration phase: try each arm  $N$  times;
- 2 Select the arm  $a^*$  with the highest average reward (break ties arbitrarily);
- 3 Exploitation phase: play arm  $a^*$  in all remaining rounds.

**Algorithm 1:** Explore-First

Algorithm 1 describes the explore-first algorithm: we explore for  $N$  time steps, and then select the arm with the highest average reward. The parameter  $N$  is fixed in advance; it will be chosen later in the analysis as function of the time horizon and #arms. In the remainder of this subsection, we analyze the algorithm in terms of regret guarantees.

Let the average reward for each action  $a$  after exploration phase be denoted  $\bar{\mu}(a)$ . We want the average reward to be a good estimate of the true expected rewards, i.e. the following quantity should be small:  $|\bar{\mu}(a) - \mu(a)|$ . We can use the Hoeffding inequality from last lecture to quantify the deviation of the average from the true expectation. By defining the confidence radius  $r(a) = \sqrt{\frac{2 \log T}{N}}$ , and using Hoeffding inequality, we get:

$$\Pr \{ |\bar{\mu}(a) - \mu(a)| \leq r(a) \} \geq 1 - \frac{1}{T^4} \quad (2.2)$$

So, the probability that the average will deviate from the true expectation is very small.

We define the *clean event* to be the event that (2.2) holds for both arms simultaneously. We will argue separately the clean event, and the “bad event” – the complement of the clean event.

*Remark 2.3.* With this approach, one does not need to worry about probability in the rest of the proof. Indeed, the probability has been taken care of by defining the clean event and observing that (2.2) holds! And we do not need to worry about the bad event either — essentially, because its probability is so tiny.

We will use this “clean event” approach in many other proofs, to help simplify the technical details. The downside is that it usually leads to worse constants that can be obtained with a more technical proof that argues about probabilities more carefully.

Let us start with the clean event. We will show that if we chose the worse arm, it is not so bad because the expected rewards for the two arms would be close.

Let the best arm be  $a^*$ , and suppose we choose the other arm  $a \neq a^*$ . But why did we choose arm  $a$ ? This must have been because its average reward was better than that of  $a^*$ ; in other words,  $\bar{\mu}(a) > \bar{\mu}(a^*)$ . Since this is a clean event, we have:

$$\mu(a) + r(a) \geq \bar{\mu}(a) > \bar{\mu}(a^*) \geq \mu(a^*) - r(a^*)$$

Re-arranging the terms, it follows that

$$\mu(a^*) - \mu(a) \leq r(a) + r(a^*) = O\left(\sqrt{\frac{\log T}{N}}\right).$$

Thus, each round in the exploitation phase contributes at most  $O\left(\sqrt{\frac{\log T}{N}}\right)$  to regret. And each round in exploration trivially contributes at most 1. So we can derive an upper bound on the regret. This regret bound consists of two parts: for the first  $N$  rounds of exploration, and then for the remaining  $T - 2N$  rounds of exploitation.

$$\begin{aligned} R(T) &\leq N + O\left(\sqrt{\frac{\log T}{N}} \times (T - 2N)\right) \\ &\leq N + O\left(\sqrt{\frac{\log T}{N}} \times T\right). \end{aligned}$$

Since we can select any value for  $N$  (as long as it is known to the algorithm before the first round), we can optimize the right-hand side to get the tightest upper bound. Noting that the two summands are, resp., monotonically increasing and monotonically decreasing in  $N$ , we set  $N$  so that they are (approximately) equal. For  $N = T^{2/3}$ , we get the following:

$$\begin{aligned} R(T) &\leq T^{2/3} + O\left(\sqrt{\frac{\log T}{T^{2/3}}} \times T\right) \\ &\leq O(\sqrt{\log T} \times T^{2/3}). \end{aligned}$$

To complete the proof, we have to analyze the bad event case. Since regret can be at most  $T$  (because each round contributes at most 1), and the bad event happens with a very small probability ( $1/T^4$ ), the (expected) regret from this case can be neglected. Formally,

$$\mathbb{E}[R(T)] = \mathbb{E}[R(T)|\text{clean event}] \times \Pr[\text{clean event}] + \mathbb{E}[R(T)|\text{bad event}] \times \Pr[\text{bad event}] \quad (2.3)$$

$$\leq \mathbb{E}[R(T)|\text{clean event}] + T \times O(T^{-4}) \quad (2.4)$$

$$\leq O(\sqrt{\log T} \times T^{2/3}). \quad (2.5)$$

This completes the proof for  $K = 2$  arms.

For  $K > 2$  arms, we have to apply the union bound for (2.2) over the  $K$  arms, and then follow the same argument as above. Note that the value of  $T$  is greater than  $K$ , since we need to explore each arm at least once. For the final regret computation, we will need to take into account the dependence on  $K$ : specifically, the confidence radius is now  $r(a) = \sqrt{\frac{2 \log T}{N/K}}$ . Working through the proof, we obtain  $R(T) \leq N + O(\sqrt{\frac{\log T}{N/K}} \times T)$ . Plugging in  $N = T^{2/3} \times O(K \log T)^{1/3}$ , and completing the proof same way as in (2.3), we obtain:

**Theorem 2.4.** *Explore-first algorithm achieves regret  $\mathbb{E}[R(T)] \leq T^{2/3} \times O(K \log T)^{1/3}$ , where  $K$  is the number of arms.*

## 2.2.2 Epsilon Greedy

One problem with Explore-first is that the “losses” are concentrated in the initial exploration phase. It may be better to have a more uniform exploration over time. This is done in the epsilon-greedy algorithm.

<p><b>input</b>     : Exploration probability <math>\epsilon</math></p> <ol style="list-style-type: none"> <li>1 Toss a coin with probability of success = <math>\epsilon</math>;</li> <li>2 <b>if success then</b></li> <li>3     explore: choose an arm uniformly at random</li> <li>4 <b>else</b></li> <li>5     exploit: choose the arm with the highest average reward so far</li> </ol>
---

### Algorithm 2: Epsilon-Greedy

Note that the exploration is uniform, which is similar to the “round-robin” exploration in explore-first. Choosing the best option in the short term is often called the “greedy” choice in the computer science literature, hence the name “epsilon-greedy”.

The analysis for this algorithm may appear on the homework.

Both exploration-first and epsilon-greedy have a big flaw that the exploration schedule does not depend on the history of the observed rewards. Whereas it is usually better to *adapt* exploration to the observed rewards. Informally, we refer to this distinction as *adaptive vs non-adaptive* exploration. In the remainder of this class, we will talk about two algorithms that implement adaptive exploration and achieve better regret.

## 2.3 Advanced algorithms

So far we’ve discussed non-adaptive exploration strategies. Now let’s talk about adaptive exploration, in a sense that the bandit feedback of different arms in previous rounds are fully utilized.

Let’s start with 2 arms. One fairly natural idea is to alternate them until we find that one arm is much better than the other, at which time we abandon the inferior one. But how to define “one arm is much better” exactly?

### 2.3.1 Clean event and confidence bounds

To flesh out the idea mentioned above, and to set up the stage for some other algorithms in this class, let’s do some Probability with our old friend Hoeffding Inequality (HI).

Let  $n_t(a)$  be the number of samples from arm  $a$  in round  $1, 2, \dots, t$ ;  $\bar{\mu}_t(a)$  be the average reward of arm  $a$  so far. We would like to use HI to derive

$$\Pr(|\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)) \geq 1 - \frac{2}{T^4}, \quad (2.6)$$

where  $r_t(a) = \sqrt{\frac{2 \log T}{n_t(a)}}$  is the confidence radius, and  $T$  is the time horizon.

In the intended application of HI, we have  $n_t(a)$  independent random variables — one per each sample of arm  $a$ . Since HI requires a fixed number of random variables, (2.6) would follow immediately if  $n_t(a)$  were fixed in advance. However,  $n_t(a)$  is itself a random variable. So we need a slightly more careful argument, presented below.

Let us imagine there is a tape of length  $T$  for each arm  $a$ , with each cell independently sampled from  $\mathcal{D}_a$ , as shown in Figure 2.1.

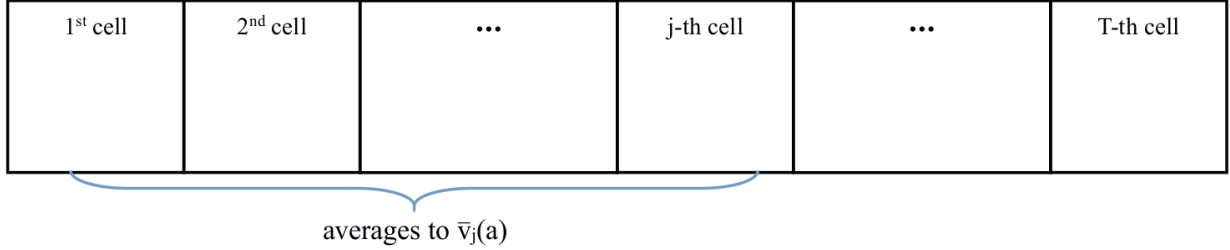


Figure 2.1: the  $j$ -th cell contains the reward of the  $j$ -th time we pull arm  $a$ , i.e., reward of arm  $a$  when  $n_t(a) = j$

Now one can use HI to derive that

$$\forall a \forall j \Pr(|\bar{v}_j(a) - \mu(a)| \leq r_t(a)) \geq 1 - \frac{2}{T^4}$$

where  $\bar{v}_j(a)$  represents the average reward at arm  $a$  from first  $j$  times arm  $a$  is chosen. Taking a union bound, it follows that (assuming  $K = \#\text{arms} \leq T$ )

$$\Pr(\forall a \forall j \quad |\bar{v}_j(a) - \mu(a)| \leq r_t(a)) \geq 1 - \frac{2}{T^2}.$$

Now, observe that this event is precisely the same as

$$\mathcal{E} := \{\forall a \forall t \quad |\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)\}. \quad (2.7)$$

Therefore, we have proved:

**Lemma 2.5.**  $\Pr[\mathcal{E}] \geq 1 - \frac{2}{T^2}$ , where  $\mathcal{E}$  is given by (2.7).

The event in (2.7) will be called the *clean event*.

Under the assumption of clean event, we can now draw the range of average reward for arm  $a$  at round  $t$  as in Figure 2.2. We introduce the concepts of *upper confidence bound*  $\text{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a)$ , and *lower confidence bound*  $\text{LCB}_t(a) = \bar{\mu}_t(a) - r_t(a)$ . These concepts will be used in several algorithms in this course.

### 2.3.2 Successive Elimination

Let's recap our idea for two arms: alternate them until we find that one arm is much better than the other. Now, using the confidence bounds, we can naturally define the criterion of "one arm is much better". The full algorithm for two arms is as follows:

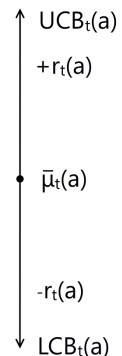


Figure 2.2: The true expected reward  $\mu(a)$  will fall in the interval centered at  $\bar{\mu}_t(a)$  with radius  $r_t(a)$  under clean event

we alternate two arms until  $UCB_t(a) < LCB_t(a')$  (check the condition after the even rounds). When the condition is met, we abandon  $a$ , use  $a'$  forever.

For analysis, assume the clean event. Note that the “disqualified” arm cannot be the best arm. But how much regret do we accumulate *before* disqualifying one arm?

Let  $t$  be the last round when we did *not* invoke the stopping rule, i.e., when the confidence intervals of the two arms still overlap (see Figure 2.3). Then

$$\Delta := |\mu(a) - \mu(a')| \leq 2(r_t(a) + r_t(a')).$$

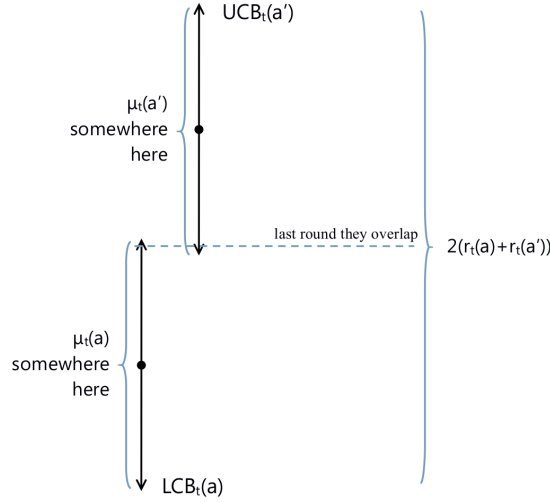


Figure 2.3:  $t$  is the last round that the two confidence intervals still overlap

Since we’ve been alternating the two arms before time  $t$ , we have  $n_t(a) = \frac{t}{2}$  (up to floor and ceiling), which yields

$$\Delta \leq 2(r_t(a) + r_t(a')) \leq 4\sqrt{\frac{2 \log T}{\lfloor t/2 \rfloor}} = O\left(\sqrt{\frac{\log T}{t}}\right).$$

Then total regret accumulated till round  $t$  is

$$R(t) \leq \Delta \times t \leq O\left(t \cdot \sqrt{\frac{\log T}{t}}\right) = O(\sqrt{t \log T}).$$

Since we’ve chosen the best arm from then on, we have  $R(T) \leq O(\sqrt{T \log T})$ .

To complete the proof, we need to argue that the “bad event”  $\bar{\mathcal{E}}$  contributes a negligible amount to regret (much like we did for explore-first):

$$\begin{aligned} \mathbb{E}[R(T)] &= \mathbb{E}[R(T)|\text{clean event}] \times \Pr[\text{clean event}] + \mathbb{E}[R(T)|\text{bad event}] \times \Pr[\text{bad event}] \\ &\leq \mathbb{E}[R(T)|\text{clean event}] + T \times O(T^{-2}) \\ &\leq O(\sqrt{T \log T}). \end{aligned}$$

This completes the analysis for  $K = 2$  arms.

□

This stopping rule can also be generalized to  $K$  arms.

- 1 Initially all arms are set “active”;
- 2 Each phase:
- 3     try all active arms (thus each phase may contain multiple rounds);
- 4     deactivate all arms  $a$  s.t.  $\exists$  arm  $a'$  with  $\text{UCB}_t(a) < \text{LCB}_t(a')$ ;
- 5 Repeat until end of rounds.

**Algorithm 3: Successive Elimination**

To analyze the performance of this algorithm, it suffices to focus on the clean event (2.7) (indeed, as in the case of  $k = 2$  arms, the contribution of the “bad event”  $\bar{\mathcal{E}}$  can be neglected).

Let  $a^*$  be an optimal arm, and consider any arm  $a$  such that  $\mu(a) < \mu(a^*)$ . Look at the last round  $t$  when we did not deactivate arm  $a$  yet (or the last round  $T$  if  $a$  is still active at the end). As in the argument for two arms, the confidence intervals of the two arms  $a$  and  $a^*$  before round  $t$  must overlap. Therefore:

$$\Delta(a) := \mu(a^*) - \mu(a) \leq 2(r_t(a^*) + r_t(a)) = O(r_t(a)).$$

The last equality is because  $n_t(a)$  and  $n_t(a^*)$  differ at most 1, as the algorithm has been alternating active arms. Since arm  $a$  is never played after round  $t$ , we have  $n_t(a) = n_T(a)$ , and therefore  $r_t(a) = r_T(a)$ .

We have proved the following crucial property:

$$\Delta(a) \leq O(r_T(a)) = O\left(\sqrt{\frac{\log T}{n_T(a)}}\right) \quad \text{for each arm } a \text{ with } \mu(a) < \mu(a^*). \quad (2.8)$$

The rest of the analysis will only rely on property (2.8). In other words, it does not matter which algorithm has achieved this property.

The contribution of arm  $a$  to regret, denoted  $R(T; a)$ , can be expressed as  $\Delta(a)$  for each round this arm is played; by (2.8) we can bound this quantity as

$$R(T; a) \leq n_T(a) \cdot O\left(\sqrt{\frac{\log T}{n_T(a)}}\right) = O(\sqrt{n_T(a) \log T}).$$

Let  $\mathcal{A}$  be the set of all  $K$  arms, and let  $\mathcal{A}^+ = \{a : \mu(a) < \mu(a^*)\}$  be the set of all arms that contribute to regret. Summing up over all arms  $a \in \mathcal{A}^+$ , we obtain:

$$R(T) = \sum_{a \in \mathcal{A}^+} R(T; a) = O(\sqrt{\log T}) \sum_{a \in \mathcal{A}^+} \sqrt{n_T(a)} \leq O(\sqrt{\log T}) \sum_{a \in \mathcal{A}} \sqrt{n_T(a)}. \quad (2.9)$$

Note that  $f(x) = \sqrt{x}$  is a real continuous concave function, and  $\sum_{a \in \mathcal{A}} n_T(a) = T$ , by Jensen’s Inequality we have

$$\frac{1}{K} \sum_{a \in \mathcal{A}} \sqrt{n_T(a)} \leq \sqrt{\frac{1}{K} \sum_{a \in \mathcal{A}} n_T(a)} = \sqrt{\frac{T}{K}}$$

Plugging this into (2.9), we see that  $R(T) \leq O(\sqrt{KT \log T})$ . Thus, we have proved:

**Theorem 2.6.** *Successive Elimination algorithm achieves regret  $\mathbb{E}[R(T)] = O(\sqrt{KT \log T})$ , where  $K$  is the number of arms and  $T$  is the time horizon.*

*Remark 2.7.* The  $\sqrt{T}$  dependence on  $T$  in the regret bound for Successive Elimination should be contrasted with the  $T^{2/3}$  dependence for Explore-First. This improvement is possible due to adaptive exploration.

We can also use (2.8) to obtain another regret bound. Rearranging the terms in (2.8), we obtain  $n_T(a) \leq O\left(\frac{\log T}{[\Delta(a)]^2}\right)$ . Therefore, for each arm  $a \in \mathcal{A}^+$  we have:

$$R(T; a) = \Delta(a) \cdot n_T(a) \leq \Delta(a) \cdot O\left(\frac{\log T}{[\Delta(a)]^2}\right) = O\left(\frac{\log T}{\Delta(a)}\right). \quad (2.10)$$

Summing up over all arms  $a \in \mathcal{A}^+$ , we obtain:

$$R(T) \leq O(\log T) \left[ \sum_{a \in \mathcal{A}^+} \frac{1}{\Delta(a)} \right].$$

**Theorem 2.8.** *Successive Elimination algorithm achieves regret*

$$\mathbb{E}[R(T)] \leq O(\log T) \left[ \sum_{\text{arms } a \text{ with } \mu(a) < \mu(a^*)} \frac{1}{\mu(a^*) - \mu(a)} \right]. \quad (2.11)$$

*Remark 2.9.* Regret of Successive Elimination is (at most) logarithmic in  $T$  with an *instance-dependent constant*. The latter constant can be arbitrarily large, depending on a problem instance. The distinction between regret bounds achievable with an absolute constant (as in Theorem 2.6) and regret bounds achievable with an instance-dependent constant is typical for multi-armed bandit problems. The existence of the logarithmic regret bound is another benefit of adaptive exploration compared to non-adaptive exploration.

*Remark 2.10.* It is instructive to derive Theorem 2.6 in a different way: starting from the logarithmic regret bound in (2.10). Informally, we need to get rid of arbitrarily small  $\Delta(a)$ 's in the denominator. Let us fix some  $\epsilon > 0$ , then regret consists of two parts:

- all arms  $a$  with  $\Delta(a) \leq \epsilon$  contribute at most  $\epsilon$  per round, for a total of  $\epsilon T$ ;
- each arms  $a$  with  $\Delta(a) > \epsilon$  contributes at most  $R(T; a) \leq O(\frac{1}{\epsilon} \log T)$  to regret; thus, all such arms contribute at most  $O(\frac{K}{\epsilon} \log T)$ .

Combining these two parts, we see that (assuming the clean event)

$$R(T) \leq O\left(\epsilon T + \frac{K}{\epsilon} \log T\right).$$

Since this holds for  $\forall \epsilon > 0$ , we can choose the  $\epsilon$  that minimizes the right-hand side. Ensuring that  $\epsilon T = \frac{K}{\epsilon} \log T$  yields  $\epsilon = \sqrt{\frac{K}{T} \log T}$ , and therefore  $R(T) \leq O(\sqrt{KT \log T})$ .

### 2.3.3 UCB1 Algorithm

Let us consider another approach for adaptive exploration. The algorithm is based on the *optimism under uncertainty* principle: assume each arm is as good as it can possibly be given the observations so far, and choose the best arm based on these optimistic estimates. This intuition leads to the following simple algorithm called UCB1:

- 1 Try each arm once;
- 2 In each round  $t$ , pick  $\operatorname{argmax}_{a \in \mathcal{A}} \text{UCB}_t(a)$ , where  $\text{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a)$ ;

**Algorithm 4:** UCB1 Algorithm

*Remark 2.11.* Here's more intuition for this algorithm. An arm  $a$  is chosen in round  $t$  because it has a large  $\text{UCB}_t(a)$ . The latter can happen for two reasons: (i)  $\bar{\mu}_t(a)$  is large, implying a high reward; or (ii)  $r_t(a)$  is large, *i.e.*,  $n_t(a)$  is small, implying an under-explored arm. In both cases, this arm is worth choosing. Further, the  $\bar{\mu}_t(a)$  and  $r_t(a)$  summands represent exploitation and exploration, resp., and summing them up is a natural way to implement exploration-exploitation tradeoff.

To analyze this algorithm, let us focus on the clean event (2.7), as before. Let us use the same notation as before:  $a^*$  is a best arm, and  $a_t$  is the arm chosen by the algorithm in round  $t$ . According to the algorithm,  $\text{UCB}_t(a_t) \geq \text{UCB}_t(a^*)$ . Under clean events,  $\mu(a_t) + r_t(a_t) \geq \bar{\mu}_t(a_t)$  and  $\text{UCB}_t(a^*) \geq \mu(a^*)$ . Therefore:

$$\mu(a_t) + 2r_t(a_t) \geq \bar{\mu}_t(a_t) + r_t(a_t) = \text{UCB}_t(a_t) \geq \text{UCB}_t(a^*) \geq \mu(a^*).$$



It follows that

$$\Delta(a_t) := \mu(a^*) - \mu(a_t) \leq 2r_t(a_t) = 2\sqrt{\frac{2 \log T}{n_t(a_t)}}. \quad (2.12)$$

*Remark 2.12.* This is a very cute trick, which resurfaces in the analyses of several UCB-like algorithms for more general bandit settings.

In particular, for each arm  $a$  consider the last round  $t$  when this arm is chosen by the algorithm. Applying (2.12) to this round gives us property (2.8). The rest of the analysis follows from that property, as in the analysis of Successive Elimination.

**Theorem 2.13.** *Algorithm UCB1 achieves regret  $\mathbb{E}[R(T)] \leq O(\sqrt{TK \log T})$  and also the logarithmic regret bound in (2.11).*

## 2.4 Remarks

**Bibliography.** Successive Elimination is from Even-Dar et al. (2002), and UCB1 is from Auer et al. (2002a).

**Regret for all rounds at once.** So far we considered a fixed time horizon  $T$ . What if it is not known in advance, and we wish to achieve good regret bounds for all rounds  $t$ ? In all algorithms we have considered, the knowledge of the time horizon  $T$  is needed (only) in the definition of the confidence radius  $r_t(a)$ . There are several remedies:

- Suppose we wish to achieve same regret bounds for all rounds  $t \leq T$ , for some time horizon  $T$  that is *not* known to the algorithm. If an upper bound  $T' \geq T$  is known, then one can use  $T'$  instead of  $T$  in the definition of the confidence radius. Since  $r_t(a)$  depends on  $T$  only logarithmically, a rather significant over-estimates can be tolerated.
- In UCB1, one can use an “online” version of confidence radius:  $r_t(a) = \sqrt{\frac{2 \log t}{n_t(a)}}$ . In fact, this is the original version of UCB1 from (Auer et al., 2002a). It achieves the same regret bounds, and with better constants, at the cost of a somewhat more complicated analysis.
- Any algorithm for known time horizon can be converted to an algorithm for arbitrary time horizon using the *doubling trick*. Here, the new algorithm proceeds in phases of exponential duration. Each phase  $i = 1, 2, \dots$  lasts  $2^i$  rounds, and executes a fresh run of the original algorithm. This approach achieves the “right” theoretical guarantees (more on this on the homework). However, it is somewhat impractical because the algorithm needs to “forget” everything it has learned after each phase.

**Instantaneous regret.** In this lecture we discussed cumulative regret. Another type of regret is *instantaneous regret* (also called *simple regret*), defined as  $\mathbb{E}[r(t)] = \mu^* - \mathbb{E}[\mu(a_t)]$ . In particular, it may be desirable to spread the regret “more uniformly” over rounds, so as to avoid spikes in instantaneous regret. Then (in addition to a “good” cumulative regret) one would also like an upper bound on instantaneous regret that decreases monotonically over time.

**Bandits with predictions.** The standard goal for bandit algorithms is (as discussed in this lecture) to maximize cumulative reward. An alternative goal is to output a prediction  $a_t^*$  after each round  $t$ . The algorithm is then graded only on the quality of these predictions (so that it does not matter how much reward is accumulated). There are two standard ways to make this objective formal: (i) minimize instantaneous regret  $\mu^* - \mu(a_t^*)$ , and (ii) maximize the probability of choosing the best arm:  $\Pr[a_t^* = a^*]$ . Essentially, good algorithms for cumulative regret, such as Successive Elimination and UCB1, are also good for this version (more on the homework). However, improvements are possible in some regimes, and there are some papers on that.



## Chapter 3

# Lower Bounds

In this lecture (and the first half of the next one), we prove a  $\Omega(\sqrt{KT})$  lower bound for regret of bandit algorithms. This gives us a sense of what are the best possible upper bounds on regret that we can hope to prove.

On a high level, there are two ways of proving a lower bound on regret:

- (1) Give a family  $\mathcal{F}$  of problem instances, which is the same for all algorithms, such that any algorithm ‘fails’ (has high regret) on some instance in  $\mathcal{F}$ .
- (2) Give a distribution over problem instances, and show that, in expectation over this distribution, any algorithm will fail.

Note that (2) implies (1) since: if regret is high in expectation over problem instances, then there exists at least one problem instance with high regret. Also, (1) implies (2) if  $|\mathcal{F}|$  is a constant. This can be seen as follows: suppose we know that for any algorithm we have high regret (say  $H$ ) with one problem instance in  $\mathcal{F}$  and low regret with all other instances in  $\mathcal{F}$ , then, taking a uniform distribution over  $\mathcal{F}$ , we can say that any algorithm has expected regret at least  $H/|\mathcal{F}|$ . (So this argument breaks if  $|\mathcal{F}|$  is large.) If we prove a stronger version of (1) that says that for any algorithm, regret is high for a *constant fraction* of the problem instances in  $\mathcal{F}$ , then, considering a uniform distribution over  $\mathcal{F}$ , this implies (2) regardless of whether  $|\mathcal{F}|$  is large or not.

In this lecture, for proving lower bounds, we consider 0-1 rewards and the following family of problem instances (with fixed  $\epsilon$  to be adjusted in the analysis):

$$\mathcal{I}_j = \begin{cases} \mu_i = 1/2 & \text{for each arm } i \neq j, \\ \mu_i = (1 + \epsilon)/2 & \text{for arm } i = j \end{cases} \quad \text{for each } j = 1, 2, \dots, K. \quad (3.1)$$

(Recall that  $K$  is the number of arms.)

In the previous lecture, we saw that sampling each arm  $\tilde{O}(1/\epsilon^2)$  times is sufficient for the upper bounds on regret that we derived. In this lecture, we prove that sampling each arm  $\Omega(1/\epsilon^2)$  times is necessary to determine whether an arm is bad or not.

The proof methods will require *KL divergence*, an important tool from Information Theory. In the next section, we briefly study the KL divergence and some of its properties.

### 3.1 KL-divergence (brief intro)

Consider a finite sample space  $\Omega$ , and let  $p, q$  be two probability distributions defined on  $\Omega$ . Then, the Kullback-Leibler divergence or *KL-divergence* is defined as:

$$KL(p, q) = \sum_{x \in \Omega} p(x) \ln \frac{p(x)}{q(x)} = \mathbb{E}_p \left[ \ln \frac{p(x)}{q(x)} \right].$$

The KL divergence is similar to a notion of distance with the properties that it is non-negative, 0 iff  $p = q$ , and small if the distributions  $p$  and  $q$  are close. However, it is not strictly a distance function since it is not symmetric and does not satisfy the triangle-inequality.

The intuition for the formula is as follows: we are interested in how certain we are that data, with underlying distribution  $q$ , can be generated from distribution  $p$ . The KL divergence effectively answers this question by measuring the average log likelihood of observing data with distribution  $p$  when the underlying distribution of the data is actually given by  $q$ .

*Remark 3.1.* The definition of KL-divergence, as well as the properties discussed below, extend to infinite sample spaces. However, KL-divergence for finite sample spaces suffices for this class, and is much easier to work with.

### 3.1.1 Properties of KL-divergence

We present several basic properties of KL-divergence that will be needed later. The proofs of these properties are fairly simple, we include them here for the sake of completeness.

1. *Gibbs' Inequality:*  $KL(p, q) \geq 0, \forall p, q$ . Further,  $KL(p, q) = 0$  iff  $p = q$ .

*Proof.* Let us define:  $f(y) = y \ln(y)$ .  $f$  is a convex function under the domain  $y > 0$ . Now, from the definition of the KL divergence we get:

$$\begin{aligned}
 KL(p, q) &= \sum_{x \in \Omega} p(x) \ln \frac{p(x)}{q(x)} \\
 &= \sum_{x \in \Omega} q(x) f\left(\frac{p(x)}{q(x)}\right) \\
 &\geq f\left(\sum_{x \in \Omega} q(x) \frac{p(x)}{q(x)}\right) && \text{[follows from Jensen's inequality]} \\
 &= f\left(\sum_{x \in \Omega} p(x)\right) = f(1) = 0,
 \end{aligned}$$

where Jensen's inequality states that  $\varphi(\lambda_1 x_1 + \lambda_2 x_2) \leq \lambda_1 \varphi(x_1) + \lambda_2 \varphi(x_2)$ , if  $\varphi$  is a convex function and  $\lambda_1 + \lambda_2 = 1$  with  $\lambda_1, \lambda_2 > 0$ . Jensen's inequality further has the property that  $\varphi(\lambda_1 x_1 + \lambda_2 x_2) = \lambda_1 \varphi(x_1) + \lambda_2 \varphi(x_2)$  iff  $x_1 = x_2$  or if  $\varphi$  is a linear function. In this case, since  $f$  is not a linear function, equality holds (i.e.,  $KL(p, q) = 0$  iff  $p(x) = q(x), \forall x$ ).  $\square$

2. Let the sample space  $\Omega$  be composed as  $\Omega = \Omega_1 \times \Omega_1 \times \dots \times \Omega_n$ . Further, let  $p$  and  $q$  be two distributions defined on  $\Omega$  as  $p = p_1 \times p_2 \times \dots \times p_n$  and  $q = q_1 \times q_2 \times \dots \times q_n$ , such that  $\forall j = 1, \dots, n, p_j$  and  $q_j$  are distributions defined on  $\Omega_j$ . Then we have the property:  $KL(p, q) = \sum_{j=1}^n KL(p_j, q_j)$ .

*Proof.* Let  $x = (x_1, x_2, \dots, x_n) \in \Omega$  st  $x_i \in \Omega_i, \forall i = 1, \dots, n$ . Let  $h_i(x_i) = \ln \frac{p_i(x_i)}{q_i(x_i)}$ . Then:

$$\begin{aligned}
KL(p, q) &= \sum_{x \in \Omega} p(x) \ln \frac{p(x)}{q(x)} \\
&= \sum_{i=1}^n \sum_{x \in \Omega} p(x) h_i(x_i) && \left[ \text{since } \ln \frac{p(x)}{q(x)} = \sum_{i=1}^n h_i(x_i) \right] \\
&= \sum_{i=1}^n \sum_{x_i^* \in \Omega_i} h_i(x_i^*) \sum_{\substack{x \in \Omega, \\ x_i = x_i^*}} p(x) \\
&= \sum_{i=1}^n \sum_{x_i \in \Omega_i} p_i(x_i) h_i(x_i) && \left[ \text{since } \sum_{x \in \Omega, x_i = x_i^*} p(x) = p_i(x_i^*) \right] \\
&= \sum_{i=1}^n KL(p_i, q_i). \quad \square
\end{aligned}$$

3. *Weaker form of Pinsker's inequality:*  $\forall A \subset \Omega : 2(p(A) - q(A))^2 \leq KL(p, q)$ .

*Proof.* To prove this property, we first claim the following:

**Claim 3.2.** For each event  $A \subset \Omega$ ,

$$\sum_{x \in A} p(x) \ln \frac{p(x)}{q(x)} \geq p(A) \ln \frac{p(A)}{q(A)}.$$

*Proof.* Let us define the following:

$$p_A(x) = \frac{p(x)}{p(A)} \quad \text{and} \quad q_A(x) = \frac{q(x)}{q(A)} \quad \forall x \in A.$$

Then the claim can be proved as follows:

$$\begin{aligned}
\sum_{x \in A} p(x) \ln \frac{p(x)}{q(x)} &= p(A) \sum_{x \in A} p_A(x) \ln \frac{p(A)p_A(x)}{q(A)q_A(x)} \\
&= p(A) \left( \sum_{x \in A} p_A(x) \ln \frac{p_A(x)}{q_A(x)} \right) + p(A) \ln \frac{p(A)}{q(A)} \sum_{x \in A} p_A(x) \\
&\geq p(A) \ln \frac{p(A)}{q(A)}. && \left[ \text{since } \sum_{x \in A} p_A(x) \ln \frac{p_A(x)}{q_A(x)} = KL(p_A, q_A) \geq 0 \right]
\end{aligned}$$

□

Fix  $A \subset \Omega$ . Using Claim 3.2 we have the following:

$$\begin{aligned}
\sum_{x \in A} p(x) \ln \frac{p(x)}{q(x)} &\geq p(A) \ln \frac{p(A)}{q(A)}, \\
\sum_{x \notin A} p(x) \ln \frac{p(x)}{q(x)} &\geq p(\bar{A}) \ln \frac{p(\bar{A})}{q(\bar{A})},
\end{aligned}$$

where  $\bar{A}$  denotes the complement of  $A$ . Now, let  $a = p(A)$  and  $b = q(A)$ . Further, assume  $a < b$ . Then, we have:

$$\begin{aligned}
KL(p, q) &= a \ln \frac{a}{b} + (1 - a) \ln \frac{1 - a}{1 - b} \\
&= \int_a^b \left( -\frac{a}{x} + \frac{1 - a}{1 - x} \right) dx \\
&= \int_a^b \frac{x - a}{x(1 - x)} dx \\
&\geq \int_a^b 4(x - a) dx = 2(b - a)^2. \quad [\text{since } x(1 - x) \leq 1/4]
\end{aligned}$$

This proves the property. □

4. Let  $p_\epsilon$  denote a distribution on  $\{0, 1\}$  such that  $p_\epsilon(1) = (1 + \epsilon)/2$ . Thus,  $p_\epsilon(0) = (1 - \epsilon)/2$ . Further, let  $p_0$  denote the distribution on  $\{0, 1\}$  where  $p_0(0) = p_0(1) = 1/2$ . Then we have the property:  $KL(p_\epsilon, p_0) \leq 2\epsilon^2$ .

*Proof.*

$$\begin{aligned}
KL(p_\epsilon, p_0) &= \frac{1 + \epsilon}{2} \ln(1 + \epsilon) + \frac{1 - \epsilon}{2} \ln(1 - \epsilon) \\
&= \frac{1}{2} (\ln(1 + \epsilon) + \ln(1 - \epsilon)) + \frac{\epsilon}{2} (\ln(1 + \epsilon) - \ln(1 - \epsilon)) \\
&= \frac{1}{2} \ln(1 - \epsilon^2) + \frac{\epsilon}{2} \ln \frac{1 + \epsilon}{1 - \epsilon}.
\end{aligned}$$

Now,  $\ln(1 - \epsilon^2) < 0$  and we can write  $\ln \frac{1 + \epsilon}{1 - \epsilon} = \ln \left( 1 + \frac{2\epsilon}{1 - \epsilon} \right) \leq \frac{2\epsilon}{1 - \epsilon}$ . Thus, we get:

$$KL(p_\epsilon, p_0) < \frac{\epsilon}{2} \cdot \frac{2\epsilon}{1 - \epsilon} = \frac{\epsilon^2}{1 - \epsilon} \leq 2\epsilon^2. \quad \square$$

### 3.1.2 How are these properties going to be used?

We start with the same setting as in Property 2. From Property 3, we have:

$$\begin{aligned}
2(p(A) - q(A))^2 &\leq KL(p, q) \\
&= \sum_{j=1}^n KL(p_j, q_j). \quad (\text{follows from Property 2})
\end{aligned}$$

For example, we can define  $p_j$  and  $q_j$  to be distributions of a biased coin with small  $\epsilon$  ( $p_j(1) = (1 + \epsilon)/2$ ,  $p_j(0) = (1 - \epsilon)/2$ ) vs an unbiased coin ( $q_j(0) = q_j(1) = 1/2$ ). Then, we can use Property 4 to bound the above as:

$$2(p(A) - q(A))^2 \leq \sum_{j=1}^n KL(p_j, q_j) \leq \sum_{j=1}^n \delta = n\delta,$$

where  $\delta = 2\epsilon^2$ . Thus, we arrive at the following bound:

$$|p(A) - q(A)| \leq \sqrt{n\delta/2}.$$

### 3.2 A simple example: flipping one coin

We start with a simple example, which illustrates our proof technique *and* is interesting as a standalone result. We have a single coin, whose outcome is a 0 or 1. The coin's mean is unknown. We assume that the true mean  $\mu \in [0, 1]$  is either  $\mu_1$  or  $\mu_2$  for two known values  $\mu_1 > \mu_2$ . The coin is flipped  $T$  times. The goal is to identify if  $\mu = \mu_1$  or  $\mu = \mu_2$ .

Define  $\Omega := \{0, 1\}^T$  to be the sample space of the outcomes of the  $T$  coin tosses. We need a decision rule  $Rule : \Omega \rightarrow \{High, Low\}$  with the following two properties:

$$\Pr[Rule(observations) = High | \mu = \mu_1] \geq 0.99 \quad (3.2)$$

$$\Pr[Rule(observations) = Low | \mu = \mu_2] \geq 0.99 \quad (3.3)$$

The question is how large should  $T$  be for for such a  $Rule$  to exist? We know that if  $\delta = |\mu_1 - \mu_2|$ , then  $T \geq \Omega(\frac{1}{\delta^2})$  is sufficient. We will prove that it is also necessary. We will focus on the special case when both  $\mu_1$  and  $\mu_2$  are close to  $\frac{1}{2}$ .

**Claim 3.3.** *Let  $\mu_1 = \frac{1+\epsilon}{2}$  and  $\mu_2 = \frac{1}{2}$ . For any rule to “work” (i.e., satisfy equations (3.2) and (3.3)) we need*

$$T \geq \Omega(\frac{1}{\epsilon^2}) \quad (3.4)$$

*Proof.* Define for any event  $A \subseteq \Omega$ , the following quantities

$$P_1(A) = \Pr[A | \mu = \mu_1]$$

$$P_2(A) = \Pr[A | \mu = \mu_2]$$

To prove this claim, we will consider the following equation. For an event  $A \subseteq \Omega$ , such that  $Rule(A) = High$  (i.e.  $A = \{\omega \subseteq \Omega : Rule(\omega) = High\}$ ),

$$P_1(A) - P_2(A) \geq 0.98 \quad (3.5)$$

We prove the claim by showing that if (3.4) is false, then (3.5) is false, too. Specifically, we will assume that  $T < \frac{1}{4\epsilon^2}$ . (In fact, the argument below holds for an arbitrary event  $A \subset \Omega$ .)

Define, for all  $i \in \{1, 2\}$ ,  $P_{i,t}$  to be the distribution of the  $t^{th}$  coin toss for  $P_i$ . Then,  $P_i = P_{i,1} \times P_{i,2} \times \dots \times P_{i,T}$ . From KL divergence property, we have

$$\begin{aligned} 2(P_1(A) - P_2(A))^2 &\leq KL(P_1, P_2) && \text{From KL divergence property 3} \\ &= \sum_{t=1}^T KL(P_{1,t}, P_{2,t}) && \text{From KL divergence property 2} \\ &\leq 2T\epsilon^2 && \text{From KL divergence property 4} \end{aligned}$$

Hence, we have  $|P_1(A) - P_2(A)| \leq \sqrt{2T} \cdot \epsilon < \frac{1}{2}$ . □

### 3.3 Flipping several coins: “bandits with prediction”

Let us extend the previous example to flipping multiple coins. More formally, we consider a bandit problem with  $K$  arms (where each arm corresponds to a coin). Each arm gives a 0-1 reward, drawn independently from a fixed but unknown distribution. After  $T$  rounds, the algorithm outputs a guess  $y_T \in \mathcal{A}$  for which arm is the best arm (where  $\mathcal{A}$  is the set of all arms).<sup>1</sup> We call this version “bandits with predictions”. In this section, we will only be concerned with a quality of prediction, rather than accumulated rewards and regret.

For each arm  $a \in \mathcal{A}$ , the mean reward is denoted as  $\mu(a)$ . (We will also write it as  $\mu_a$  whenever convenient.) A particular problem instance is specified as a tuple  $\mathcal{I} = (\mu(a) : \forall a \in \mathcal{A})$ .

<sup>1</sup>Recall that the “best arm” is the arm with the highest mean reward.

A good algorithm for the “bandits with prediction” problem described above should satisfy

$$\Pr[y_T \text{ is correct} \mid \mathcal{I}] \geq 0.99 \quad (3.6)$$

for each problem instance  $\mathcal{I}$ . We will use the family (3.1) of problem instances to argue that one needs  $T \geq \Omega\left(\frac{K}{\epsilon^2}\right)$  for any algorithm to “work”, i.e., satisfy (3.6), on all instances in this family.

**Lemma 3.4.** *Suppose an algorithm for “bandits with predictions” satisfies (3.6) for all problem instances  $\mathcal{I}_1, \dots, \mathcal{I}_K$ . Then  $T \geq \Omega\left(\frac{K}{\epsilon^2}\right)$ .*

This result is of independent interest (regardless of the lower bound on regret). In fact, we will prove a stronger lemma which will (also) be the crux in the proof of the regret bound.

**Lemma 3.5.** *Suppose  $T \leq \frac{cK}{\epsilon^2}$ , for a small enough absolute constant  $c$ . Fix any deterministic algorithm for “bandits with prediction”. Then there exists at least  $\lceil K/3 \rceil$  arms  $j$  such that*

$$\Pr[y_T = j \mid \mathcal{I}_j] < \frac{3}{4}$$

*Remark 3.6.* The proof for  $K = 2$  arms is particularly simple, so we will do it first. We will then extend this proof to arbitrary  $K$  with more subtleties. While the lemma holds for an arbitrary  $K$ , we will present a simplified proof which requires  $K \geq 24$ .

We will use a standard shorthand  $[T] := \{1, 2, \dots, T\}$ .

Let us set up the sample space to be used in the proof. Let  $(r_t(a) : a \in \mathcal{A}, t \in [T])$  be mutually independent 0-1 random variables such that  $r_t(a)$  has expectation  $\mu(a)$ . We refer to this tuple as the *rewards table*, where we interpret  $r_t(a)$  as the reward received by the algorithm for the  $t$ -th time it chooses arm  $a$ . The sample space is  $\Omega = \{0, 1\}^{K \times T}$ , where each outcome  $\omega \in \Omega$  corresponds to a particular realization of the rewards table. Each problem instance  $\mathcal{I}_j$  defines distribution  $P_j$  on  $\Omega$ :

$$P_j(A) = \Pr[A \mid \mathcal{I}_j] \quad \text{for each } A \subset \Omega.$$

Also, let  $P_j^{a,t}$  be the distribution of  $r_t(a)$  under instance  $\mathcal{I}_j$ , so that  $P_j = \prod_{a \in \mathcal{A}, t \in [T]} P_j^{a,t}$ .

*Proof ( $K = 2$  arms).* Define  $A = \{\omega \subseteq \Omega : y_T = 1\}$ . In other words,  $A$  is the set of all events such that the “correct” arm is arm 1. (But the argument below holds for any any event  $A \subset \Omega$ .)

Similar to the previous section, we use the properties of KL divergence as follows:

$$\begin{aligned} 2(P_1(A) - P_2(A))^2 &\leq KL(P_1, P_2) \\ &= \sum_{a=1}^{K=2} \sum_{t=1}^T KL(P_1^{a,t}, P_2^{a,t}) \\ &\leq 2T \cdot 2\epsilon^2 \end{aligned} \quad (3.7)$$

The last inequality is because KL divergence of  $P_1^{a,t}$  and  $P_2^{a,t}$  is non-zero if and only if  $P_1^{a,t} \neq P_2^{a,t}$ . And when they are non-equal, their KL divergence is  $2\epsilon^2$ . Hence,

$$|P_1(A) - P_2(A)| \leq 2\epsilon\sqrt{T} < \frac{1}{2}.$$

The last inequality holds whenever  $T \leq \left(\frac{1}{4\epsilon}\right)^2$ .

To complete the proof, observe that if  $\Pr[y_T = j \mid \mathcal{I}_j] \geq \frac{3}{4}$  for both problem instances, then  $P_1(A) \geq \frac{3}{4}$  and  $P_2(A) < \frac{1}{4}$ , so their difference is at least  $\frac{1}{2}$ , contradiction.  $\square$

*Proof ( $K \geq 24$ ).* Compared to the 2-arms case, time horizon  $T$  can be larger by a factor of  $O(K)$ . The crucial improvement is a more delicate version of the KL-divergence argument in (3.7) which results in the right-hand side of the form  $O(T\epsilon^2/K)$ .

For the sake of the analysis, we will consider an additional problem instance

$$\mathcal{I}_0 = \left\{ \mu_i = \frac{1}{2} \quad : \text{for all arms } a \right\}$$



which we call the “base instance”. Let  $\mathbb{E}_0[\cdot]$  be the expectation given this problem instance. Also, let  $T_a$  be the total number of times arm  $a$  is played.

We consider the algorithm’s performance on problem instance  $\mathcal{I}_0$ , and focus on arms  $j$  that are “neglected” by the algorithm, in the sense that the algorithm does not choose arm  $j$  very often *and* is not likely to pick  $j$  for the guess  $y_T$ . Formally, we observe that

$$\exists \geq \frac{2K}{3} \text{ arms } j \text{ such that } \mathbb{E}_0(T_j) \leq \frac{3T}{K} \quad (3.8)$$

$$\exists \geq \frac{2K}{3} \text{ arms } j \text{ such that } P_0(y_T = j) \leq \frac{3}{K}. \quad (3.9)$$

(To prove (3.8), assume for contradiction that we have more than  $\frac{K}{3}$  arms with  $\mathbb{E}_0(T_j) > \frac{3T}{K}$ . Then the expected total number of times these arms are played is strictly greater than  $T$ , which is a contradiction. (3.9) is proved similarly.) By Markov inequality,

$$\mathbb{E}_0(T_j) \leq \frac{3T}{K} \text{ implies that } \Pr[T_j \leq \frac{24T}{K}] \geq \frac{7}{8}.$$

Since the sets of arms in (3.8) and (3.9) must overlap on least  $\frac{K}{3}$  arms, we conclude:

$$\exists \geq \frac{K}{3} \text{ arms } j \text{ such that } \Pr[T_j \leq m] \geq \frac{7}{8} \text{ and } P_0(y_T = j) \leq \frac{3}{K}, \quad (3.10)$$

where  $m = \frac{24T}{K}$ .

We will now refine our definition of the sample space to get the required claim. For each arm  $a$ , define the  $t$ -round sample space  $\Omega_a^t = \{0, 1\}^t$ , where each outcome corresponds to a particular realization of the tuple  $(r_s(a) : s \in [t])$ . (Recall that we interpret  $r_t(a)$  as the reward received by the algorithm for the  $t$ -th time it chooses arm  $a$ .) Then the “full” sample space we considered before can be expressed as  $\Omega = \prod_{a \in \mathcal{A}} \Omega_a^T$ .

Fix an arm  $j$  satisfying the two properties in (3.10). We will consider a “reduced” sample space in which arm  $j$  is played only  $m = \frac{24T}{K}$  times:

$$\Omega^* = \Omega_j^m \times \prod_{\text{arms } a \neq j} \Omega_a^T. \quad (3.11)$$

Each problem instance  $\mathcal{I}_\ell$  defines a distribution  $P_\ell^*$  on  $\Omega^*$ :

$$P_\ell^*(A) = \Pr[A|\mathcal{I}_\ell] \text{ for each } A \subset \Omega^*.$$

In other words, distribution  $P_\ell^*$  is a restriction of  $P_\ell$  to the reduced sample space  $\Omega^*$ .

We apply the KL-divergence argument to distributions  $P_0^*$  and  $P_j^*$ . For each event  $A \subset \Omega^*$ :

$$\begin{aligned} 2(P_0^*(A) - P_j^*(A))^2 &\leq KL(P_0^*, P_j^*) \\ &= \sum_{\text{arm } a \neq j} \sum_{t=1}^T KL(P_0^{a,t}, P_j^{a,t}) + \sum_{t=1}^m KL(P_0^{j,t}, P_j^{j,t}) \\ &\leq 0 + m * 2\epsilon^2. \end{aligned}$$

Note that each arm  $a \neq j$  has identical distributions under instances  $\mathcal{I}_0$  and  $\mathcal{I}_j$  (namely, its mean reward is  $\frac{1}{2}$ ). So distributions  $P_0^{a,t}$  and  $P_j^{a,t}$  are the same, and therefore their KL-divergence is 0. Whereas for arm  $j$  we only need to sum up over  $m$  samples.

Therefore, assuming  $T \leq \frac{cK}{\epsilon^2}$  with small enough constant  $c$ , we can conclude that

$$|P_0^*(A) - P_j^*(A)| \leq \epsilon\sqrt{m} < \frac{1}{8} \text{ for all events } A \subset \Omega^*. \quad (3.12)$$

To apply (3.12), we need to make sure that the event  $A$  is in fact contained in  $\Omega^*$ , *i.e.*, whether  $A$  holds is completely determined by the first  $m$  samples of arm  $j$  (and arbitrarily many samples of other arms). In particular, we cannot take

$A = \{y_t = j\}$ , which would be the most natural extension of the proof technique from the 2-arms case. Instead, we apply (3.12) twice: to events

$$A = \{y_T = j \text{ and } T_j \leq m\} \text{ and } A' = \{T_j > m\}. \quad (3.13)$$

Indeed, note that whether the algorithm samples arm  $j$  more than  $m$  times is completely determined by the first  $m$  coin tosses!

We are ready for the final computation:

$$\begin{aligned} P_j(A) &\leq \frac{1}{8} + P_0(A) && \text{by (3.12)} \\ &\leq \frac{1}{8} + P_0(y_T = j) \\ &\leq \frac{1}{4} && \text{by our choice of arm } j. \\ P_j(A') &\leq \frac{1}{8} + P_0(A') && \text{by (3.12)} \\ &\leq \frac{1}{4} && \text{by our choice of arm } j. \\ P_j(Y_T = j) &\leq P_j^*(Y_T = j \text{ and } T_j \leq m) + P_j^*(T_j > m) \\ &= P_j(A) + P_j(A') \leq \frac{1}{4}. \end{aligned}$$

Recall that this holds for any arm  $j$  satisfying the properties in (3.10). Since there are at least  $K/3$  such arms, the lemma follows.  $\square$

### 3.4 From “bandits with predictions” to regret

**Notation.**  $\Pr[\cdot \mid \mathcal{I}]$  and  $\mathcal{E}[\cdot \mid \mathcal{I}]$  denote probability and expectation, resp., for problem instance  $\mathcal{I}$ . As usual,  $\mu(a)$  denotes the mean reward of arm  $a$ , and  $\mu^* = \max_a \mu(a)$  is the best mean reward.

We use Lemma 3.5 to finish our proof of the  $\sqrt{KT}$  lower bound on regret.

**Theorem 3.7** (Auer et al. (2002b)). *Fix time horizon  $T$  and the number of arms  $K$ . For any bandit algorithm, there exists a problem instance such that  $\mathbb{E}[R(T)] \geq \Omega(\sqrt{KT})$ . In fact, this lower bound holds for a uniform distribution over the instances  $\mathcal{I}_j$ : for any algorithm we have*

$$\mathbb{E}_{j \sim \text{uniform}} [\mathbb{E}[R(T) \mid \mathcal{I}_j]] \geq \Omega(\sqrt{KT}). \quad (3.14)$$

*Proof.* Fix  $\epsilon$  in (3.1) (to be adjusted later), and assume that  $T \leq \frac{cK}{\epsilon^2}$ , where  $c$  is the constant from Lemma 3.5. Fix a bandit algorithm. Let us interpret it as a “bandits with predictions” algorithm by saying that the prediction in each round  $t$  is simply  $a_t$ , the arm chosen in this round.

Fix round  $t$ . Denote

$$S_t = \{\text{arms } j : \Pr[a_t = j \mid \mathcal{I}_j] \leq \frac{3}{4}\},$$

the set of problem instances  $\mathcal{I}_j$  with small probability of making a correct prediction in round  $t$ . Note that we can apply Lemma 3.5 to round  $t$  (then round  $t$  becomes the time horizon in Lemma 3.5), to deduce that  $|S_t| \geq \frac{K}{3}$ .

Let us consider a uniform distribution over problem instances  $\mathcal{I}_j$ . In what follows, the expectations are over the choice of  $\mathcal{I}_j$  (as well as the random rewards and the randomness in the algorithm). Since the problem instances  $\mathcal{I}_j$  have mean reward  $\mu^* = \frac{1+\epsilon}{2}$  on the best arm  $a^*$ , and mean reward  $\frac{1}{2}$  on all other arms, we have:

$$\begin{aligned} \mathbb{E}[\mu(a_t) \mid a^* \in S_t] &\leq \frac{1}{2} \Pr[a_t \neq a^* \mid a^* \in S_t] + \frac{1+\epsilon}{2} \Pr[a_t = a^* \mid a^* \in S_t] \\ &= \frac{1}{2} + \frac{\epsilon}{2} p[a_t = a^* \mid a^* \in S_t] \\ &= \frac{1}{2} + \frac{\epsilon}{2} \frac{3}{4} \\ &= \mu^* - \frac{\epsilon}{8}. \\ \mathbb{E}[\mu(a_t)] &= \Pr[a^* \in S_t] \mathbb{E}[\mu(a_t) \mid a^* \in S_t] + \Pr[a^* \notin S_t] \mathbb{E}[\mu(a_t) \mid a^* \notin S_t] \\ &\leq \mu^* - p[a^* \in S_t] \frac{\epsilon}{8} \\ &\leq \mu^* - \epsilon/24. \end{aligned}$$

Now summing up over all rounds, we obtain  $\mathbb{E}[R(T)] \geq \epsilon T/24$ . Using  $\epsilon = \sqrt{\frac{cK}{T}}$ , we obtain (3.14).  $\square$

### 3.5 Per-instance lower bound (no proofs)

In addition to Theorem 3.7, there is another fundamental lower bound on regret. This lower bound applies to any given problem instance and talks about  $\log(T)$  regret with an instance-dependent constant, complementing the  $\log(T)$  *upper* bound that we proved for algorithms UCB1 and Successive Elimination. We will present and discuss this lower bound without giving a proof.<sup>2</sup>

As before, we focus on 0-1 rewards. For a particular problem instance, we view  $\mathbb{E}[R(t)]$  a function of  $t$ , and we are interested in how this function grows with  $t$ . We start with a simpler and slightly weaker version of the lower bound:

**Theorem 3.8.** *No algorithm can achieve regret  $\mathbb{E}[R(t)] = o(c_{\mathcal{I}} \log t)$  for all problem instances  $\mathcal{I}$ , where the “constant”  $c_{\mathcal{I}}$  can depend on the problem instance but not on the time  $t$ .*

A stronger version is stated as follows:

**Theorem 3.9.** *Fix  $K$ , the number of arms. Consider an algorithm such that*

$$\mathbb{E}[R(t)] \leq O(C_{\mathcal{I},\alpha} t^\alpha) \quad \text{for each problem instance } \mathcal{I} \text{ and each } \alpha > 0. \quad (3.15)$$

Here the “constant”  $C_{\mathcal{I},\alpha}$  can depend on the problem instance  $\mathcal{I}$  and the  $\alpha$ , but not on time  $t$ .

Fix an arbitrary problem instance  $\mathcal{I}$ . For this problem instance:

$$\text{There exists time } t_0 \text{ such that for any } t \geq t_0 \quad \mathbb{E}[R(t)] \geq C_{\mathcal{I}} \ln(t), \quad (3.16)$$

for some constant  $C_{\mathcal{I}}$  that depends on the problem instance, but not on time  $t$ .

*Remark 3.10.* The assumption (3.15) is necessary to rule out trivial counterexamples: e.g., an algorithm which always play arm 1 will have zero regret on a problem instance for which arm 1 is the best arm. Thus, we cannot hope to prove (3.16) (or any non-trivial lower bound on regret that applies to every problem instance) without ruling out such trivial algorithms.

Let us specify how the instance-dependent constant  $C_{\mathcal{I}}$  in (3.16) can be chosen. Let  $\Delta(a) = \mu^* - \mu(a)$  be the “badness” of arm  $a$ .

**Theorem 3.11.** *For each problem instance  $\mathcal{I}$  and any algorithm that satisfies (3.15),*

(a) *the bound (3.16) holds with*

$$C_{\mathcal{I}} = \sum_{a: \Delta(a) > 0} \frac{\mu^*(1 - \mu^*)}{\Delta(a)}.$$

(b) *for each  $\epsilon > 0$ , the bound (3.16) holds with*

$$C_{\mathcal{I}} = \sum_{a: \Delta(a) > 0} \frac{\Delta(a)}{KL(\mu(a), \mu^*)} - \epsilon.$$

*Remark 3.12.* The lower bound from part (a) is similar to the upper bound achieved by UCB1 and Successive Elimination:  $R(T) \leq \sum_{a: \Delta(a) > 0} \frac{O(\log T)}{\Delta(a)}$ . In particular, we see that the upper bound is optimal up to a constant factor when  $\mu^*$  is bounded away from 0 and 1 (e.g., when  $\mu^* \in [\frac{1}{4}, \frac{3}{4}]$ ).

*Remark 3.13.* Part (b) is a stronger (i.e., larger) lower bound which implies the more familiar form in part (a). Several algorithms in the literature are known to come arbitrarily close to this lower bound. In particular, a version of Thompson Sampling (discussed soon) achieves regret

$$R(t) \leq (1 + \delta) C_{\mathcal{I}} \ln(t) + C'_{\mathcal{I}}/\epsilon^2, \quad \forall \delta > 0,$$

where  $C_{\mathcal{I}}$  is from part (b) and  $C'_{\mathcal{I}}$  is some other instance-dependent constant.

<sup>2</sup>The proof is also based on a KL-divergence technique. It can be found in the original paper (Lai and Robbins, 1985), as well as in the survey (Bubeck and Cesa-Bianchi, 2012).

### 3.6 Bibliographic notes

The  $\Omega(\sqrt{KT})$  lower bound on regret is from Auer et al. (2002b). KL-divergence and its properties is “textbook material” from Information Theory, *e.g.*, see Cover and Thomas (1991). The present exposition — the outline and much of the technical details — is based on Robert Kleinberg’s lecture notes from (Kleinberg, 2007, week 9).

We present a substantially simpler proof compared to (Auer et al., 2002b) and (Kleinberg, 2007, week 9) in that we avoid the general “chain rule” for KL-divergence. Instead, we only use the special case of independent distributions (Property 2 in Section 3.1), which is much easier to state and to apply. The proof of Lemma 3.5 (for general  $K$ ), which in prior work relies on the general “chain rule”, is modified accordingly. In particular, we define the “reduced” sample space  $\Omega^*$  with only a small number of samples from the “bad” arm  $j$ , and apply the KL-divergence argument to carefully defined events in (3.13), rather than a seemingly more natural event  $A = \{y_T = j\}$ .

## Interlude A:

# Bandits with Initial Information

Sometimes some information about the problem instance is known to the algorithm beforehand; informally, we refer to it as “initial information”. When and if such information is available, one would like to use it to improve algorithm’s performance. Using the “initial information” has been a subject of much recent work on bandits.

However, how does the “initial information” look like, what is a good theoretical way to model it? Several approaches has been suggested in the literature.

**Constrained reward functions.** Here the “initial information” is that the reward function<sup>3</sup> must belong to some family  $\mathcal{F}$  of feasible reward functions with nice properties. Several examples are below:

- $\mathcal{F}$  is a product set:  $\mathcal{F} = \prod_{\text{arms } a} I_a$ , where  $I_a \subset [0, 1]$  is the interval of possible values for  $\mu(a)$ , the mean reward of arm  $a$ . Then each  $\mu(a)$  can take an arbitrary value in this interval, regardless of the other arms.
- one good arm, all other arms are bad: *e.g.*, , the family of instances  $\mathcal{I}_j$  from the lower bound proof.
- “embedded” reward functions: each arm corresponds to a point in  $\mathbb{R}^d$ , so that the set of arms  $\mathcal{A}$  is interpreted as a subset of  $\mathbb{R}^d$ , and the reward function maps real-valued vectors to real numbers. Further, some assumption is made on these functions. Some of the typical assumptions are:  $\mathcal{F}$  is all linear functions,  $\mathcal{F}$  is all concave functions, and  $\mathcal{F}$  is a Lipschitz function. Each of these assumptions gave rise to a fairly long line of work.

From a theoretical point of view, we simply assume that  $\mu \in \mathcal{F}$  for the appropriate family  $\mathcal{F}$  of problem instances. Typically such assumption introduces dependence between arms, and one can use this dependence to infer something about the mean reward of one arm by observing the rewards of some other arms. In particular, Lipschitz assumption allows only “short-range” inferences: one can learn something about arm  $a$  only by observing other arms that are not too far from  $a$ . Whereas linear and concave assumptions allow “long-range” inferences: it is possible to learn something about arm  $a$  by observing arms that lie very far from  $a$ .

When one analyzes an algorithm under this approach, one usually proves a regret bound for each  $\mu \in \mathcal{F}$ . In other words, the regret bound is only as good as the *worst case* over  $\mathcal{F}$ . The main drawback is that such regret bound may be overly pessimistic: what if the “bad” problem instances in  $\mathcal{F}$  occur very rarely in practice? In particular, what if most of instances in  $\mathcal{F}$  share some nice property such as linearity, whereas a few bad-and-rare instances do not.

**Bayesian bandits.** Another major approach is to represent the “initial information” as a distribution  $\mathbb{P}$  over the problem instances, and assume that the problem instance is drawn independently from  $\mathbb{P}$ . This distribution is called “prior distribution”, or “Bayesian prior”, or simply a “prior”. One is typically interested in *Bayesian regret*: regret in expectation over the prior. This approach a special case of *Bayesian models* which are very common in statistics and machine learning: an instance of the model is sampled from a prior distribution which (typically) is assumed to be known, and one is interested in performance in expectation over the prior.

A prior  $\mathbb{P}$  also defines the family  $\mathcal{F}$  of feasible reward functions: simply,  $\mathcal{F}$  is the support of  $\mathbb{P}$ . Thus, the prior can specify the family  $\mathcal{F}$  from the “constrained rewards functions” approach. However, compared to that approach, the prior can also specify that some reward functions in  $\mathcal{F}$  are more likely than others.

---

<sup>3</sup>Recall that the *reward function*  $\mu$  maps arms to its mean rewards. We can also view  $\mu$  as a vector  $\mu \in [0, 1]^K$ .

An important special case is *independent priors*: mean reward  $(\mu(a) : a \in \mathcal{A})$  are mutually independent. Then the prior  $\mathbb{P}$  can be represented as a product  $\mathbb{P} = \prod_{\text{arms } a} \mathbb{P}_a$ , where  $\mathbb{P}_a$  is the prior for arm  $a$  (meaning that the mean reward  $\mu(a)$  is drawn from  $\mathbb{P}_a$ ). Likewise, the support  $\mathcal{F}$  is a product set  $\mathcal{F} = \prod_{\text{arms } a} \mathcal{F}_a$ , where each  $\mathcal{F}_a$  is the set of all possible values for  $\mu(a)$ . Per-arm priors  $\mathbb{P}_a$  typically considered in the literature include a uniform distribution over a given interval, a Gaussian (truncated or not), and just a discrete distribution over several possible values.

Another typical case is when the support  $\mathcal{F}$  is a highly constrained family such as the set of all linear functions, so that the arms are very dependent on one another.

The prior can substantially restrict the set of feasible functions that we are likely to see even if it has “full support” (*i.e.*, if  $\mathcal{F}$  includes all possible functions). For simple example, consider a prior such that the reward function is linear with probability 99%, and with the remaining probability it is drawn from some distribution with full support.

The main drawback — typical for all Bayesian models — is that the Bayesian assumption (that the problem instance is sampled from a prior) may be very idealized in practice, and/or the “true” prior may not be fully known.

**Hybrid approach.** One can, in principle, combine these two approaches: have a Bayesian prior over some, but not all of the uncertainty, and use worst-case analysis for the rest. To make this more precise, suppose the reward function  $\mu$  is fully specified by two parameters,  $\theta$  and  $\omega$ , and we have a prior on  $\theta$  but nothing is known about  $\omega$ . Then the hybrid approach would strive to prove a regret bound of the following form:

For each  $\omega$ , the regret of this algorithm in expectation over  $\theta$  is at most ... .

For a more concrete example, arms could correspond to points in  $[0, 1]$  interval, and we could have  $\mu(x) = \theta \cdot x + \omega$ , for parameters  $\theta, \omega \in \mathbb{R}$ , and we may have a prior on the  $\theta$ . Another example: the problem instances  $\mathcal{I}_j$  in the lower bound are parameterized by two things: the best arm  $a^*$  and the number  $\epsilon$ ; so, *e.g.*, we could have a uniform distribution over the  $a^*$ , but no information on the  $\epsilon$ .

## Chapter 4

# Thompson Sampling

We consider Bayesian bandits, and discuss an important algorithm for this setting called *Thompson Sampling* (also known as *posterior sampling*). It is the first bandit algorithm in the literature (Thompson, 1933). It is a very general algorithm, in the sense that it is well-defined for an arbitrary prior, and it is known to perform well in practice. The exposition will be self-contained; in particular I will introduce Bayesian concepts as I need them.

### 4.1 Bayesian bandits: preliminaries and notation

To recap, Bayesian bandit problem is defined as follows. We start with “bandits with IID rewards” which we have studied before, and make an additional *Bayesian assumption*: the bandit problem instance  $\mathcal{I}$  is drawn initially from some known distribution  $\mathbb{P}$  over problem instances (called the *prior*). The goal is to optimize *Bayesian regret*, defined as

$$\mathbb{E}_{\mathcal{I} \sim \mathbb{P}} [ \mathbb{E}[R(T)|\mathcal{I}] ],$$

where the inner expectation is the (expected) regret for a given problem instance  $\mathcal{I}$ , and the outer expectation is over the prior.

**Simplifications.** We make several assumptions to simplify presentation.

First, we assume that the (realized) rewards come from a *single-parameter family* of distributions: specifically, there is a family of distributions  $\mathcal{D}_\nu$  parameterized by a number  $\nu \in [0, 1]$  such that the reward of arm  $a$  is drawn from distribution  $\mathcal{D}_\nu$ , where  $\nu = \mu(a)$  is the mean reward of this arm. Typical examples are 0-1 rewards and Gaussian rewards with unit variance. Thus, the reward distribution for a given arm  $a$  is completely specified by its mean reward  $\mu(a)$ . It follows that the problem instance is completely specified by the reward function  $\mu$ , and so the prior  $\mathbb{P}$  is a distribution over the reward functions.

Second, we assume that there are only finitely many arms, the (realized) rewards can take only finitely many different values, and the prior  $\mathbb{P}$  has a finite support. Then we can focus on concepts and arguments essential to Thompson Sampling, rather than worry about the intricacies of probability densities, integrals and such. However, all claims stated below hold for arbitrary priors, and the exposition can be extended to infinitely many arms.

Third, we assume that the best arm  $a^*$  is unique for each reward function in the support of  $\mathbb{P}$ .

**Notation.** Let  $\mathcal{F}$  be the support of  $\mathbb{P}$ , *i.e.*, the set of all feasible reward functions. For a particular run of a particular algorithm on a particular problem instance, let  $h_t = (a_t, r_t)$  be the history for round  $t$ , where  $a_t$  is the chosen arm and  $r_t$  is the reward. Let  $H_t = (h_1, h_2, \dots, h_t)$  be the history up to time  $t$ . Let  $\mathcal{H}_t$  be the set of all possible histories  $H_t$ . As usual,  $[t]$  denotes the set  $\{1, 2, \dots, t\}$ .

**Sample space.** Consider a fixed bandit algorithm. While we defined  $\mathbb{P}$  as a distribution over reward functions  $\mu$ , we can also treat it as a distribution over the sample space

$$\Omega = \{(\mu, H_\infty) : \mu \in \mathcal{F}, H_\infty \in \mathcal{H}_\infty\},$$

the set of all possible pairs  $(\mu, H_t)$ . This is because the choice of  $\mu$  also specifies (for a fixed algorithm) the probability distribution over the histories. (And we will do the same for any distribution over reward functions.)

**Bayesian terminology.** Given time- $t$  history  $H_t$ , one can define a conditional distribution  $\mathbb{P}_t$  over the reward functions by  $\mathbb{P}_t(\mu) = \mathbb{P}[\mu|H_t]$ . Such  $\mathbb{P}_t$  is called the *posterior distribution*. The act of deriving the posterior distribution from the prior is called *Bayesian update*.

Say we have a quantity  $X = X(\mu)$  which is completely defined by the reward function  $\mu$ , such as the best arm for a given  $\mu$ . One can view  $X$  as a random variable whose distribution  $\mathbb{P}_X$  is induced by the prior  $\mathbb{P}$ . More precisely,  $\mathbb{P}_X$  is given by  $\mathbb{P}_X(x) = \mathbb{P}[X_\mu = x]$ , for all  $x$ . Such  $\mathbb{P}_X$  is called the *prior distribution* for  $X$ . Likewise, we can define the conditional distribution  $\mathbb{P}_{X,t}$  induced by the posterior  $\mathbb{P}_t$ : it is given by  $\mathbb{P}_{X,t}(x) = \mathbb{P}[X = x|H_t]$  for all  $x$ . This distribution is called *posterior distribution* for  $X$  at time  $t$ .

## 4.2 Thompson Sampling: definition and characterizations

**Main definition.** For each round  $t$ , consider the posterior distribution for the best arm  $a^*$ . Formally, it is distribution  $p_t$  over arms given by

$$p_t(a) = \mathbb{P}[a = a^* | H_t] \quad \text{for each arm } a. \quad (4.1)$$

Thompson Sampling is a very simple algorithm:

$$\text{In each round } t, \text{ arm } a_t \text{ is drawn independently from distribution } p_t. \quad (4.2)$$

Sometimes we will write  $p_t(a) = p_t(a|H_t)$  to emphasize the dependence on history  $H_t$ .

**Alternative characterization.** Thompson Sampling can be stated differently: in each round  $t$ ,

1. sample reward function  $\mu_t$  from the posterior distribution  $\mathbb{P}_t(\mu) = \mathbb{P}(\mu|H_t)$ .
2. choose the best arm  $\tilde{a}_t$  according to  $\mu_t$ .

Let us prove that this characterization is in fact equivalent to the original algorithm.

**Lemma 4.1.** *For each round  $t$  and each history  $H_t$ , arms  $a_t$  and  $\tilde{a}_t$  are identically distributed.*

*Proof.* For each arm  $a$  we have:

$$\begin{aligned} \Pr(\tilde{a}_t = a) &= \mathbb{P}_t(\text{arm } a \text{ is the best arm}) && \text{by definition of } \tilde{a}_t \\ &= \mathbb{P}(\text{arm } a \text{ is the best arm} | H_t) && \text{by definition of the posterior } \mathbb{P}_t \\ &= p_t(a | H_t) && \text{by definition of } p_t. \end{aligned}$$

Thus,  $\tilde{a}_t$  is distributed according to distribution  $p_t(a|H_t)$ . □

**Independent priors.** Things get simpler when we have independent priors. (We will state some properties without a proof.) Then for each arm  $a$  we have a prior  $\mathbb{P}_a$  for the mean reward  $\mu(a)$  for this arm, so that the “overall” prior is the product over arms:  $\mathbb{P}(\mu) = \prod_{\text{arms } a} \mathbb{P}_a(\mu(a))$ . The posterior  $\mathbb{P}_t$  is also a product over arms:

$$\mathbb{P}_t(\mu) = \prod_{\text{arms } a} \mathbb{P}_t^a(\mu(a)), \quad \text{where } \mathbb{P}_t^a(x) = \mathbb{P}[\mu(a) = x | H_t]. \quad (4.3)$$

So one simplification is that it suffices to consider the posterior on each arm separately.

Moreover, the posterior  $\mathbb{P}_t^a$  for arm  $a$  does not depend on the observations from other arms and (in some sense) it does not depend on the algorithm’s choices. Stating this formally requires some care. Let  $S_t^a$  be the vector of rewards received from arm  $a$  up to time  $t$ ; it is the  $n$ -dimensional vector,  $n = n_t(a)$ , such that the  $j$ -th component of this vector corresponds to the reward received the  $j$ -th time arm  $a$  has been chosen, for  $j \in [n_t(a)]$ . We treat  $S_t^a$  as a “summary” of the history of arm  $a$ . Further, let  $Z_t^a \in [0, 1]^t$  be a random vector distributed as  $t$  draws from arm  $a$ . Then for a particular realization of  $S_t^a$  we have

$$\mathbb{P}_t^a(x) := \mathbb{P}[\mu(a) = x | H_t] = \mathbb{P}^a[\mu(a) = x | Z_t^a \text{ is consistent with } S_t^a]. \quad (4.4)$$

Here two vectors  $v, v'$  of dimension  $n$  and  $n'$ , resp., are called *consistent* if they agree on the first  $\min(n, n')$  coordinates.

One can restate Thompson Sampling for independent priors as follows:



1. for each arm  $a$ , sample mean reward  $\mu_t(a)$  from the posterior distribution  $\mathbb{P}_t^a$ .
2. choose an arm with maximal  $\mu_t(a)$  (break ties arbitrarily).

### 4.3 Computational aspects

While Thompson Sampling is mathematically well-defined, the arm  $a_t$  may be difficult to compute efficiently. Hence, we have two distinct issues to worry about: algorithm's statistical performance (as expressed by Bayesian regret, for example), and algorithm's running time. It is may be the first time in this course when we have this dichotomy; for all algorithms previously considered, computationally efficient implementation was not an issue.

Ideally, we'd like to have both a good regret bound (RB) and a computationally fast implementation (FI). However, either one of the two is interesting: an algorithm with RB but without FI can serve as a proof-of-concept that such regret bound can be achieved, and an algorithm with FI but without RB can still achieve good regret in practice. Besides, due to generality of Thompson Sampling, techniques developed for one class of priors can potentially carry over to other classes.

**A brute-force attempt.** To illustrate the computational issue, let us attempt to compute probabilities  $p_t(a)$  by brute force. Let  $\mathcal{F}_a$  be the set of all reward functions  $\mu$  for which the best arm is  $a$ . Then:

$$p_t(a|H_t) = \frac{\mathbb{P}(a^* = a \ \& \ H_t)}{\mathbb{P}(H_t)} = \frac{\sum_{\mu \in \mathcal{F}_a} \mathbb{P}(\mu) \cdot \Pr[H_t|\mu]}{\sum_{\mu \in \mathcal{F}} \mathbb{P}(\mu) \cdot \Pr[H_t|\mu]}.$$

Thus,  $p_t(a|H_t)$  can be computed in time  $|\mathcal{F}|$  times the time needed to compute  $\Pr[H_t|\mu]$ , which may be prohibitively large if there are too many feasible reward functions.

**Sequential Bayesian update.** Faster computation can sometimes be achieved by using the alternative characterization of Thompson Sampling. In particular, one can perform the Bayesian update *sequentially*: use the prior  $\mathbb{P}$  and round-1 history  $h_1$  to compute round-1 posterior  $\mathbb{P}_1$ ; then treat  $\mathbb{P}_1$  as the new prior, and use  $\mathbb{P}_1$  and round-2 history  $h_2$  to compute round-2 posterior  $\mathbb{P}_2$ ; then treat  $\mathbb{P}_2$  as the new prior and so forth. Intuitively, the round- $t$  posterior  $\mathbb{P}_t$  contains all relevant information about the prior  $\mathbb{P}$  and the history  $H_t$ ; so once we have  $\mathbb{P}_t$ , one can forget the  $\mathbb{P}$  and the  $H_t$ . Let us argue formally that this is a sound approach:

**Lemma 4.2.** *Fix round  $t$  and history  $H_t$ . Then  $\mathbb{P}_t(\mu) = \mathbb{P}_{t-1}(\mu|h_t)$  for each reward function  $\mu$ .*

*Proof.* Let us use the definitions of conditional expectation and posterior  $\mathbb{P}_{t-1}$ :

$$\mathbb{P}_{t-1}(\mu|h_t) = \frac{\mathbb{P}_{t-1}(\mu \ \& \ h_t)}{\mathbb{P}_{t-1}(h_t)} = \frac{\mathbb{P}(\mu \ \& \ h_t \ \& \ H_{t-1})/\mathbb{P}(H_{t-1})}{\mathbb{P}(h_t \ \& \ H_{t-1})/\mathbb{P}(H_{t-1})} = \frac{\mathbb{P}(\mu \ \& \ H_t)}{\mathbb{P}(H_t)} = \mathbb{P}_t(\mu).$$

□

With independent priors, one can do the sequential update for each arm separately:

$$\mathbb{P}_t^a(\mu(a)) = \mathbb{P}_{t-1}^a(\mu(a)|h_t),$$

and only when this arm is chosen in this round.

### 4.4 Example: 0-1 rewards and Beta priors

Assume 0-1 rewards and independent priors. Let us provide some examples in which Thompson Sampling admits computationally efficient implementation.

The full  $t$ -round history of arm  $a$  is denoted  $H_t^a$ . We summarize it with two numbers:

- $\alpha_t(a)$ : the number of 1's seen for arm  $a$  till round  $t$ ,
- $n_t(a)$ : total number of samples drawn from arm  $a$  till round  $t$ .

If the prior for each arm is a uniform distribution over finitely many possible values, then we can easily derive a formula for the posterior.

**Lemma 4.3.** *Assume 0-1 rewards and independent priors. Further, assume that prior  $\mathbb{P}^a$  is a uniform distribution over  $N_a < \infty$  possible values, for each arm  $a$ . Then  $\mathbb{P}_t^a$ , the  $t$ -round posterior for arm  $a$ , is given by a simple formula:*

$$\mathbb{P}_t^a(x) = C \cdot x^\alpha (1-x)^{n-\alpha}$$

for every feasible value  $x$  for the mean reward  $\mu(a)$ , where  $\alpha = \alpha_t(a)$  and  $n = n_t(a)$ , and  $C$  is the normalization constant.

*Proof.* Fix arm  $a$ , round  $t$ , and a feasible value  $x$  for the mean reward  $\mu(a)$ . Fix a particular realization of history  $H_t$ , and therefore a particular realization of the summary  $S_t^a$ . Let  $A$  denote the event in (4.4): that the random vector  $Z_t^a$  is consistent with the summary  $S_t^a$ . Then:

$$\begin{aligned} \mathbb{P}_t^a(x) &= \mathbb{P}[\mu(a) = x | H_t] && \text{By definition of the arm-}a \text{ posterior} \\ &= \mathbb{P}^a[\mu(a) = x | A] && \text{by (4.4)} \\ &= \frac{\mathbb{P}^a(\mu(a) = x \text{ and } A)}{\mathbb{P}^a(A)} \\ &= \frac{\mathbb{P}^a[\mu(a) = x] \cdot \Pr(A | \mu(a) = x)}{\mathbb{P}^a(A)} \\ &= \frac{\frac{1}{N_a} \cdot x^\alpha (1-x)^{n-\alpha}}{\mathbb{P}^a(A)} \\ &= C x^\alpha (1-x)^{n-\alpha} \end{aligned}$$

for some normalization constant  $C$ . □

One can prove a similar result for a prior that is uniform over a  $[0, 1]$  interval; we present it without a proof. Note that in order to compute the posterior for a given arm  $a$ , it suffices to assume 0-1 rewards and uniform prior for this one arm.

**Lemma 4.4.** *Assume independent priors. Focus on a particular arm  $a$ . Assume this arm gives 0-1 rewards, and its prior  $\mathbb{P}^a$  is a uniform distribution on  $[0, 1]$ . Then the posterior  $\mathbb{P}_t^a$  is a distribution with density*

$$f(x) = \frac{(n+1)! \alpha!}{(n+\alpha)!} \cdot x^\alpha (1-x)^{n-\alpha}, \quad \forall x \in [0, 1],$$

where  $\alpha = \alpha_t(a)$  and  $n = n_t(a)$ .

A distribution with such density is called a *Beta distribution* with parameters  $\alpha + 1$  and  $n + 1$ , and denoted  $\text{Beta}(\alpha + 1, n + 1)$ . This is a well-studied distribution, e.g., see the corresponding Wikipedia page.  $\text{Beta}(1, 1)$  is the uniform distribution on the  $[0, 1]$  interval.

In fact, we have a more general version of Lemma 4.4, in which the prior can be an arbitrary Beta-distribution:

**Lemma 4.5.** *Assume independent priors. Focus on a particular arm  $a$ . Assume this arm gives 0-1 rewards, and its prior  $\mathbb{P}^a$  is a Beta distribution  $\text{Beta}(\alpha_0, n_0)$ . Then the posterior  $\mathbb{P}_t^a$  is a Beta distribution  $\text{Beta}(\alpha + \alpha_0, n + n_0)$ , where  $\alpha = \alpha_t(a)$  and  $n = n_t(a)$ .*

*Remark 4.6.* By Lemma 4.4, starting with  $\text{Beta}(\alpha_0, n_0)$  prior is the same as starting with a uniform prior and several samples of arm  $a$ . (Namely,  $n_0 - 1$  samples with exactly  $\alpha_0 - 1$  1's.)

## 4.5 Example: Gaussian rewards and Gaussian priors

Assume that the priors are Gaussian and independent, and the rewards are Gaussian, too. Then the posteriors are also Gaussian, and their mean and variance can be easily computed in terms of the parameters and the history.

**Lemma 4.7.** *Assume independent priors. Focus on a particular arm  $a$ . Assume this arm gives rewards that are Gaussian with mean  $\mu(a)$  and standard deviation  $\hat{\sigma}$ . Further, suppose the prior  $\mathbb{P}^a$  for this arm is Gaussian with mean  $\mu_0^a$  and standard deviation  $\sigma_0^a$ . Then the posterior  $\mathbb{P}_t^a$  is a Gaussian whose mean and variance are determined by the known parameters  $(\mu_0^a, \sigma_0^a, \hat{\sigma})$ , as well as the average reward and the number of samples from this arm so far.*

## 4.6 Bayesian regret

For each round  $t$ , we defined a posterior distribution over arms  $a$  as:

$$p_t(a) = \mathbb{P}(a = a^* | H_t)$$

where  $a^*$  is the best arm for the problem instance and  $H_t$  is the history of rewards up to round  $t$ . By definition, the Thompson algorithm draws arm  $a_t$  independently from this distribution  $p_t$ . If we consider  $a^*$  to be a random variable dependent on the problem instance,  $a_t$  and  $a^*$  are identically distributed. We will use this fact later on:

$$a_t \sim a^* \text{ when } H_t \text{ is fixed}$$

Our aim is to prove an upper bound on Bayesian regret for Thompson sampling. Bayesian regret is defined as:

$$\mathbb{E}_{\mu \sim \text{prior}} \left[ \mathbb{E}_{\text{rewards}} [R(t) | \mu] \right]$$

Notice that Bayesian regret is simply our usual regret placed inside of an expectation conditioned over problem instances. So a regular regret bound (holding over all problem instances) implies the same bound on Bayesian regret.

Recall our definition of the lower and upper confidence bounds on an arm's expected rewards at a certain time  $t$  (given history):

$$\begin{aligned} r_t(a) &= \sqrt{2 * \frac{\log(T)}{n_t(a)}} \\ UCB_t &= \bar{\mu}_t(a) + r_t(a) \\ LCB_t &= \bar{\mu}_t(a) - r_t(a) \end{aligned}$$

Here  $T$  is the time horizon,  $n_t(a)$  is the number of times arm  $a$  has been played so far, and  $\bar{\mu}_t(a)$  is the average reward from this arm so far. The quantity  $r_t(a)$  is called the *confidence radius*. As we've seen before,  $\mu(a) \in [LCB_t(a), UCB_t(a)]$  with high probability.

Now we want to generalize this formulation of upper and lower confidence bounds to be explicit functions of the arm  $a$  and the history  $H_t$  up to round  $t$ : respectively,  $U(a, H_t)$  and  $L(a, H_t)$ . There are two properties we want these functions to have, for some  $\gamma > 0$  to be specified later:

$$\forall a, t, \mathbb{E}[ [U(a, H_t) - \mu(a)]^- ] \leq \frac{\gamma}{T \cdot K} \quad (4.5)$$

$$\forall a, t, \mathbb{E}[ [\mu(a) - L(a, H_t)]^- ] \leq \frac{\gamma}{T \cdot K}. \quad (4.6)$$

As usual,  $K$  denotes the number of arms. As a matter of notation,  $x^-$  is defined to be the negative portion of the number  $x$ , that is, 0 if  $x$  is non-negative, and  $|x|$  if  $x$  is negative.

Intuitively, the first property says that the upper bound  $U$  does not exceed the mean reward by too much *in expectation*, and the second property makes a similar statement about the lower bound  $L$ .

The confidence radius is then defined as  $r(a, H_t) = \frac{U(a, H_t) - L(a, H_t)}{2}$ .

**Lemma 4.8.** *Assuming we have lower and upper bound functions that satisfy those two properties, the Bayesian Regret of Thompson sampling can be bounded as follows:*

$$BR(T) \leq 2\gamma + 2 \sum_{t=1}^T \mathbb{E}[r(a_t, H_t)] \quad (4.7)$$

*Proof.* First note that:

$$\mathbb{E}[U(a^*, H_t) | H_t] = \mathbb{E}[U(a_t, H_t) | H_t] \quad (4.8)$$

This is because, as noted previously,  $a^* \sim a_t$  for any fixed  $H_t$ , and  $U$  is deterministic.

Fix round  $t$ . Then the Bayesian Regret for that round is:

$$\begin{aligned} BR_t(T) &= \mathbb{E}[R(t)] && \text{expectation over everything} \\ &= \mathbb{E}[\mu(a^*) - \mu(a_t)] && \text{instantaneous regret for round } t \\ &= \mathbb{E}_{H_t} [\mathbb{E}[\mu(a^*) - \mu(a_t) | H_t]] && \text{bring out expectation over history} \\ &= \mathbb{E}_{H_t} [\mathbb{E}[U(a_t, H_t) - \mu(a_t) + \mu(a^*) - U(a^*, H_t) | H_t]] && \text{by the equality in equation 4.8 above} \\ &= \underbrace{\mathbb{E}[U(a_t, H_t) - \mu(a_t)]}_{\text{Part 1}} + \underbrace{\mathbb{E}[\mu(a^*) - U(a^*, H_t)]}_{\text{Part 2}}. \end{aligned}$$

We will use properties (4.5) and (4.6) to bound Part 1 and Part 2. Note that we cannot *immediately* use these properties because they assume a fixed arm  $a$ , whereas both  $a_t$  and  $a^*$  are random variables. (The best arm  $a^*$  is a random variable because we take an expectation over everything, including the problem instance.)

Let's handle Part 2:

$$\begin{aligned} \mathbb{E}[\mu(a^*) - U(a^*, H_t)] &\leq \mathbb{E}[(\mu(a^*) - U(a^*, H_t))^+] \\ &\leq \mathbb{E}\left[\sum_{\text{arms } a} (\mu(a) - U(a, H_t))^+\right] \\ &= \mathbb{E}\left[\sum_{\text{arms } a} (\mu(a) - U(a, H_t))^+\right] \\ &= \sum_{\text{arms } a} \mathbb{E}[(U(a, H_t) - \mu(a))^-] \\ &\leq k * \frac{\gamma}{T * k} && \text{by 4.5 over all arms} \\ &= \frac{\gamma}{T} \end{aligned}$$

Let's handle Part 1:

$$\begin{aligned} \mathbb{E}[U(a_t, H_t) - \mu(a_t)] &= \mathbb{E}[2r(a_t, H_t) + L(a_t, H_t) - \mu(a_t)] && \text{from definition of } r \\ &= \mathbb{E}[2r(a_t, H_t)] + \mathbb{E}[L(a_t, H_t) - \mu(a_t)] \end{aligned}$$

Then

$$\begin{aligned} \mathbb{E}[L(a_t, H_t) - \mu(a_t)] &\leq \mathbb{E}[(L(a_t, H_t) - \mu(a_t))^+] \\ &\leq \mathbb{E}\left[\sum_{\text{arms } a} ((L(a, H_t) - \mu(a))^+)\right] \\ &= \sum_{\text{arms } a} \mathbb{E}[(\mu(a) - L(a, H_t))^-] \\ &\leq k * \frac{\gamma}{T * k} && \text{by 4.6 over all arms} \\ &= \frac{\gamma}{T} \end{aligned}$$

Putting parts 1 and 2 together, (for fixed  $t$ )  $BR_t(T) \leq \frac{\gamma}{T} + \frac{\gamma}{T} + \mathbb{E}[2r(a_t, H_t)]$ .

Summing up over  $t$ ,  $BR(T) \leq 2\gamma + 2 \sum_{t=1}^T \mathbb{E}[r(a_t, H_t)]$  as desired.  $\square$

*Remark 4.9.* Lemma 4.8 holds regardless of what the  $U$  and  $L$  functions are, so long as they satisfy properties (4.5) and (4.6). Furthermore, Thompson Sampling does not need to know what  $U$  and  $L$  are, either.

*Remark 4.10.* While Lemma 4.8 does not assume any specific structure of the prior, it embodies a general technique: it can be used to upper-bound Bayesian regret of Thompson Sampling for arbitrary priors, and it also for a particular class of priors (e.g., priors over linear reward functions) whenever one has “nice” confidence bounds  $U$  and  $L$  for this class.

Let us use Lemma 4.8 to prove a  $\mathcal{O}(\sqrt{KT \log T})$  upper bound on regret, which matches the best possible result for Bayesian regret of  $K$ -armed bandits.

**Theorem 4.11.** *Over  $k$  arms,  $BR(T) = \mathcal{O}(\sqrt{kT \log(T)})$*

*Proof.* Let the confidence radius be  $r(a, H_t) = \sqrt{\frac{2 \log T}{n_t(a)}}$  and  $\gamma = 2$  where  $n_t(a)$  is the number of rounds arm  $a$  is played up to time  $t$ . Then, by lemma above,

$$BR(T) \leq a + 2 \sum_{t=1}^T \mathbb{E} \left[ \sqrt{\frac{2 \log T}{n_t(a)}} \right] = \mathcal{O}(\sqrt{\log T}) \sum_{t=1}^T \mathbb{E} \left[ \sqrt{\frac{1}{n_t(a)}} \right]$$

Additionally,

$$\begin{aligned} \sum_{t=1}^T \sqrt{\frac{1}{n_t(a)}} &= \sum_{\text{arms } a} \sum_{t: a_t=a} \frac{1}{\sqrt{n_t(a)}} \\ &= \sum_{\text{arms } a} \sum_{j=1}^{n(a)} \frac{1}{\sqrt{j}} && \text{n(a) is total times arm a is picked} \\ &= \sum_{\text{arms } a} \mathcal{O}(\sqrt{n(a)}) && \text{by taking an integral} \end{aligned}$$

So,

$$BR(T) \leq \mathcal{O}(\sqrt{\log T}) \sum_{\text{arms } a} \sqrt{n(a)} \leq \mathcal{O}(\sqrt{\log T}) \sqrt{k \sum_{\text{arms } a} n(a)} = \mathcal{O}(\sqrt{kT \log T}),$$

where the last inequality is using the fact that the arithmetic mean is less than (or equal to) the quadratic mean.  $\square$

## 4.7 Thompson Sampling with no prior (and no proofs)

We can use Thompson Sampling for the “basic” bandit problem that we have studied before: the bandit problem with IID rewards in  $[0, 1]$ , but without priors on the problem instances.

We can treat a prior just as a parameter to Thompson Sampling (rather than a feature of reality). This way, we can consider an arbitrary prior (we’ll call it a “fake prior”, and it will give a well-defined algorithm for IID bandits without priors. We This approach makes sense as long as this algorithm performs well.

Prior work considered two such “fake priors”:

- (i) independent, uniform priors and 0-1 rewards,
- (ii) independent, Gaussian priors and Gaussian unit-variance rewards (so each reward is distributed as  $\mathcal{N}(\mu(a), 1)$ , where  $\mu(a)$  is the mean).

To fully specify approach (i), we need to specify how to deal with rewards  $r$  that are neither 0 or 1; this can be handled very easily: flip a random coin with expectation  $r$ , and pass the outcome of this coin flip as a reward to Thompson Sampling. In approach (ii), note that the prior allows the realized rewards to be arbitrarily large, whereas we assume the rewards are bounded on  $[0, 1]$ ; this is OK because the algorithm is still well-defined.

We will state the regret bounds for these two approaches, without a proof.

**Theorem 4.12.** Consider IID bandits with no priors. For Thompson Sampling with both approaches (i) and (ii) we have:  $\mathbb{E}[R(T)] \leq \mathcal{O}(\sqrt{kT \log T})$ .

**Theorem 4.13.** Consider IID bandits with no priors. For Thompson sampling with approach (i),

$$\mathbb{E}[R(T)] \leq (1 + \epsilon)(\log T) \underbrace{\sum_{\substack{\text{arms } a \\ \text{s.t. } \Delta(a) > 0}} \frac{\Delta(a)}{KL(\mu(a), \mu^*)}}_{(*)} + \frac{f(\mu)}{\epsilon^2},$$

for all  $\epsilon > 0$ . Here  $f(\mu)$  depends on the reward function  $\mu$ , but not on the  $\epsilon$ , and  $\Delta(a) = \mu(a^*) - \mu(a)$ .

The (\*) is the optimal constant: it matches the constant in the logarithmic lower bound which we have seen before. So this regret bound gives a partial explanation for why Thompson Sampling is so good in practice. However, it is not quite satisfactory because the term  $f(\mu)$  can be quite big, as far as they can prove.

**Interlude B:**

**Practical Aspects (to be added)**





## Chapter 5

# Lipschitz Bandits

**Motivation: similarity between arms.** In various bandit problems, we may have information on similarity between arms, in the sense that ‘similar’ arms have similar expected rewards. For example, arms can correspond to “items” (e.g., documents) with feature vectors, and similarity can be expressed as some notion of distance between feature vectors. Another example would be the dynamic pricing problem, where we have numerical similarity between prices, and it is often assumed that similar prices correspond to similar expected rewards.

Throughout this lecture, we will consider bandits with IID rewards: the reward for each arm  $x$  is an independent sample from some fixed distribution with expectation  $\mu(x)$ . We use  $x, y$  to denote arms, as they will also denote “points” on a line or in a metric space.

### 5.1 Continuum-armed bandits

Let us start with a special case called *continuum-armed bandits* (CAB). In this problem, the set of arms is  $X = [0, 1]$ , and we have a *Lipschitz Condition*:

$$|\mu(x) - \mu(y)| \leq L \cdot |x - y| \quad \text{for any two arms } x, y \in X, \quad (5.1)$$

where  $L$  is a constant known to the algorithm.<sup>1</sup>

Note that we have infinitely many arms, and in fact, *continuously* many. While the MAB problem with infinitely many arms is hopeless in general, CAB turns out tractable due to the Lipschitz condition.

#### 5.1.1 Simple solution: fixed discretization

The idea is to pick a fixed, finite set of arms  $S \subset X$ , and use this set as an “approximation” for the full set  $X$ . Then we focus only on arms in  $S$ , and run an off-the-shelf MAB algorithm ALG, such as UCB1 or Successive Elimination, that only considers these arms. In this context, the set  $S$  is called a *discretization* of  $X$ .

The best arm in  $S$  will be denoted  $\mu^*(S) = \sup_{x \in S} \mu(x)$ . In each round, algorithm ALG can only hope to approach expected reward  $\mu^*(S)$ , and additionally suffers *discretization error*

$$\text{DE}(S) = \mu^*(X) - \mu^*(S). \quad (5.2)$$

More formally, we can represent regret of ALG as a sum

$$\begin{aligned} R(T) &= \mu^*(X) - W(\text{ALG}) \\ &= (\mu^*(S) - W(\text{ALG})) + (\mu^*(X) - \mu^*(S)) \\ &= R_S(T) + \text{DE}(S), \end{aligned}$$

---

<sup>1</sup>A function  $\mu : X \rightarrow \mathbb{R}$  which satisfies (5.1) is called *Lipschitz-continuous* on  $X$ , with *Lipschitz constant*  $L$ .

where  $W(ALG)$  is the total reward of the algorithm, and  $R_S(T)$  is the regret relative to  $\mu^*(S)$ .

Suppose ALG attains optimal regret  $O(\sqrt{KT \log T})$  on any problem instance with time horizon  $T$  and  $K$  arms. Then

$$\mathbb{E}[R(T)] \leq O(\sqrt{|S|T \log T}) + \text{DE}(S) \cdot T. \quad (5.3)$$

Thus, we have a tradeoff between the size of  $S$  and its discretization error.

One simple and natural solution is to use *uniform discretization* with  $k$  arms: divide the interval  $[0, 1]$  into intervals of fixed length  $\epsilon = \frac{1}{k-1}$ . Then

$$S = \{i \cdot \epsilon : i = 0, 1, 2, \dots, k-1\} \subset X.$$

It is easy to see that  $\text{DE}(S) \leq L\epsilon$ . Picking  $\epsilon = (TL^2 / \log T)^{-1/3}$ , we obtain

$$\mathbb{E}[R(T)] \leq O\left(L^{1/3} \cdot T^{2/3} \cdot \log^{1/3}(T)\right).$$

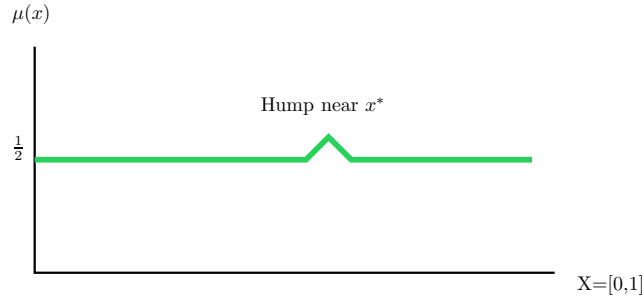
### 5.1.2 Lower Bound

The simple solution described above is in fact optimal in the worst case: we have an  $\Omega(T^{2/3})$  lower bound on regret. We prove this lower bound via a relatively simple reduction from the “main lower bound”: the  $\Omega(\sqrt{KT})$  lower bound that we’ve seen in Lecture 4 (henceforth called **MainLB**).

The new lower bound will involve problem instances with 0-1 rewards such that all arms have mean reward  $\mu(x) = \frac{1}{2}$  except those near the unique best arm  $x^*$ , whose mean reward is  $\mu(x^*) = \frac{1}{2} + \epsilon$ . More formally, let us define a problem instance  $\mathcal{I}(x^*, \epsilon)$  by

$$\mu(x) = \begin{cases} \frac{1}{2}, & |x - x^*| \geq \epsilon/L \\ \frac{1}{2} + \epsilon - L \cdot |x - x^*|, & \text{otherwise.} \end{cases} \quad (5.4)$$

It is easy to see that any such problem instance satisfies the Lipschitz Condition (5.1). Note that we need to have a small “bump” near  $x^*$ , due to the Lipschitz Condition. Informally, we will refer to  $\mu(\cdot)$  as the “bump function”:



The lower bound for CAB is stated as follows:

**Theorem 5.1** (Lower bound for CAB). *Consider CAB with time horizon  $T$ . Let ALG be any algorithm for this problem. There exists a problem instance  $\mathcal{I} = \mathcal{I}(x^*, \epsilon)$  such that*

$$\mathbb{E}[R(T) \mid \mathcal{I}] \geq \Omega(T^{2/3}). \quad (5.5)$$

The intuition is as follows. Fix  $K \in \mathbb{N}$ , let  $\epsilon = 1/2K$ , and assume  $L = 1$  for simplicity. Focus on problem instances  $\mathcal{I}(a^*/K, \epsilon)$ , where  $a^* \in [K] := \{1, \dots, K\}$ . Thus, the set of arms  $X = [0, 1]$  is partitioned into  $K$  disjoint intervals of length  $2\epsilon$ , and the ‘bump’ can be in any one of these intervals. Further, whenever an algorithm chooses an arm  $x$  in one such interval, choosing an arm in the *center* of the same interval appears to be the best option: either this interval contains a bump, and the center is the best arm, or all arms in this interval have the same mean

reward  $\frac{1}{2}$ . But if an algorithm only chooses among arms that are multiples of  $1/K$ , we have  $K$ -armed bandit problem instances where all arms have mean reward  $\frac{1}{2}$  except one with mean reward  $\frac{1}{2} + \epsilon$ . Which is precisely the family of instances from MainLB. Then one can apply the lower bound from MainLB, and tune the parameters to obtain (5.5). To turn this intuition into a proof, the main obstacle is to argue that “choosing the center of the same interval is the best option”.

Let us recap MainLB, restating it in a way that is convenient for this lecture. Recall that MainLB considered problem instances  $\mathcal{I}(a^*, \epsilon)$  with 0-1 rewards such that all arms  $a$  have mean reward  $\frac{1}{2}$ , except the best arm  $a^*$  whose mean reward is  $\frac{1}{2} + \epsilon$ .

**Theorem 5.2 (MainLB).** *Consider bandits with IID rewards, with  $K$  arms and time horizon  $T$  (for any  $K, T$ ). Let ALG be any algorithm for this problem. Pick any positive  $\epsilon \leq \sqrt{cK/T}$ , where  $c$  is an absolute constant from Lemma 1.1 in Lecture 4. Then there exists a problem instance  $\mathcal{I} = \mathcal{I}(a^*, \epsilon)$*

$$\mathbb{E}[R(T) \mid \mathcal{I}] \geq \Omega(\epsilon T).$$

To prove Theorem 5.1, we show how to reduce the problem instances from MainLB to CAB in a way that we can apply MainLB and derive the claimed lower bound (5.5). On a high level, our plan is as follows: (i) take any problem instance  $\mathcal{I}$  from MainLB and “embed” it into CAB, and (ii) show that any algorithm for CAB will, in fact, need to solve  $\mathcal{I}$ , (iii) tune the parameters to derive (5.5).

*Proof of Theorem 5.1.* For simplicity of exposition, assume the Lipschitz constant is  $L = 1$  (arbitrary  $L$  is treated similarly). Let  $K \in \mathbb{N}$  be a parameter to be fixed later, and let  $\epsilon = 1/2K$ .

We will consider problem instances from MainLB with  $K$  arms and the same time horizon  $T$ ; the set of arms will be denoted as  $[K] = \{1, 2, \dots, K\}$ . Each problem instance  $\mathcal{I} = \mathcal{I}(a^*, \epsilon)$  from MainLB will correspond to an instance  $\mathcal{I} = \mathcal{I}(x^*, \epsilon)$  of CAB with  $x^* = a^*/K$ ; in particular, each arm  $a \in [K]$  in  $\mathcal{I}$  corresponds to an arm  $x = a/K$  in  $\mathcal{I}$ . We view  $\mathcal{I}$  as a discrete version of  $\mathcal{I}$ ; in particular, note that  $\mu_{\mathcal{I}}(a) = \mu(a/K)$ , where  $\mu(\cdot)$  is the reward function for  $\mathcal{I}$ , and  $\mu_{\mathcal{I}}(\cdot)$  is the reward function for  $\mathcal{I}$ .

Let us consider ALG as applied to problem instance  $\mathcal{I}$  of CAB, and use it as a subroutine to construct an algorithm ALG’ which solves instance  $\mathcal{I}$  of  $K$ -armed bandits. Each round in algorithm ALG’ proceeds as in the following table:

ALG for CAB instance $\mathcal{I}$	ALG’ for $K$ -armed bandits instance $\mathcal{I}$
chooses arm $x \in [\frac{a}{K} - \epsilon, \frac{a}{K} + \epsilon), a \in [K]$	chooses arm $a$
receives reward $r_x \in \{0, 1\}$ with mean $\mu(x)$	receives reward $r \in \{0, 1\}$ with mean $\mu_{\mathcal{I}}(a)$

In other words, ALG is called and returns some arm  $x \in [0, 1]$ . This arm falls into the interval  $[\frac{a}{K} - \epsilon, \frac{a}{K} + \epsilon)$ , for some  $a \in [K]$ . Then algorithm ALG’ chooses arm  $a$ . When ALG’ receives reward  $r$ , it uses  $r$  and  $x$  to compute reward  $r_x \in \{0, 1\}$  such that  $\mathbb{E}[r_x \mid x] = \mu(x)$ , and feed it back to ALG.

To complete the construction, it remains to define  $r_x$  so that  $\mathbb{E}[r_x \mid x] = \mu(x)$ . and  $r_x$  can be computed using information available to ALG’ in a given round. (In particular, the computation of  $r_x$  cannot use  $\mu_{\mathcal{I}}(a)$  or  $\mu(x)$ , since they are not know to ALG’.) We define  $r_x$  as follows:

$$r_x = \begin{cases} r & \text{with probability } p_x \\ X & \text{otherwise} \end{cases} \quad \text{where } p_x = 1 - |x - \frac{a}{K}|/\epsilon, \quad (5.6)$$

and  $X$  is an independent toss of an unbiased random coin, i.e., a 0-1 random variable with expectation  $\frac{1}{2}$ . Then

$$\begin{aligned} \mathbb{E}[r_x \mid x] &= p_x \cdot \mu_{\mathcal{I}}(a) + (1 - p_x) \cdot \frac{1}{2} \\ &= \frac{1}{2} + (\mu_{\mathcal{I}}(a) - \frac{1}{2}) \cdot p_x \\ &= \begin{cases} \frac{1}{2} & \text{if } a \neq a^* \\ \frac{1}{2} + \epsilon p_x & \text{if } a = a^* \end{cases} \\ &= \mu(x). \end{aligned}$$

For each round  $t$ , let  $x_t$  and  $a_t$  be the arms chosen by ALG' and ALG, respectively. By construction, the mean reward of  $x_t$  cannot exceed that of  $a_t$ ; rigorously:

$$\mathbb{E}[\mu_{\mathcal{I}}(a_t) \mid x_t] \geq \mathbb{E}[\mu(x_t)].$$

It follows that the total expected reward of ALG on instance  $\mathcal{I}$  cannot exceed that of ALG' on instance  $\mathcal{I}$ . Since the best arms in both problem instances have the same expected rewards  $\frac{1}{2} + \epsilon$ , it follows that

$$\mathbb{E}[R(T) \mid \mathcal{I}] \geq \mathbb{E}[R'(T) \mid \mathcal{I}],$$

where  $R(T)$  and  $R'(T)$  denote regret of ALG and ALG', respectively.

Recall that algorithm ALG' can solve any  $K$ -armed bandits instance of the form  $\mathcal{I} = \mathcal{I}(a^*, \epsilon)$ . Let us apply Theorem 5.2 to derive a lower bound on the regret of ALG'. Specifically, let us fix  $K = (T/4c)^{1/3}$ , so as to ensure that  $\epsilon \leq \sqrt{cK/T}$ , as required in Theorem 5.2. Then there exists some instance  $\mathcal{I} = \mathcal{I}(a^*, \epsilon)$  such that

$$\mathbb{E}[R'(T) \mid \mathcal{I}] \geq \Omega(\sqrt{\epsilon T}) = \Omega(T^{2/3})$$

Thus, taking the corresponding instance  $\mathcal{I}$  of CAB, we conclude that  $\mathbb{E}[R(T) \mid \mathcal{I}] \geq \Omega(T^{2/3})$ .  $\square$

## 5.2 Lipschitz MAB

The *Lipschitz MAB Problem* is a multi-armed bandit problem with IID rewards, where the reward function  $\mu$  satisfies the *Lipschitz condition*:

$$|\mu(x) - \mu(y)| \leq \mathcal{D}(x, y) \quad \text{for any two arms } x, y. \quad (5.7)$$

Here  $\mathcal{D}$  is a metric on arms which is known to the algorithm. The set of arms  $X$  can be finite or infinite. Recall that  $\mu(x)$  denotes the mean reward of arm  $x$ , and the realized rewards are in  $[0, 1]$ .

Informally, the metric  $\mathcal{D}$  defines a notion of similarity between arms such that similar arms have similar mean rewards. Note that w.l.o.g.  $\mathcal{D}(x, y) \leq 1$ .

### 5.2.1 (Very brief) background on metric spaces

A Metric Space is a pair  $(X, \mathcal{D})$ , where  $\mathcal{D}$  is a *metric*: a function  $\mathcal{D} : X \times X \rightarrow \mathbb{R}$  which represents “distance” between the elements of  $X$ . Formally, a metric satisfies the following axioms:

- Non-negativity:  $\mathcal{D}(x, y) \geq 0$ .
- Zero iff equal:  $\mathcal{D}(x, y) = 0$  iff  $x = y$
- Symmetry:  $\mathcal{D}(x, y) = \mathcal{D}(y, x)$
- Triangle inequality:  $\mathcal{D}(x, y) + \mathcal{D}(y, z) \geq \mathcal{D}(x, z)$

For  $Y \subset X$ , the pair  $(Y, \mathcal{D})$  is also a metric space, where (by a slight abuse of notation)  $\mathcal{D}$  denotes the same metric restricted to the points in  $Y$ .

Some examples:

- in CAB,  $X = [0, 1]$  and the metric is  $\mathcal{D}(x, y) = |x - y|$ .
- more generally, one could consider  $X = [0, 1]^d$ , and the distance function is the  $p$ -norm,  $p \geq 1$ :

$$\ell_p(x, y) = \|x - y\|_p := \left( \sum_{i=1}^d (x_i - y_i)^p \right)^{1/p}.$$

Most commonly considered are  $\ell_1$ -metric (a.k.a. Manhattan metric) and  $\ell_2$ -metric (a.k.a. Euclidean distance).

- $X$  can be any subset of  $[0, 1]^d$ , finite or infinite, under the same distance function  $\ell_p$ ,  $p \geq 1$ .
- The shortest-path metric of a graph:  $X$  is the nodes of a graph, and  $\mathcal{D}(x, y)$  is the length of a shortest path between  $x$  and  $y$ .

## 5.2.2 Fixed discretization

The developments in Section 5.1.1 carry over word-by-word, up until (5.3). As before, for a given  $\epsilon > 0$  we would like to pick a subset  $S \subset X$  with discretization error  $\text{DE}(S) \leq \epsilon$  and  $|S|$  as small as possible.

**Example 5.3.** Suppose the metric space is  $X = [0, 1]^d$  under the  $\ell_p$  metric,  $p \geq 1$ . Let us extend the notion of *uniform discretization* from Section 5.1.1: consider a subset  $S \subset X$  that consists of all points whose coordinates are multiples of a given  $\epsilon > 0$ . Then  $S$  consists of  $(\lceil 1/\epsilon \rceil)^d$  points, and its discretization error is  $\text{DE}(S) \leq c_{p,d} \cdot \epsilon$ , where  $c_{p,d}$  is a constant that depends only on  $p$  and  $d$  (e.g.,  $c_{p,d} = d$  for  $p = 1$ ). Plugging this into (5.3), we obtain

$$\mathbb{E}[R(T)] \leq O\left(\sqrt{\left(\frac{1}{\epsilon}\right)^d \cdot T \log T} + \epsilon T\right) = O\left(T^{\frac{d+1}{d+2}} (c \log T)^{\frac{1}{d+2}}\right), \quad (5.8)$$

where the last equality holds if we take  $\epsilon = (\frac{\log T}{T})^{1/(d+2)}$ .

As it turns out, the  $\tilde{O}(T^{(d+1)/(d+2)})$  regret in the above example is typical for Lipschitz MAB problem. However, we will need to define a suitable notion of “dimension” that will apply to an arbitrary metric space.

Let us generalize the notion of uniform discretization to an arbitrary metric space.

**Definition 5.4.** A subset  $S \subset X$  is called an  $\epsilon$ -mesh if  $\forall x \in X \exists y \in S$  such that  $\mathcal{D}(x, y) \leq \epsilon$ .

In words,  $S$  is an  $\epsilon$ -mesh if every point in  $X$  is within distance  $\epsilon$  from some point in  $S$ . It is easy to see that  $\text{DE}(S) \leq \epsilon$ . In what follows, we characterize the smallest size of an  $\epsilon$ -mesh, using some notions from the analysis of metric spaces.

**Definition 5.5** ( $\epsilon$ -covering and covering number). A collection of subsets  $X_i \subset X$  such that the diameter of each subset is at most  $\epsilon$  and  $X = \bigcup X_i$ . The smallest number of subsets in an  $\epsilon$ -covering is called the *covering number* and denoted  $N_\epsilon(X)$ .

The *diameter* of a metric space is the maximal distance between any two points; the diameter of a subset  $X' \subset X$  is the diameter of  $(X', \mathcal{D})$ , as long as the metric  $\mathcal{D}$  is clear from the context.

The covering number characterizes the “complexity” of a metric space (in a specific sense that happens to be relevant to Lipschitz MAB). This notion of complexity is “robust”, in the sense that a subset  $X' \subset X$  cannot be more “complicated” than  $X$ : indeed,  $N_\epsilon(X') \leq N_\epsilon(X)$ .

**Fact 5.6.** Consider an  $\epsilon$ -covering  $\{X_1, \dots, X_N\}$ . For each subset  $X_i$ , pick an arbitrary representative  $x_i \in X_i$ . Then  $S = \{x_1, \dots, x_N\}$  is an  $\epsilon$ -mesh.

Thus, for each  $\epsilon > 0$  there exists an  $\epsilon$ -mesh of size  $N_\epsilon(X)$ , so we can plug it into (5.3). But how do we optimize over all  $\epsilon > 0$ ? It is convenient to characterize  $N_\epsilon(X)$  for all  $\epsilon$  simultaneously:

**Definition 5.7.** The *covering dimension* of  $X$ , with multiplier  $c > 0$ , is

$$\text{COV}_c(X) = \inf_{d \geq 0} \left\{ N_\epsilon(X) \leq \frac{c}{\epsilon^d} \quad \forall \epsilon \geq 0 \right\}.$$

In particular, the covering dimension in Example 5.3 is  $d$  (with an appropriate multiplier). The covering dimension of a subset  $X' \subset X$  cannot exceed that of  $X$ : indeed,  $\text{COV}_c(X') \leq \text{COV}_c(X)$ .

Now one can plug in  $|S| = N_\epsilon(X) \leq c/\epsilon^d$  into (5.3), and do the same computation as in Example 5.3. Therefore, we obtain the following theorem:

**Theorem 5.8** (fixed discretization). Consider Lipschitz MAB problem on a metric space  $(X, \mathcal{D})$ , with time horizon  $T$ . Fix any  $c > 0$ , and let  $d = \text{COV}_c(X)$  be the covering dimension. There exists a subset  $S \subset X$  such that if one runs UCB1 using  $S$  as a set of arms, one obtains regret

$$\mathbb{E}[R(T)] = O\left(T^{\frac{d+1}{d+2}} (c \log T)^{\frac{1}{d+2}}\right).$$

There is a matching lower bound of  $\Omega(T^{(d+1)/(d+2)})$ . This lower bound can be achieved on  $X = [0, 1]^d$ , with any  $\ell_p$ -metric,  $p \geq 1$ . It can be proved similarly to Theorem 5.1.

### 5.3 Adaptive discretization: the Zooming Algorithm

Despite the existence of a matching lower bound, the fixed discretization approach is wasteful:

1. Thinking of arms in  $S$  as “probes”: the probes are placed uniformly in the metric space, and in the regret analysis there is a penalty associated with each probe. Whereas it is better to have less probes, as long as the best arm is “covered”.
2. Similarity information is discarded once we select the mesh  $S$ .

To elaborate on the first point, observe that the discretization error of  $S$  is at most the minimal distance between  $S$  and the best arm  $x^*$ :

$$\text{DE}(S) \leq \mathcal{D}(S, x^*) := \min_{x \in S} \mathcal{D}(x, x^*).$$

So it should help to decrease  $|S|$  while keeping  $\mathcal{D}(S, x^*)$  constant. In particular, if we know that  $x^*$  lies in a particular “region” of the metric space, then we do not need to place probes in other regions. Unfortunately, we do not know in advance where  $x^*$  is, so we cannot optimize  $S$  that way — at least, not in the fixed discretization approach!

However, one may approximately learn the mean rewards of arms over time, and adjust the placement of the probes accordingly, making sure that one has more probes in more “promising” regions of the metric space. This approach is called *adaptive discretization*. Below we describe one implementation of this approach, called the *zooming algorithm*.

What can we hope to improve with this algorithm, given the existence of a matching lower bound? So the goal here is to attain the same worst-case regret as in Theorem 5.8, but do “better” on “nice” problem instances. An important aspect of the overall research challenge here is to quantify the “better” and “nice” in the previous sentence.

The zooming algorithm will bring together three techniques: the “UCB technique” from algorithm UCB1, the new technique of “adaptive discretization”, and the “clean event” technique in the analysis.

In what follows, we make two assumptions to simplify presentation:

- There is a known time horizon  $T$ .
- The realized rewards can take only finitely many possible values.

Both assumptions can be relaxed with a little more work.

#### 5.3.1 Algorithm

The algorithm maintains a set  $S \subset X$  of “active arms”. In each round, some arms may be “activated” according to the “activation rule”, and one active arm is selected according to the “selection rule”. Once an arm is activated, it cannot be “deactivated”. That is the whole algorithm; we just need to specify the activation rule and the selection rule.

**Confidence radius/ball.** In each round  $t$ , for each active arm  $x$ , let  $n_t(x)$  is the number of samples for this arm,  $\mu_t(x)$  is the reward so far. The confidence radius is defined as

$$r_t(x) = \sqrt{\frac{2 \log T}{n_t(x) + 1}}.$$

The *confidence ball* of arm  $x$  is a closed ball in the metric space with center  $x$  and radius  $r_t(x)$ .

$$\mathbf{B}_t(x) = \{y \in X : \mathcal{D}(x, y) \leq r_t(x)\}.$$

**Activation rule.** The activation rule is very simple:

If some arm  $y$  becomes uncovered by confidence balls of the active arms, activate  $y$ .

The intuition is that for any arm  $y \in \mathbf{B}_t(x)$ , the algorithm does not have enough samples of  $x$  to distinguish  $x$  and  $y$ , so the two arms look the same, as far as the algorithm is concerned. Therefore the algorithm can sample arm  $x$

instead of arm  $y$ . So, there is no reason to activate  $y$  yet. Going further with this reasoning, we'd like to maintain the invariant:

In each round, all arms are covered by confidence balls of the active arms.

As the algorithm plays some arms over time, their confidence radii (and hence the confidence balls) get smaller, and some arm  $y$  may become uncovered. Then we simply activate it! Since immediately after activation the confidence ball of  $y$  includes the entire metric space, we see that the invariant is preserved.

Due to the activation rule, the zooming algorithm has the following “self-adjusting property”. The algorithm “zooms in” on a given region  $R$  of the metric space (i.e., activates many arms in  $R$ ) if and only if the arms in  $R$  are played often. The latter happens (under any reasonable selection rule) if and only if the arms in  $R$  have high mean rewards.

**Selection rule.** We extend the UCB technique from algorithm UCB1. For each active arm  $x$  we define an *index* at time  $t$  as

$$\text{index}_t(x) = \bar{\mu}_t(x) + 2r_t(x)$$

The selection rule is very simple:

Play an active arm with the largest index (break ties arbitrarily).

Recall that algorithm UCB1 chooses an arm with largest UCB on the mean reward, where the UCB is defined as  $UCB_t(x) = \bar{\mu}_t(x) + r_t(x)$ . So  $\text{index}_t(x)$  is very similar, and shares the intuition behind UCB1: if  $\text{index}_t(x)$  is large, then either  $\bar{\mu}_t(x)$  is large (so  $x$  is a good arm), or  $r_t(x)$  is large (so arm  $x$  has not been played very often, and should be explored more). And the ‘+’ in the index is a way to trade off exploration and exploitation.

What is new here, compared to UCB1, is that  $\text{index}_t(x)$  is a UCB not only on the mean reward of  $x$ , but also on the mean reward of any arm in the confidence ball of  $x$ .

**Summary.** To sum up the algorithm:

- Maintain a set of active arms  $S$ .
- Initially  $S = \emptyset$ , activate arms one by one.
- In each round  $t$ ,
  - Activate uncovered arms according to *Activation Rule*.
  - Play the active arm with the largest index  $\text{index}_t(x)$ .

### 5.3.2 Analysis: clean event

As in Lecture 2, we define a “clean event”  $\mathcal{E}$  and prove that it holds with high probability. Then the rest of the analysis can safely assume that this event holds. The proof that  $\mathcal{E}$  holds with high probability is more delicate than the corresponding proofs in Lecture 2, essentially because we cannot immediately take the Union Bound over all of  $X$ .

As in Lecture 2, we consider the table of the realized rewards: a row for each arm and  $T$  columns, where the  $j$ -th column for arm  $x$  is the reward for the  $j$ -time this arm is chosen by the algorithm. Let us assume, w.l.o.g., that this entire table is chosen before round 1.

We define the *clean event* for arm  $x$  as

$$\mathcal{E}_x = \{|\mu_t(x) - \mu(x)| \leq r_t(x), \quad \forall t\}$$

Here, for convenience, we define  $\mu_t(x)$  to be 0 if arm  $x$  has not yet been played by the algorithm, so that in this case the clean event holds trivially. We are interested in the event  $\mathcal{E} = \bigcap_{x \in X} \mathcal{E}_x$ .

**Claim 5.9.**  $\Pr[\mathcal{E}] \geq 1 - \frac{1}{T^2}$ .

*Proof.* By Hoeffding Inequality,  $\Pr[\mathcal{E}_x] \geq 1 - \frac{1}{T^4}$  for each arm  $x \in X$ . However, one cannot immediately apply the Union Bound here because there may be too many arms.

Let  $X_0$  be the set of all arms that can possibly be activated by the algorithm. Note that  $X_0$  is finite; this is because the algorithm is deterministic, the time horizon  $T$  is fixed, and, as we assumed upfront, each realized reward can take only finitely many values. (This is the only place where we use that assumption.)

Let  $N$  be the total number of arms activated by the algorithm. Denote  $[T] := \{1, 2, \dots, T\}$ . Define arms  $y_j \in X_0, j \in [T]$ , as follows

$$y_j = \begin{cases} j\text{-th arm activated,} & \text{if } j \leq N \\ X_N, & \text{otherwise.} \end{cases}$$

Here  $N$  and  $y_j$ 's are random variables, the randomness coming from the reward realizations. Note that  $\{y_1, \dots, y_N\}$  is precisely the set of arms activated in a given execution of the algorithm.

We will prove that  $\mathcal{E}_{y_j}$  happens with high probability, for each  $j$ . This suffices, because the clean event holds trivially for all arms that are not activated.

Fix an arm  $x \in X_0$  and fix  $j \in [T]$ . Whether or not the event  $\{y_j = x\}$  holds is determined by the rewards of other arms. (Indeed, by the time arm  $x$  is selected by the algorithm, it is already determined whether  $x$  is the  $j$ -th arm activated!) Whereas whether or not the clean event  $\mathcal{E}_x$  holds is determined by the rewards of arm  $x$  alone. It follows that the events  $\{y_j = x\}$  and  $\mathcal{E}_x$  are independent. Thus:

$$\Pr[\mathcal{E}_{y_j} | y_j = x] = \Pr[\mathcal{E}_x | y_j = x] = \Pr[\mathcal{E}_x] \geq 1 - \frac{1}{T^4}$$

Now we can sum over all all  $x \in X_0$ :

$$\Pr[\mathcal{E}_{y_j}] = \sum_{x \in X_0} \Pr[y_j = x] \cdot \Pr[\mathcal{E}_{y_j} | x = y_j] \geq 1 - \frac{1}{T^4}$$

To complete the proof, we apply the Union bound over all  $j \in [T]$ :

$$\Pr[\mathcal{E}_{y_j}, j \in [T]] \geq 1 - \frac{1}{T^3}. \quad \square$$

We assume the clean event  $\mathcal{E}$  from here on.

### 5.3.3 Analysis: bad arms

Let us analyze the ‘‘bad arms’’: arms with low mean rewards. We establish two crucial properties: that active bad arms must be far apart in the metric space (Corollary 5.11), and that each ‘‘bad’’ arm cannot be played too often (Corollary 5.12).

*Notation.* As usual, let  $\Delta(x) = \mu^* - \mu(x)$  denote the ‘‘badness’’ of arm  $x$ . Let  $n(x) = n_{T+1}(x)$  be the total number of samples from arm  $x$ .

The following lemma encapsulates a crucial argument which connects the best arm and the arm played in a given round. In particular, we use the main trick from the analysis of UCB1 and the Lipschitz property.

**Lemma 5.10.**  $\Delta(x) \leq 3r_t(x)$  for each arm  $x$  and each round  $t$ .

*Proof.* Fix arm  $x$  and round  $t$ . Suppose  $x$  is played in this round. By the covering invariant, in this round the best arm  $x^*$  was covered by the confidence ball of some arm  $y$ . It follows that

$$\text{index}(x) \geq \text{index}(y) = \underbrace{\mu_t(y) + r_t(y)}_{\geq \mu(y)} + r_t(y) \geq \mu(x^*) = \mu^*$$

The last inequality holds because of the Lipschitz condition. On the other hand:

$$\text{index}(x) = \underbrace{\mu_t(x)}_{\leq \mu(x) + r_t(x)} + 2 \cdot r_t(x) \leq \mu(x) + 3 \cdot r_t(x)$$



Putting these two equations together:  $\Delta(x) := \mu^* - \mu(x) \leq 3 \cdot r_t(x)$ .

Now suppose arm  $x$  is not played in round  $t$ . If it has never been played before round  $t$ , then  $r_t(x) > 1$  and the lemma follows trivially. Else, letting  $s$  be the last time when  $x$  has been played before round  $t$ , we see that  $r_t(x) = r_s(x) \geq \Delta(x)/3$ .  $\square$

**Corollary 5.11.** *For any two active arms  $x, y$ , we have  $\mathcal{D}(x, y) > \frac{1}{3} \min(\Delta(x), \Delta(y))$ .*

*Proof.* W.l.o.g. assume that  $x$  has been activated before  $y$ . Let  $s$  be the time when  $y$  has been activated. Then  $\mathcal{D}(x, y) > r_s(x)$  by the activation rule. And  $r_s(x) \geq \Delta(x)/3$  by Lemma 5.10.  $\square$

**Corollary 5.12.** *For each arm  $x$ , we have  $n(x) \leq O(\log T) \Delta^{-2}(x)$ .*

*Proof.* Use Lemma 5.10 for  $t = T$ , and plug in the definition of the confidence radius.  $\square$

### 5.3.4 Analysis: regret

For  $r > 0$ , consider the set of arms whose badness is between  $r$  and  $2r$ :

$$X_r = \{x \in X : r \leq \Delta(x) < 2r\}.$$

Fix  $i \in \mathbb{N}$  and let  $Y_i = X_r$ , where  $r = 2^{-i}$ . By Corollary 5.11, for any two arms  $x, y \in Y_i$ , we have  $\mathcal{D}(x, y) > r/3$ . If we cover  $Y_i$  with subsets of diameter  $r/3$ , then arms  $x$  and  $y$  cannot lie in the same subset. Since one can cover  $Y_i$  with  $N_{r/3}(Y_i)$  such subsets, it follows that  $|Y_i| \leq N_{r/3}(Y_i)$ .

Using Corollary 5.12, we have:

$$R_i(T) := \sum_{x \in Y_i} \Delta(x) \cdot n_t(x) \leq \frac{O(\log T)}{\Delta(x)} \cdot N_{r/3}(Y_i) \leq \frac{O(\log T)}{r} \cdot N_{r/3}(Y_i).$$

Pick  $\delta > 0$ , and consider arms with  $\Delta(\cdot) \leq \delta$  separately from those with  $\Delta(\cdot) > \delta$ . Note that the total regret from the former cannot exceed  $\delta$  per round. Therefore:

$$\begin{aligned} R(T) &\leq \delta T + \sum_{i: r=2^{-i} > \delta} R_i(T) \\ &\leq \delta T + \sum_{i: r=2^{-i} > \delta} \frac{\Theta(\log T)}{r} N_{r/3}(Y_i) \\ &\leq \delta T + O(c \cdot \log T) \cdot \left(\frac{1}{\delta}\right)^{d+1}, \end{aligned} \tag{5.9}$$

where  $c$  is a constant and  $d$  is some number such that

$$N_{r/3}(X_r) \leq \frac{c}{r^d} \quad \forall r > 0.$$

The smallest (infimum) such  $d$  is called the *zooming dimension* with multiplier  $c$ .

By choosing  $\delta = \left(\frac{\log T}{T}\right)^{1/(d+2)}$ , we obtain

$$R(T) = O\left(T^{\frac{d+1}{d+2}} (c \log T)^{\frac{1}{d+2}}\right).$$

Note that we make this choice in the analysis only; the algorithm does not depend on the  $\delta$ .

**Theorem 5.13.** *Consider Lipschitz MAB problem with time horizon  $T$ . For any given problem instance and any  $c > 0$ , the zooming algorithm attains regret*

$$\mathbb{E}[R(T)] \leq O\left(T^{\frac{d+1}{d+2}} (c \log T)^{\frac{1}{d+2}}\right),$$

where  $d$  is the zooming dimension with multiplier  $c$ .

While the covering dimension is a property of the metric space, the zooming dimension is a property of the problem instance: it depends not only on the metric space, but on the rewards function  $\mu$ . In general, the zooming dimension is at most as large as the covering dimension,<sup>2</sup> but may be much smaller. This is because in the definition of the covering dimension one needs to cover all of  $X$ , whereas in the definition of the zooming dimension one only needs to cover set  $X_r$ .

While the regret bound in Theorem 5.13 is appealingly simple, a more precise regret bound is given in (5.9). Since the algorithm does not depend on  $\delta$ , this bound holds for all  $\delta > 0$ .

## 5.4 Remarks

The setting of Lipschitz bandits makes some idealized assumptions: that the Lipschitz condition holds exactly and that it holds with respect to a metric that is fully known to the algorithm. Of course, these assumptions do not necessarily hold in practice.

On the other hand, adaptive discretization is a general technique that may be of use even if the precise assumptions do not hold. To that end, it is worth noting that some of the assumptions in the analysis of the zooming algorithm can be relaxed:

- No need to assume triangle inequality.
- Lipschitz condition needs to hold only for pairs  $(x^*, y)$  (for this algorithm), or only in the vicinity of  $x^*$  (for another “adaptive discretization” algorithm with essentially the same guarantees).
- Zooming algorithm can be extended to settings in which the similarity between arms is given by a tree / taxonomy on arms (two arms in the same subtree have similar mean rewards), but the algorithm is only given the tree, rather than specific numbers that characterize similarity.

**Bibliographic notes.** Uniform discretization for continuum-armed bandits is from (Kleinberg, 2004); the extension to Lipschitz bandits is from (Kleinberg et al., 2008).

The zooming algorithm is from Kleinberg et al. (2008) (see Kleinberg et al., 2015, for the most recent version). A different algorithm that implements adaptive discretization, with very similar regret bounds, appeared in the follow-up paper (Bubeck et al., 2011).

The lower bound in Section 5.1.2 has been proved in (Kleinberg, 2004), in a much stronger formulation and a much more complicated proof. The present proof is unpublished.

For a discussion of the work on Lipschitz bandits and related problems, see “related work” section in (Kleinberg et al., 2015).

---

<sup>2</sup>More precisely: if  $d = \text{COV}_c$ , then the zooming dimension with multiplier  $3^d \cdot c$  is at most  $d$ .

## Chapter 6

# Full Feedback and Adversarial Rewards

In today lecture, we will shift our focus from problems with bandit feedback to problems with full feedbacks and adversarial rewards. In a full-feedback setting, in the end of each round, we observe rewards of not only the arm we chose to play, but of all other arms as well.

In each round  $t \in [T]$ :

1. Algorithm picks arm  $a \in [K]$ .
2. Algorithm incurs cost  $c_t(a)$  for the chosen arm.
3. Also, the costs of all arms,  $c_t(a) : a \in [K]$ , are revealed.

Multi-armed bandits with full feedback

A real-life example is the investment problem. Each morning, we choose a stock to invest. At the end of the day, we observe not only the price of our target stock but prices of all stocks. Based on this kind of “full“ feedback, we determine which stock to invest for the next day.

A motivating special case of “bandits with full feedback” can be framed as a question-answering problem with experts advice. Concretely, suppose we are assisted with a committee of experts to answer questions. In each round, a question arrives and each expert suggests an answer for it. In the stock example, a question would be “which stock to invest today?”; in binary prediction with experts advice, typical questions are “which label to assign to this example?”. After receiving the question, we (or our algorithms) listen to the experts and pick an answer to respond. We then observe the correct answer and costs (penalties) of all other answers. Our goal is to do approximately as well as the best expert.

Such a process can be described by the following protocol:

For each round  $t \in [T]$ :

1. Question  $x_t$  arrives.
2.  $K$  experts give answers  $z_{1,t}, \dots, z_{K,t}$ .
3. Algorithm picks expert  $i$ .
4. Correct answer  $y_t^*$  is observed.
5. Also, the costs for all submitted answers,  $c(z_{j,t}, y_t^*)$ , are revealed.

Because of this motivation, the bandit problem with full feedback is also called the *best-expert problem*. Since in the context of Q&A it is natural to talk about *costs* (as penalties for incorrect answers), full-feedback bandit problems are usually defined in terms of *costs* rather than *rewards*. We will follow this convention, too.

## 6.1 Warm-up: full feedback and i.i.d. rewards

When costs of all arms are realized, there is no need to explore. If we apply a naive strategy such as playing arm with the lowest average cost, one can prove that this strategy achieves an upper regret bound of  $O\left(\sqrt{T \log(KT)}\right)$  and a lower regret bound of  $\Omega(\sqrt{T})$ . The upper bound can be proved by a simple application of clean event/confidence radius technique that we have seen in previous lectures. The lower bound follows from the same argument as the bandit lower bound for 2 arms (it turns out we can just assume full feedback in that proof).

*Remark 6.1.* The  $\sqrt{T}$  is still present in the upper and lower bounds, even with full feedback. We will later see a simple special case with a lower bound  $\Omega(\min(T, \log K))$ . So the best lower bound we have for full feedback and i.i.d. rewards can be summarized as  $\Omega(\sqrt{T}, \min(T, \log K))$ .

## 6.2 Full feedback and adversarial rewards

What does “adversarial rewards” mean? It does not necessarily mean that there is a real adversary that tries hard to make our algorithm fail. This term denotes situations when the costs change over time. For example, at a news website, everyday an algorithm decides which news to show to the users. At some point, since people are mostly interested in the presidential election, the cost of showing election may be low. However, in the future, people may be attracted to other events (e.g. the Olympics), the cost of the election will get higher and the cost of the more interesting events will become lower.

In general, the problem can be casted as playing a game with the adversary:

Each round  $t \in [T]$ :

1. Adversary chooses costs  $c_t(a)$  for each arm  $a \in [K]$ .
2. Algorithm picks arm  $a_t$ .
3. Algorithm receives cost  $c_t(a_t)$  for the chosen arm.
4. Algorithm observes  $c_t(a)$  for all arms  $a$ .

Best-expert problem with adversarial rewards.

An adversary is called *oblivious* if the chosen costs at each round  $t$  do not depend on the algorithm’s choices before round  $t$ . Otherwise, it is called *adaptive*. Also, an adversary is called *randomized* if in each round it chooses distribution over the cost vectors  $(c_t(a) : a \in [K])$ , and then draws the cost vector independently from this distribution.

The total cost of each arm  $a$  is defined as  $\text{cost}(a) = \sum_{t=1}^T c_t(a)$ . Intuitively, the “best arm” is an arm with a lowest cost. However, defining the “best arm” and “regret” precisely is a little tricky, depending on the “type” of an adversary:

- *Deterministic-oblivious adversary*: equivalently, the entire “cost table”  $c_t(a)$  for all arms and all rounds is chosen before the first round. Then, naturally, the best arm is defined as  $a^* \in \operatorname{argmin}_{a \in [K]} \text{cost}(a)$ , and regret is defined as

$$R(T) = \text{cost}(ALG) - \min_{a \in [K]} \text{cost}(a), \tag{6.1}$$

where  $\text{cost}(ALG)$  denotes the total cost incurred by the algorithm.

One drawback of considering this adversary is that it does not model i.i.d. costs (because there is no randomness in costs), whereas we'd like to think of i.i.d. rewards as a simple special case of adversarial rewards.

- *Randomized-oblivious adversary*: equivalently, the adversary fixes a distribution  $\mathcal{D}$  over the cost tables before round one, and then draws a cost table from this distribution. Thus,  $\text{cost}(a)$ , the total cost of each arm  $a$ , is a random variable whose distribution is specified by  $\mathcal{D}$ . Accordingly, there are two natural ways to define the “best arm”:
  - $\text{argmin}_a \text{cost}(a)$ : this is the best arm *in hindsight* – after all costs have been observed. It is a natural notion of the best arm if we start from the deterministic-oblivious adversary.
  - $\text{argmin}_a \mathbb{E}[\text{cost}(a)]$ : this is the best arm *in foresight* – and arm that you'd pick before round 1 if you only know the distribution  $\mathcal{D}$  and you need to pick one arm to play in all rounds. This is a natural notion of “best arm” if we start from i.i.d. costs.

Accordingly, there are two natural notions of regret: w.r.t. the best-arm-in-hindsight (as in (6.1)), and w.r.t. best-arm-in-foresight:

$$R(T) = \text{cost}(ALG) - \min_{a \in [K]} \mathbb{E}[\text{cost}(a)], \quad (6.2)$$

(We used the second notion for bandits with i.i.d. rewards/costs.) We will refer to them as *hindsight regret* and *foresight regret*, respectively.<sup>1</sup>

Foresight regret cannot exceed hindsight regret (because the best-arm-in-foresight is a weaker benchmark), and sometimes it allows for much stronger positive results. In particular, the  $\log(T)$  regret bounds for i.i.d. rewards/costs do not extend to foresight regret.

- *Adaptive adversary* models scenarios when algorithm's actions may alter the environment that the algorithm operates in. For example:
  - an algorithm that adjusts the layout and text formatting of a website may cause users to permanently change their behavior, e.g., users may gradually get used to a new design, and get dissatisfied with the old one.
  - Consider a bandit algorithm that selects news articles to show to users. A change in how the articles are selected may attract some users and repel some others, or perhaps cause the users to alter their reading preferences.
  - if a dynamic pricing algorithm offers a discount on a new product, it may cause many people to buy this product and (eventually) grow to like it and spread the good word. Then more people would be willing to buy this product at full price.
  - if a bandit algorithm adjusts the parameters of a repeated auction (e.g., a reserve price), auction participants may adjust their behavior over time, as they become more familiar with the algorithm.

Usually adaptive adversary is used to model non-malicious environment that adjusts itself in response to the algorithm's actions, rather than an actual adversary that strives to hurt us.

An adaptive adversary is assumed to be randomized by default, because the adversary can adapt the costs to the algorithm's choice of arms in the past, and the algorithm is usually randomized. Thus, the distinction between foresight and hindsight regret comes into play.

More importantly, which arm is best may depend on the algorithm's actions. For example, an algorithm that always chooses arm 1 may see that arm 2 is consistently much better, whereas *if the algorithm always played arm 2*, arm 1 may have been better. In this course we use a simple notion: best-in-hindsight-arm according to the costs as they are actually observed by the algorithm; we call it the *best observed arm*.

---

<sup>1</sup>In the literature, the hindsight regret is usually referred to as *regret*, while the foresight regret is referred to as *weak regret* or *pseudo-regret*.

There are three good reasons to consider this notion. First, this notion *is* meaningful in some scenarios (but not in some others). Second, results for adaptive adversary and best observed arm often follow from the analysis of oblivious adversary with very little extra work. Third, best-realized-arm analysis of an adaptive adversary may be used as a tool to prove a statement about oblivious adversaries (e.g., next lecture).

### 6.3 Majority vote algorithm

We consider *binary prediction with experts advice*, a special case of “Q&A with expert advice”, where the expert answers  $z_{i,t}$  are binary. For example: is this image a face or not? Is it going to rain today or not?

Let us start with a simple example when there exists a *perfect expert* who never makes a mistake. We propose the following algorithm, called the *majority vote algorithm*:

In round  $t$ , pick the action chosen by the majority of the experts.

**Theorem 6.2.** *Consider binary prediction with experts advice. Assuming a perfect expert, the majority vote algorithm makes at most  $\log_2 K$  mistakes, where  $K$  is the number of experts.*

*Proof.* Let  $S_t$  be the set of experts who make no mistakes up to the beginning of round  $t$ .

Let  $W_t = |S_t|$ . Since the perfect expert is always in  $S_t$ ,  $W_t \geq 1$ .

If the algorithm makes a mistake at round  $t$ , then  $W_{t+1} \leq W_t/2$  because the majority is wrong and thus excluded from  $S_t$ . Since  $W_t \geq 1$ , the algorithm cannot make more than  $\log_2 K$  mistakes.  $\square$

While this proof is very simple, it introduces a general technique that will be essential in the subsequent proofs:

- Define a quantity  $W_t$  which measures the total remaining amount of “credibility” of the the experts. Make sure  $W_t$  decreases by definition as experts make mistakes (incur costs).
- In the analysis, connect  $W_t$  with the behavior of the algorithm. First, prove that  $W_t$  decreases by a constant factor whenever the algorithm makes mistake (incurs a cost). Second, derive a lower bound on  $W_T$  from the existence of a “good expert”.

One can prove a simple lower bound for this problem:

**Theorem 6.3.** *Consider binary prediction with experts advice. Assume a perfect expert. For any algorithm, there is a problem instance such that the algorithm makes at least  $\Omega(\min(T, \log K))$  mistakes, where  $T$  is the time horizon and  $K$  is the number of experts.*

*Proof Sketch.* Assume  $\log_2 K \in \mathbb{N}$ . For the first  $\log_2 K$  rounds, consider all possible sequences of predictions, create one expert for each prediction. Pick the perfect expert at random.  $\square$

### 6.4 Binary Prediction with Expert Advice: Weighted Majority Algorithm

Previously, we introduced the best-expert problem, and we proved a  $O(\ln K)$  mistake bound for the majority vote algorithm when a *perfect expert* exists, i.e., there is an expert that never makes mistakes. Now let us turn to the more realistic case where there is no perfect expert among the committee.

We extend the majority vote algorithm with a *confidence weight*. At each round, we maintain a weight  $w_i$  for each expert  $i$ , and we choose the prediction that has the highest total weights. After observing the feedback, we decay the weights of incorrect experts with a factor of  $(1 - \epsilon)$ . This algorithm is called *Weighted Majority Algorithm (WMA)*.

To analyze the algorithm, we first introduce some notations.

- Let  $\text{cost}(a)$  be the total number of mistakes for expert  $a$ .
- Let  $a^*$  be the best expert: an expert  $a$  with the smallest  $\text{cost}(a)$ . Let  $\text{cost}^* = \text{cost}(a^*)$ .
- Let  $W_t = \sum_{a=1}^K w_t(a)$ , where  $w_t(a)$  is the weight of expert  $a$  before round  $t$ .

**parameter:**  $\epsilon \in [0, 1]$

- 1 Initialize the weights  $w_i = 1$  for all experts.
- 2 For each round  $t$ :
- 3     Make predictions using weighted majority vote based on  $w$ .
- 4     For each expert  $i$ :
- 5         If the  $i$ -th expert's prediction is correct,  $w_i$  stays the same.
- 6         Otherwise,  $w_i \leftarrow w_i(1 - \epsilon)$ .

**Algorithm 5:** Weighted Majority Algorithm

- Let  $S_t$  be the set of experts that made incorrect prediction at round  $t$ .

We will also use the following fact about logarithm:

**Fact 6.4.** For any  $x \in (0, 1)$ ,  $\ln(1 - x) < -x$ .

From the algorithm, we can easily see that  $W_1 = K$  and  $W_{T+1} > w_t(a^*) = (1 - \epsilon)^{\text{cost}^*}$ . Therefore, we have

$$\frac{W_{T+1}}{W_1} > \frac{(1 - \epsilon)^{\text{cost}^*}}{K}. \quad (6.3)$$

Since the weights are non-increasing, we must have

$$W_{t+1} \leq W_t \quad (6.4)$$

If the algorithm makes mistake at round  $t$ , then

$$\begin{aligned} W_{t+1} &= \sum_{a=1}^K w_{t+1}(a) \\ &= \sum_{a \in S_t} (1 - \epsilon)w_t(a) + \sum_{a \notin S_t} w_t(a) \\ &= W_t - \epsilon \sum_{a \in S_t} w_t(a). \end{aligned}$$

Since we are using weighted majority vote, the incorrect prediction must have the majority vote:

$$\sum_{a \in S_t} w_t(a) \geq \frac{1}{2}W_t.$$

Therefore, if the algorithm makes mistake at round  $t$ , we have

$$W_{t+1} \leq (1 - \frac{\epsilon}{2})W_t.$$

Combining with (6.3) and (6.4), we get

$$\frac{(1 - \epsilon)^{\text{cost}^*}}{K} < \frac{W_{T+1}}{W_1} = \prod_{t=1}^T \frac{W_{t+1}}{W_t} \leq (1 - \frac{\epsilon}{2})^M,$$

where  $M$  is the number of mistakes. Taking logarithm of both sides, we get

$$\text{cost}^* \cdot \ln(1 - \epsilon) - \ln K < M \cdot \ln(1 - \frac{\epsilon}{2}) < M \cdot (-\frac{\epsilon}{2}),$$

where the last inequality follows from Fact 6.4. Rearranging the terms, we get

$$M < \text{cost}^* \cdot \frac{2}{\epsilon} \ln(\frac{1}{1-\epsilon}) + \frac{2}{\epsilon} \ln K < \frac{2}{1-\epsilon} \cdot \text{cost}^* + \frac{2}{\epsilon} \cdot \ln K.$$

Where the last step follows from Fact 6.4 with  $x = \frac{\epsilon}{1-\epsilon}$ . To summarize, we have proved the following theorem.

**Theorem 6.5.** *The number of mistakes made by WMA with parameter  $\epsilon \in (0, 1)$  is at most*

$$\frac{2}{1-\epsilon} \cdot \text{cost}^* + \frac{2}{\epsilon} \cdot \ln K$$

*Remark 6.6.* This bound is non-trivial if  $\text{cost}^*$  is small, but it is not strong enough to give us sublinear regret.

## 6.5 Hedge Algorithm

Deterministic algorithms are not ideal, because they can be easily “fooled” by an adversary, as demonstrated by the following theorem.

*Notation.* Let  $\text{cost}(a) = \sum_{t=1}^T c_t(a)$  be the total cost expert  $a$ . Let  $a^*$  be the best expert: an expert  $a$  with the smallest  $\text{cost}(a)$ . Let  $\text{cost}^* = \text{cost}(a^*)$ . Let  $K$  be the number of arms and  $T$  be the time horizon; denote the set of arms as  $[K] = \{1, 2, \dots, K\}$ .

**Theorem 6.7.** *Consider the best-expert problem with two arms and 0-1 costs. Any deterministic algorithm ALG has regret at least  $T/2$  for some deterministic-oblivious adversary.*

*Proof.* Consider the following family of problem instances: at each round  $t$ , exactly one arm has cost 0, and the other arm cost 1 (denote it  $b_t$ ). Note that  $\text{cost}(1) + \text{cost}(2) = T$ , so  $\text{cost}^* \leq T/2$ .

Let us construct the problem instance by induction on  $t$  so as to ensure that  $a_t \neq b_t$ . For round 1, ALG picks some fixed arm  $a_1$ , so we define  $b_1 = a_1$ . Suppose we have defined  $b_1, \dots, b_{t-1}$  for some  $t$ . Since ALG is deterministic, it picks some fixed arm  $a_t$  in round  $t$ ; define  $b_t = a_t$ . Thus, we have constructed a problem instance with  $\text{cost}(\text{ALG}) = T$ , which implies regret at least  $T/2$ .  $\square$

We define a randomized algorithm for the best-expert problem, called Hedge. This algorithm maintains a weight  $w_t(a)$  for each arm (expert)  $a$ , with the same update rule as in WMA (generalized beyond 0-1 costs in an obvious way). At each round, the algorithm chooses an arm with probability proportional to the weights. The complete algorithm is shown below:

<p><b>parameter:</b> <math>\epsilon \in (0, \frac{1}{2})</math></p> <ol style="list-style-type: none"> <li>1 Initialize the weights as <math>w_1(a) = 1</math> for each arm <math>a</math>.</li> <li>2 For each round <math>t</math>:</li> <li>3     Let <math>p_t(a) = \frac{w_t(a)}{\sum_{a'=1}^K w_t(a')}</math>.</li> <li>4     Sample an arm <math>a_t</math> from distribution <math>p_t(\cdot)</math>.</li> <li>5     Observe cost <math>c_t(a)</math> for each arm <math>a</math>.</li> <li>6     For each arm <math>a</math>, update its weight <math>w_{t+1}(a) = w_t(a) \cdot (1 - \epsilon)^{c_t(a)}</math>.</li> </ol>
---

**Algorithm 6:** Hedge

let us analyze the regret of Hedge. Following the last section, we will use  $W_t = \sum_{a=1}^K w_t(a)$  for the total weight of all arms at round  $t$ .

**Step 1: easy observations.** The weight of each arm after the last round is

$$w_{T+1}(a) = w_1(a) \prod_{t=1}^T (1 - \epsilon)^{c_t(a)} = (1 - \epsilon)^{\text{cost}(a)}.$$

Hence, the total weight of last round satisfies

$$W_{T+1} > w_{T+1}(a^*) = (1 - \epsilon)^{\text{cost}^*}. \quad (6.5)$$

From the algorithm, we know that the total initial weight is  $W_1 = K$ .

**Step 2: multiplicative decrease in  $W_t$ .** We will use the following fact about exponents:



**Fact 6.8.** Fix  $\epsilon \in (0, 1)$ . There exist  $\alpha, \beta \geq 0$  such that

$$(1 - \epsilon)^x < 1 - \alpha x + \beta x^2 \quad \text{for all } x > 0. \quad (6.6)$$

In particular, this holds for (i)  $(\alpha, \beta) = (\epsilon, 0)$  and (ii)  $\alpha = \ln(\frac{1}{1-\epsilon})$  and  $\beta = \alpha^2$ .

Using this fact with  $x = c_t(a)$ , we will get the following:

$$\begin{aligned} \frac{W_{t+1}}{W_t} &= \sum_{a \in [K]} (1 - \epsilon)^{c_t(a)} \cdot \frac{w_t(a)}{W_t} \\ &< \sum_{a \in [K]} (1 - \alpha c_t(a) + \beta c_t(a)^2) \cdot p_t(a) \\ &= \sum_{a \in [K]} p_t(a) - \alpha \sum_{a \in [K]} p_t(a) c_t(a) + \beta \sum_{a \in [K]} p_t(a) c_t(a)^2 \\ &= 1 - \alpha F_t + \beta G_t, \end{aligned} \quad (6.7)$$

where

$$\begin{aligned} F_t &= \sum_a p_t(a) \cdot c_t(a) = \mathbb{E}[c_t(a_t) \mid \vec{w}_t] \\ G_t &= \sum_a p_t(a) \cdot c_t(a)^2 = \mathbb{E}[c_t(a_t)^2 \mid \vec{w}_t]. \end{aligned}$$

Here  $\vec{w}_t = (w_t(a) : a \in [K])$  is the vector of weights at round  $t$ . Notice that the total expected cost of the algorithm is  $\mathbb{E}[\text{cost}(ALG)] = \sum_t \mathbb{E}[F_t]$ .

**Step 3: a naive attempt.** Using the (6.7), we can obtain:

$$\frac{(1 - \epsilon)^{\text{cost}^*}}{K} \leq \frac{W_{T+1}}{W_1} = \prod_{t=1}^T \frac{W_{t+1}}{W_t} < \prod_{t=1}^T (1 - \alpha F_t + \beta G_t).$$

However, it is not clear how to connect the right-hand side to  $\sum_t F_t$  so as to argue about the total cost of the algorithm.

**Step 4: the telescoping product.** Taking a logarithm on both sides of (6.7) and using Fact (6.4), we get

$$\ln \frac{W_{t+1}}{W_t} < \ln(1 - \alpha F_t + \beta G_t) < -\alpha F_t + \beta G_t.$$

Inverting the signs and summing over  $t$  on both sides, we get

$$\begin{aligned} \sum_{t=1}^T (\alpha F_t - \beta G_t) &< - \sum_{t=1}^T \ln \frac{W_{t+1}}{W_t} \\ &= - \ln \prod_{t=1}^T \frac{W_{t+1}}{W_t} \\ &= - \ln \frac{W_{T+1}}{W_1} \\ &= \ln W_1 - \ln W_{T+1} \\ &< \ln K - \ln(1 - \epsilon) \cdot \text{cost}^*, \end{aligned}$$

where we used (6.5) in the last step. Taking expectation on both sides, we obtain:

**Lemma 6.9.** For  $\alpha, \beta$  from Fact 6.8, we have:

$$\alpha \mathbb{E}[\text{cost}(ALG)] < \beta \sum_{t=1}^T \mathbb{E}[G_t] + \ln K - \ln(1 - \epsilon) \mathbb{E}[\text{cost}^*].$$

In what follows, we use this lemma in two different ways.

**Step 5: main theorem.** Using Lemma Lemma 6.9 with  $\alpha = \epsilon$  and  $\beta = 0$ , we obtain:

$$\begin{aligned} \mathbb{E}[\text{cost}(ALG)] &< \frac{\ln K}{\epsilon} + \underbrace{\frac{1}{\epsilon} \ln\left(\frac{1}{1-\epsilon}\right)}_{\leq 1 + 2\epsilon \text{ if } \epsilon \in (0, \frac{1}{2})} \mathbb{E}[\text{cost}^*]. \\ \mathbb{E}[\text{cost}(ALG) - \text{cost}^*] &< \frac{\ln K}{\epsilon} + 2\epsilon \mathbb{E}[\text{cost}^*]. \end{aligned}$$

This yields the following theorem:

**Theorem 6.10.** Assume we have  $\text{cost}^* \leq U$  for some number  $U \leq T$  known to the algorithm. Then the regret of Hedge with parameter  $\epsilon = \sqrt{\ln K / (2U)}$  is

$$\mathbb{E}[\text{cost}(ALG) - \text{cost}^*] < 2\sqrt{2} \cdot \sqrt{U \ln K}.$$

**Step 6: unbounded costs.** Next, we consider the case where the costs can be unbounded, but we have an upper bound on  $\mathbb{E}[G_t]$ . We use Lemma 6.9 with  $\alpha = \ln(\frac{1}{1-\epsilon})$  and  $\beta = \alpha^2$  to obtain:

$$\alpha \mathbb{E}[\text{cost}(ALG)] < \alpha^2 \sum_{t=1}^T \mathbb{E}[G_t] + \ln K + \alpha \mathbb{E}[\text{cost}^*].$$

Dividing both sides by  $\alpha$  and moving terms around, we get

$$\mathbb{E}[\text{cost}(ALG) - \text{cost}^*] < \frac{\ln K}{\alpha} + \alpha \sum_{t=1}^T \mathbb{E}[G_t] < \frac{\ln K}{\epsilon} + 3\epsilon \sum_{t=1}^T \mathbb{E}[G_t],$$

where the last step uses the fact that  $\epsilon < \alpha < 3\epsilon$  for  $\epsilon \in (0, \frac{1}{2})$ . Thus:

**Lemma 6.11.** Assume we have  $\sum_{t=1}^T \mathbb{E}[G_t] \leq U$  for some number  $U$  known to the algorithm. Then Hedge with parameter  $\epsilon = \sqrt{\ln K / (3U)}$  as regret

$$\mathbb{E}[\text{cost}(ALG) - \text{cost}^*] < 2\sqrt{3} \cdot \sqrt{U \ln K}.$$

**Step 7: unbounded costs with small expectation and variance.** We can use Lemma 6.11 to derive a regret bound when costs may be unbounded, but their expectation and variance are small. Specifically, consider an *randomized-oblivious* adversary, and assume that the costs are independent across rounds. Further, assume an upper bound:

$$\mathbb{E}[c_t(a)] \leq \mu \text{ and } \text{Var}(c_t(a)) \leq \sigma^2 \text{ for all rounds } t \text{ and all arms } a. \quad (6.8)$$

Then for each round  $t$  we have:

$$\begin{aligned} \mathbb{E}[c_t(a)^2] &= \text{Var}(c_t(a)) + \mathbb{E}[c_t(a)]^2 \leq \sigma^2 + \mu^2. \\ \mathbb{E}[G_t] &= \sum_{a \in [K]} p_t(a) \mathbb{E}[c_t(a)^2] \leq \sum_{a \in [K]} p_t(a) (\mu^2 + \sigma^2) = \mu^2 + \sigma^2. \end{aligned}$$

Summing this upper bound over all rounds  $t$ , and plugging it into Lemma 6.11, we obtain:

**Theorem 6.12.** Consider the best-experts problem with a randomized-oblivious adversary. Assume the costs are independent across rounds. Assume upper bound (6.8) for some  $\mu$  and  $\sigma$  known to the algorithm. Then Hedge with parameter  $\epsilon = \sqrt{\ln K / (3T(\mu^2 + \sigma^2))}$  has regret

$$\mathbb{E}[\text{cost}(ALG) - \text{cost}^*] < 2\sqrt{3} \cdot \sqrt{T(\mu^2 + \sigma^2) \ln K}.$$

*Remark 6.13.* In the next lecture, we will use Lemma 6.11 to analyze a bandit algorithm.

## **6.6 Bibliographic remarks**

This material is presented in various courses and books on online learning, e.g. Cesa-Bianchi and Lugosi (2006) and Hazan (2015). This lecture follows a lecture plan from (Kleinberg, 2007, Week 1), but presents the analysis of Hedge a little differently (so as to make it immediately applicable to the analysis of EXP3/EXP4 in the next lecture).



## Chapter 7

# Adversarial Bandits

We study *adversarial bandits*: bandits with adversarially chosen costs. On the scale from “optimistic” to “pessimistic”, IID assumption is super-optimistic, and adversarial is super-pessimistic. We focus on deterministic, oblivious adversary: that is, the costs for all arms and all rounds are chosen in advance.

We will proceed via a reduction to the full-feedback problem studied in the previous lecture. Further, we will actually solve adversarial bandits in a more general version that explicitly includes expert advice.

**Setup and notation.** There are  $K$  actions and a fixed time horizon  $T$ . Each action  $a$  has an associated cost  $c_t(a)$  at round  $t$ . The total cost of each expert is the sum over all rounds:  $\text{cost}(a) = \sum_{t=1}^T c_t(a)$ . Regret is defined as the difference between the total cost of the algorithm and the minimum total cost of an action:

$$R(T) = \text{cost}(\text{ALG}) - \min_{\text{actions } a} \text{cost}(a). \quad (7.1)$$

(We do not take expectations in this definition; instead, we upper-bound the expected regret.)

**Results.** We will design an algorithm with regret bound

$$\mathbb{E}[R(T)] \leq O\left(\sqrt{KT \log K}\right)$$

Curiously, this upper bound not only matches the result for IID bandits, but in fact improves it a little bit, replacing the  $\log(T)$  term with  $\log(K)$ .

Recall that we had a lower bound  $\Omega(\sqrt{KT})$  (which holds even for IID bandits). Note the  $\sqrt{\log K}$  gap between the upper and lower bound. It turns out the lower bound is in fact the right one: there is another algorithm with regret  $O(\sqrt{KT})$ .

### 7.1 Recap: best-expert problem

Last lecture, we studied bandits with *full feedback* (cost of each arm is revealed after every round) and adversarially chosen costs. A common interpretation of this problem is that each action corresponds to an “expert” that gives advice or makes predictions, and in each round the algorithm needs to choose which expert to follow. Hence, this problem is also known as the *best-expert problem*.

To facilitate exposition in this lecture, when we talk about the full-feedback problem, we will refer to actions as *experts*  $x \in X$ , where  $X$  is the set of all experts. The number of experts is  $N = |X|$ . Regret is defined as in (7.1).

We studied an algorithm for the best-expert problem called *Hedge*. In each round, this algorithm computes a distribution over experts, denoted  $p_t$ , and samples an expert from this distribution. We will use the following regret bounds proved in last lecture:

**Theorem 7.1** (Hedge). *For the best-expert problem with adaptive adversary, Hedge satisfies*

$$\mathbb{E}[R(T)] \leq 2\sqrt{3} \cdot \sqrt{UT \log N},$$

where  $U$  is a number known to the algorithm such that

- (a)  $c_t(x) \leq U$  for all experts  $x$  and all rounds  $t$ , or  
 (b)  $\mathbb{E}[G_t] \leq U$  for all rounds  $t$ , where  $G_t = \sum_{x \in X} p_t(x) c_t^2(x)$ .

This regret bound is essentially optimal. Specifically, we have the following lower bound:

**Theorem 7.2** (lower bound). *Consider the best-expert problem with randomized, oblivious adversary and 0-1 costs. For each  $T, N$  there is a problem instance with  $N$  experts and time horizon  $T$  such that every algorithm suffers regret*

$$\mathbb{E}[R(T)] \geq \Omega\left(\sqrt{T \log N}\right). \quad (7.2)$$

The problem instance is very simple: each  $c_t(x)$  is chosen independently and uniformly at random from  $\{0, 1\}$ . It is crucial for this theorem that the benchmark in regret is the “hindsight” benchmark  $\min_x \text{cost}(x)$  rather than the “foresight” benchmark  $\min_x \mathbb{E}[\text{cost}(x)]$ . (Obviously, with respect to the latter benchmark any algorithm will have 0 regret for this problem instance.)

The theorem also implies that for each algorithm, there exists a deterministic problem instance with 0-1 costs such that this algorithm suffers regret (7.2) on this problem instance.

## 7.2 Reduction: from bandit feedback to full feedback

Our algorithm will be a reduction from bandit feedback to full feedback. Specifically, we take Hedge, an algorithm for the full-feedback setting, and use it as a subroutine.

The reduction proceeds as follows. For each arm, we create an expert which always recommends this arm. We use Hedge with this set of experts. In each round, we use the expert chosen by Hedge to pick an arm, and define “fake costs” on all experts in order to provide Hedge with valid inputs.

- 1 **Given:** set  $X$  of experts.
- 2 In each round  $t$ ,
  1. Call Hedge, receive the probability distribution  $p_t$  over  $X$ .
  2. Draw an expert  $x_t$  independently from  $p_t$ .
  3. Use  $x_t$  to pick arm  $a_t$  (TBD).
  4. Observe the cost  $c_t(a_t)$  of the chosen arm.
  5. Define “fake costs”  $\hat{c}_t(x)$  for all experts  $x \in X$  (TBD).
  6. Return the “fake costs” to Hedge.

**Algorithm 7:** Reduction from bandit feedback to full feedback

The details of *how* to define  $a_t$  using  $x_t$ , and *how* to define fake costs, will be provided later.

## 7.3 Adversarial bandits with expert advice

The reduction defined above suggests a more general problem: what if experts can make arbitrary predictions? This problem is called *bandits with expert advice*. We will in fact solve this problem rather than the original adversarial bandits problem. We do it for three reasons: because it is a very interesting generalization, because we can solve it with very little extra work, and because it separating experts from actions makes the solution clearer.

Formally, the problem of bandits with expert advice is defined as follows. There are  $K$  arms and  $N$  experts, and a fixed time horizon  $T$ . In each round  $t \in [T]$ ,

- adversary picks cost  $c_t(a)$  for each arm  $a$ ,
- each expert  $x$  recommends an arm  $a_{t,x}$ ,

- algorithm picks arm  $a_t$  and receive the corresponding cost  $c_t(a_t)$ .

The total cost of each expert is defined as  $\text{cost}(x) = \sum_{t \in [T]} c_t(a_{t,x})$ . The goal is to minimize regret w.r.t. best expert (rather than w.r.t. best action):

$$R(T) = \text{cost}(\text{ALG}) - \min_{\text{experts } x} \text{cost}(x).$$

We focus on deterministic, oblivious adversary: the costs for all arms and all rounds are selected in advance. Further, we assume that the expert recommendations  $a_{t,x}$  are also chosen in advance; in other words, experts cannot learn over time.

We will use the reduction in Algorithm 7 to solve this problem. In fact, we will *really* use the same reduction: the pseudocode applies as is! Our algorithm will have regret

$$\mathbb{E}[R(T)] \leq O\left(\sqrt{KT \log N}\right).$$

Note the logarithmic dependence on  $N$  – this regret bound allows to handle *lots* of experts.

In fact, there is a matching lower bound, for any given  $K, T, N$ . It can be proved by a simple reduction to the basic  $\Omega(\sqrt{KT})$  lower bound for bandits (and we may see it on HW2).

## 7.4 Preliminary analysis: unbiased estimates

The cost of each expert  $x$  at time step  $t$  is a function of the arm  $a_{t,x}$  they recommend at that time step. For brevity, denote this as  $c_t(x) = c_t(a_{t,x})$ . We will also define “fake costs”, denoted  $\hat{c}_t(x)$ , which are fed to Hedge. The regret bound for Hedge will be in terms of these fake costs:

$$\widehat{R}_{\text{Hedge}}(T) = \widehat{\text{cost}}(\text{Hedge}) - \min_x \widehat{\text{cost}}(x).$$

We want these fake costs to be unbiased estimates of the true costs. This is because we will need to convert a bound on the “fake regret”  $\widehat{R}_{\text{Hedge}}(T)$  into a statement about the true costs accumulated by Hedge. Formally, we will ensure that

$$\mathbb{E}[\hat{c}_t(x) \mid \vec{p}_t] = c_t(x) \quad \text{for all experts } x, \quad (7.3)$$

where  $\vec{p}_t = (p_t(x) : \text{all experts } x)$ . We use it to show:

**Claim 7.3.** *Assuming (7.3),  $\mathbb{E}[R_{\text{Hedge}}(T)] \leq \mathbb{E}[\widehat{R}_{\text{Hedge}}(T)]$ .*

*Proof.* First, we connect true cost of Hedge with its fake cost: we prove that

$$\mathbb{E}[\widehat{\text{cost}}(\text{Hedge})] = \mathbb{E}[\text{cost}(\text{Hedge})]. \quad (7.4)$$

Indeed,

$$\begin{aligned} \mathbb{E}[\hat{c}_t(x_t) \mid \vec{p}_t] &= \mathbb{E}\left[\sum_x p_t(x) \hat{c}_t(x) \mid \vec{p}_t\right] && \text{(focus on round } t) \\ &= \sum_x p_t(x) \mathbb{E}[\hat{c}_t(x) \mid \vec{p}_t] && \text{(take } \vec{p}_t \text{ out of conditioning)} \\ &= \sum_x p_t(x) c_t(x) && \text{(use (7.3))} \\ &= \mathbb{E}[c_t(x_t) \mid \vec{p}_t]. \end{aligned}$$

Taking expectation of both sides implies that  $\mathbb{E}[\hat{c}_t(x_t)] = \mathbb{E}[c_t(x_t)]$ . Summing over all rounds, we obtain (7.4).

Second, we deal with the benchmark:

$$\mathbb{E}[\min_x \widehat{\text{cost}}(x)] \leq \min_x \mathbb{E}[\widehat{\text{cost}}(x)] = \min_x \mathbb{E}[\text{cost}(x)] = \min_x \text{cost}(x).$$

The first equality holds because by (7.3), and the second equality holds because true costs are deterministic. Combining this with (7.4) implies the claim.  $\square$

Note that we used the “full power” of assumption (7.3): a weaker assumption  $\mathbb{E}[\hat{c}_t(x)] = \mathbb{E}[c_t(x)]$  would not have sufficed to argue about true vs. fake costs of Hedge.

## 7.5 Algorithm Exp4 and crude analysis

Let us complete the specification of Algorithm 7. To define fake costs, we will use a standard trick in statistics called Inverse Propensity Score (IPS). Whatever the way arm  $a_t$  is chosen in each round  $t$  given the probability distribution  $\vec{p}_t$  returned by Hedge, this defines distribution  $q_t$  over arms  $a$ :

$$q_t(a) = \Pr[a_t = a \mid \vec{p}_t].$$

Using these probabilities, we define the fake costs on each arm as follows:

$$\hat{c}_t(a) = \begin{cases} \frac{c_t(a_t)}{q_t(a_t)} & a_t = a \\ 0 & \text{otherwise.} \end{cases}$$

This step is well-defined as long as the algorithm can compute probability  $q_t(a_t)$ , because the algorithm knows the true cost  $c_t(a_t)$  for the chosen arm  $a_t$ . The fake cost on each expert  $x$  is defined as the fake cost of the arm chosen by this expert:  $\hat{c}_t(x) = \hat{c}_t(a_{t,x})$ .

This is indeed an unbiased estimator for true costs:

**Claim 7.4.** (7.3) holds if  $q_t(a) > 0$  for all arms  $a$ .

*Proof.* Let us argue about each arm  $a$  separately:

$$\mathbb{E}[\hat{c}_t(a) \mid \vec{p}_t] = q_t(a) \cdot \frac{c_t(a_t)}{q_t(a_t)} + \Pr[a_t \neq a] \cdot 0 = c_t(a).$$

Now, for a given expert  $x$  plug in arm  $a = a_{t,x}$ , its choice in round  $t$ . □

Thus, we need to ensure that  $q_t(a) > 0$  for all arms  $a$ , i.e., that each arm is chosen with a strictly positive probability, regardless of which expert  $x_t$  is chosen by Hedge. On the other hand, we'd like to follow expert  $x_t$  with high probability so as to ensure low cost in this round.

Therefore, we complete the specification of our algorithm as follows:

- in each round  $t$ , with some low probability  $\gamma \in (0, \frac{1}{2})$ , we pick an arm independently and uniformly at random. Otherwise, follow the advice of expert  $x_t$ .

This algorithm is known as Exp4. It is parameterized by  $\gamma \in (0, \frac{1}{2})$ .

Note that  $q_t(a) \geq \gamma/K > 0$  for each arm  $a$ , as needed. Combining this with Claim 7.4 and Claim 7.3, we obtain  $\mathbb{E}[R_{\text{Hedge}}(T)] \leq \mathbb{E}[\widehat{R}_{\text{Hedge}}(T)]$ .

In each round, our algorithm accumulates cost at most 1 from the low-probability exploration, and cost  $c_t(x_t)$  from the chosen expert  $x_t$ . So the expected cost in this round is at most  $\gamma + c_t(x_t)$ , which implies that  $\text{cost}(\text{Exp4}) \leq \text{cost}(\text{Hedge}) + \gamma T$ .

To summarize, we have made sure that:

**Lemma 7.5.**  $\mathbb{E}[R_{\text{Exp4}}(T)] \leq \mathbb{E}[\widehat{R}_{\text{Hedge}}(T)] + \gamma T$ .

For a crude regret bound, observe that  $c_t(a) \leq 1/q_t(a) \leq K/\gamma$ . So we can immediately upper-bound  $\mathbb{E}[\widehat{R}_{\text{Hedge}}(T)]$  using the regret bound for Hedge with  $U = K/\gamma$ . Then:

$$\mathbb{E}[R_{\text{Exp4}}(T)] \leq O(\sqrt{(K/\gamma) T \log N} + \gamma T).$$

To minimize regret, chose  $\gamma$  to (approximately equalize the two summands:

**Theorem 7.6.** Consider adversarial bandits with expert advice. Algorithm Exp4 with parameter  $\gamma = T^{-1/3} (K \log N)^{1/3}$  achieves regret

$$\mathbb{E}[R(T)] = O(T^{2/3} (K \log N)^{1/3}).$$



## 7.6 Improved analysis

We can obtain a better regret bound by analyzing the quantity

$$\widehat{G}_t := \sum_x p_t(x) \widehat{c}_t^2(x).$$

We will prove an upper bound on  $\mathbb{E}[G_t]$ , and use the corresponding regret bound for Hedge.

**Lemma 7.7.** Fix parameter  $\gamma \in (0, \frac{1}{2})$  and round  $t$ . Then  $\mathbb{E}[G_t] \leq \frac{K}{1-\gamma}$ .

*Proof.* For each action  $a$ , let  $X_a = \{x \in X : a_{t,x} = a\}$  be the set of all experts that recommended that action. Let

$$p_t(a) := \sum_{x \in X_a} p_t(x)$$

be the probability that the expert chosen by Hedge recommends action  $a$ . Then

$$q_t(a) = p_t(a)(1 - \gamma) + \frac{\gamma}{K} \geq (1 - \gamma) p_t(a).$$

For each expert  $x$ , letting  $a = a_{t,x}$  be the recommended action, we have:

$$\widehat{c}_t(x) = \widehat{c}_t(a) \leq \frac{c_t(a)}{q_t(a)} \leq \frac{1}{q_t(a)} \leq \frac{1}{(1 - \gamma) p_t(a)}.$$

Each realization of  $G_t$  satisfies:

$$\begin{aligned} \widehat{G}_t &= \sum_a \sum_{x \in X_a} p_t(x) \cdot \widehat{c}_t(x) \cdot \widehat{c}_t(x) && \text{(re-write as a sum over arms)} \\ &\leq \sum_a \sum_{x \in X_a} \frac{p_t(x)}{(1 - \gamma) p_t(a)} \widehat{c}_t(a) && \text{(replace one } \widehat{c}_t(a) \text{ with an upper bound)} \\ &= \frac{1}{1 - \gamma} \sum_a \frac{\widehat{c}_t(a)}{p_t(a)} \sum_{x \in X_a} p_t(x) && \text{(move "constant terms" out of the inner sum)} \\ &= \frac{1}{1 - \gamma} \sum_a \widehat{c}_t(a) && \text{(the inner sum is just } p_t(a)) \end{aligned}$$

To complete the proof, take expectations over both sides and recall that  $\mathbb{E}[\widehat{c}_t(a)] = c_t(a) \leq 1$ . □

Thus, we can use the regret bound for Hedge with  $U = K/(1 - \gamma)$ . This is a big improvement over  $U = K/\gamma$  that we used in the proof of Theorem 7.6. Let us complete the analysis, being a little careful so as to derive a better constant:

$$\begin{aligned} \mathbb{E}[\widehat{R}_{\text{Hedge}}(T)] &\leq 2\sqrt{3/(1 - \gamma)} \cdot \sqrt{TK \log N} \\ \mathbb{E}[R_{\text{Exp4}}(T)] &\leq 2\sqrt{3/(1 - \gamma)} \cdot \sqrt{TK \log N} + \gamma T && \text{(by Lemma 7.5)} \\ &\leq 2\sqrt{3} \cdot \sqrt{TK \log N} + 2\gamma T && \text{(since } \sqrt{1/(1 - \gamma)} \leq 1 + \gamma) \end{aligned} \tag{7.5}$$

(To derive (7.5), we assumed w.l.o.g. that  $2\sqrt{3} \cdot \sqrt{TK \log N} \leq T$ .)

This holds for any  $\gamma > 0$ . Therefore:

**Theorem 7.8.** Consider adversarial bandits with expert advice. Algorithm Exp4 with parameter  $\gamma \in (0, 1/2T)$  achieves regret

$$\mathbb{E}[R(T)] \leq 2\sqrt{3} \cdot \sqrt{TK \log N} + 1.$$

Note that we can choose parameter  $\gamma$  to be an arbitrary strictly positive number. So if we make  $\gamma$  really small, the algorithm's behavior will essentially coincide with that for  $\gamma = 0$ . Thus, we do not need the uniform exploration step after all! More formally, let  $\text{Exp4}(\gamma)$  denote the algorithm with parameter  $\gamma$ . Then:

**Claim 7.9.**  $\text{Exp4}(\gamma)$  is the same algorithm as  $\text{Exp4}(0)$ , with probability  $1 - \gamma T$ .

*Proof.* The two algorithms are different only if there exists a round in which  $\text{Exp4}(\gamma)$  chooses uniform exploration. Such round exists with probability at most  $\gamma T$ .  $\square$

Therefore, the difference in expected regret between  $\text{Exp4}(0)$  and  $\text{Exp4}(\gamma)$  is at most  $(\gamma T) \cdot T$ . Thus, using Theorem 7.8 with  $\gamma \in (0, 1/2T^2)$ , we obtain a regret bound for  $\gamma = 0$ :

**Theorem 7.10.** Algorithm  $\text{Exp4}$  with parameter  $\gamma = 0$  achieves regret

$$\mathbb{E}[R(T)] \leq 2\sqrt{3} \cdot \sqrt{TK \log N} + 1.$$

## 7.7 Remarks on Exp4

Exp4 stands for EXPLoration, EXPloitation, EXPonentiation, and EXPerts. The version for adversarial bandits without expert advice (*i.e.*, with experts that correspond to arms) is called Exp3. Both algorithms were introduced (and named) in the seminal paper (Auer et al., 2002b), along with several extensions and the  $\Omega(\sqrt{KT})$  lower bound.

The running time for Exp3 is very nice because in each round, we only need to do a small amount of computation to update the weights. However, in Exp4, the running time is  $O(N + K)$  per round, which can become very slow when  $N$ , the number of experts, is very large.

For several important special cases it is possible to obtain good regret *and* good running time, via different algorithms. One way to accomplish that is to reduce to a best-expert algorithm other than Hedge that obtains small regret and is computationally efficient where Hedge is not. In fact, the reduction described above, and the analysis leading up to Theorem 7.6, still applies if Hedge is replaced with an arbitrary best-experts algorithm for which we have a regret bound in terms of  $(K, T, U)$  (where  $U$  is a known upper bound on the costs). We will follow this idea next class: we will apply the same reduction to a different best-experts algorithm to obtain good regret and fast running time for an interesting special case.

Exp4 is a really powerful algorithm, and can be applied in many different settings:

- *contextual bandits*: we will see this application later in this course.
- *shifting regret*: In adversarial bandits, rather than compete with the best fixed arm, we can compete with “policies” that can change the arm from one round to another, but not too often. More formally, an *S-shifting policy* is sequence of arms  $\pi = (a_t : t \in [T])$  with at most  $S$  “shifts”: rounds  $t$  such that  $a_t \neq a_{t+1}$ . *S-shifting regret* is defined as the algorithm's total cost minus the total cost of the best *S-shifting policy*:

$$R_S(T) = \text{cost}(\text{ALG}) - \min_{S\text{-shifting policies } \pi} \text{cost}(\pi).$$

Consider this as a bandit problem with expert advice, where each *S-shifting policy* is an expert. The number of experts  $N \leq (KT)^S$ ; while it may be a large number,  $\log(N)$  is not too bad! Using Exp4 and plugging  $N \leq (KT)^S$  into Theorem 7.8, we obtain

$$\mathbb{E}[R_S(T)] = O(\sqrt{KST \cdot \log(KT)}). \quad (7.6)$$

- *Slowly changing costs*: Consider a randomized oblivious adversary such that the expected cost of each arm can change by at most  $\epsilon$  in each round. Rather than compete with the best fixed arm, we'd like to compete with the (cost of) the best *current* arm:  $c_t^* = \min_a c_t(a)$ . More formally, we'd like to minimize *dynamic regret*, defined as

$$R^*(T) = \min(\text{ALG}) - \sum_{t=1}^T c_t^*.$$

(Note that dynamic regret is the same as *T-shifting regret*.) One way to solve this problem is via *S-shifting regret*, for an appropriately chosen value of  $S$ .

## 7.8 Extensions of adversarial bandits

Much research has been done on various extensions of adversarial bandits. Some of these extensions are briefly discussed below.

- a stronger version of the same results: *e.g.*, extend to adaptive adversaries, obtain regret bounds that hold with high probability (rather than merely in expectation), and improve the constants. (All of the above has been done in the original paper Auer et al. (2002b)).
- an algorithm with  $O(\sqrt{KT})$  regret – shaving off that  $\log K$  factor and matching the lower bound up to constant factors – has been achieved in Audibert and Bubeck (2010).
- improved results for shifting regret: while applying Exp4 is computationally inefficient, Auer et al. (2002b) obtain the same regret bound (7.6) via a modification of Exp3 (and a more involved analysis), with essentially the same running time as Exp3. Further, a version of that algorithm achieves the same regret bound multiplied by  $\sqrt{S}$  which holds *for all S at once*.
- While we have only considered a finite number of experts and made no assumptions about what these experts are, similar results can be obtained for infinite classes of experts with some special structure. In particular, borrowing the tools from statistical learning theory, it is possible to handle classes of experts with a small VC-dimension.
- The notion of “best fixed arm” is not entirely satisfying for adaptive adversaries. An important line of research on adversarial bandits (*e.g.*, Dekel et al., 2012) considers notions of regret in which the benchmark is “counterfactual”: it refers not to the costs realized in a given run of the algorithm, but to the costs that would have been realized had the algorithm do something different.
- For adversarial bandits with slowly changing costs, there are algorithms with better regret bounds (than those from an application of Exp4), and fast running times (Slivkins and Upfal, 2008; Slivkins, 2014). Also, they handle more general versions of slowly changing costs, *e.g.*, allow the expected cost of each arm to evolve over time as an independent random walk on a bounded interval.
- Lastly, *data-dependent* regret bounds are desirable: not only optimal in the worst case, but also better for some “nice” special cases. In particular, Hazan and Kale (2009) obtain an improved regret bound when the realized cost of each arm  $a$  does not change too much compared to its average  $\text{cost}(a)/T$ : namely, the regret bound is of the form  $\tilde{O}(\sqrt{V})$ , where  $V = \sum_{t,a} \left( c_t(a) - \frac{\text{cost}(a)}{T} \right)^2$  is the *total variation* of the costs. Note, however, that IID 0-1 costs is essentially the worst case, as far as this particular regret bound is concerned. Bubeck and Slivkins (2012); Seldin and Slivkins (2014) pursue a different direction: they design algorithms that (essentially) match the regret of Exp3 in the worst case, and achieve logarithmic regret (as UCB1) if the costs are actually IID.

## 7.9 Bibliographic notes

The material in this lecture is presented in the original paper Auer et al. (2002a), as well as in various books and courses on online learning (*e.g.*, Cesa-Bianchi and Lugosi, 2006; Bubeck and Cesa-Bianchi, 2012). Among these sources, our presentation was most influenced by (Kleinberg, 2007, Week 8), but the “reduction to Hedge” is made more explicit.



## Chapter 8

# Bandits/Experts with Linear Costs

In this lecture, we will study bandit problems with linear costs. In this setting, actions are represented by vectors in a low-dimensional real space. For simplicity, we will assume that all actions lie within a unit hypercube:  $a \in [0, 1]^d$ . The action costs  $c_t(a)$  are linear in the vector  $a$ , namely:  $c_t(a) = a \cdot v_t$  for some weight vector  $v_t \in \mathbb{R}^d$  which is the same for all actions, but depends on the current time step. This problem is useful and challenging under full feedback as well as under bandit feedback; further, we will consider an intermediate regime called *semi-bandit feedback*.

The plan for today is as follows:

1. motivate and define new bandit problems which fall under “linear bandits”;
2. tackle this problem using the reduction from previous class (from bandit feedback to full feedback), augmented with some new tricks. Plugging Hedge into this reduction immediately gives a meaningful result for the new problems;
3. introduce a new algorithm for linear bandits with full feedback (*Follow the Perturbed Leader*). This algorithm is one of the fundamental results in the online learning literature, and (among many other applications) it plugs nicely into the above reduction, yielding a substantial improvement for the new problems.

### 8.1 Recap from last lecture

Let us recap some results from last lecture, restating them in a slightly more generic way.

As before, we have  $K$  different actions and fixed time horizon  $T$ . Each action has an associated cost  $c_t(a)$  at round  $t$ . The total cost of each action is the sum over all rounds:

$$\text{cost}(a) = \sum_{t=1}^T c_t(a).$$

Regret is defined as the difference between the total cost of the algorithm and the minimum total cost of an action:

$$R(T) = \text{cost}(\text{ALG}) - \min_a \text{cost}(a).$$

(We do not take expectations in this definition; instead, we upper-bound the expected regret.)

We have algorithm Hedge for the full-feedback setting. Against an adaptive randomized adversary, it achieves regret

$$\mathbb{E}[R(T)] \leq \theta(\sqrt{UT \log K}), \tag{8.1}$$

where  $U$  is some known upper bound on the action costs, i.e.  $U \geq c_t(a)$ .

We will also use a reduction from bandit feedback to full feedback: *i.e.*, this reduction uses a best-expert algorithm (for concreteness, Hedge) as a subroutine, and constructs a bandit algorithm. The reduction creates an expert for each arm: this expert always recommends this arm. The bandit algorithm proceeds as shown in Algorithm 8.

While several steps in the algorithm are unspecified, the analysis from last lecture applies word-by-word even at this level of generality: *i.e.*, no matter how these missing steps are filled in.

- 1 **Given:** best-experts algorithm ALG and parameter  $\gamma \in (0, \frac{1}{2})$ .
- 2 In each round  $t$ :
  1. call ALG, receive an expert  $x_t$  chosen for this round (internally drawn from some distribution  $p_t$  over the experts).
  2. with probability  $1 - \gamma$  follow expert  $x_t$ ; else chose arm via “random exploration” (TBD)
  3. observe cost  $c_t$  for the chosen arm, and perhaps some extra information (TBD)
  4. define “fake costs”  $\hat{c}_t(x)$  for each expert  $x$  (TBD), and return them to ALG.

**Algorithm 8:** Reduction from bandit feedback to full feedback.

**Theorem 8.1.** *Assume deterministic, oblivious adversary. Assume “fake costs” are defined so that  $\mathbb{E}[\hat{c}_t(x)|p_t] = c_t(x)$  and  $\hat{c}_t(x) \leq U'/\gamma$  for all experts  $x$  and all rounds  $t$ , where  $U'$  is some number that is known to the algorithm. Then the reduction (Algorithm 8) with best-experts algorithm Hedge and an appropriate choice of parameter  $\gamma$  (in terms of  $K, T, U'$ ) achieves regret*

$$\mathbb{E}[R(T)] \leq O(T^{2/3})(U' \log K)^{1/3}.$$

More generally,  $\mathbb{E}[R(T)]$  is at most the expected regret of ALG (on fake costs) plus  $\gamma T$ .

Later in this lecture, we will instantiate this algorithm (*i.e.*, specify the missing pieces) to obtain a solution for combinatorial semi-bandits.

## 8.2 The Online Routing Problem

In the *online routing problem* (a.k.a. *online shortest paths* problem), we are given a graph  $G$  with  $d$  edges, a source node  $s$ , and a target destination node  $t$ . The graph can either be directed or undirected. We have costs on edges that we interpret as delays in routing, or lengths in a shortest-path problem. The cost of a path is the sum over all edges in this path. The costs can change over time. Informally, the algorithm’s goal in each round is to find the “best route” from  $s$  to  $t$ : an  $s$ - $t$  path with minimal cost. Accordingly, we interpret it as a route that minimizes the total travel time from  $s$  to  $t$ .

Thus, an algorithm chooses among “actions” that correspond to  $s$ - $t$  paths in the graph. To cast this problem as a special case of “linear bandits”, observe that each path is can be specified by a subset of edges, which in turn can be specified by a  $d$ -dimensional binary vector  $a \in \{0, 1\}^d$ . Here edges of the graph are numbered from 1 to  $d$ , and for each edge  $e$  the corresponding entry  $a_e$  equals 1 if and only if this edge is included in the path. Let  $v_t = (c_t(e) : \text{edges } e \in G)$  be the vector of edge costs at round  $t$ . Then the cost of a path can be represented as a linear product  $c_t(a) = a \cdot v_t = \sum_{e \in [d]} a_e c_t(e)$ .

More formally, each round  $t$  in this problem proceeds as follows:

1. Adversary chooses costs  $c_t(e) \in [0, 1]$  for all edges  $e$ .
2. Algorithm chooses  $s$ - $t$ -path  $a_t \subset \text{Edges}(G)$ .
3. Incurs cost  $c_t(a_t) = \sum_{e \in a_t} a_e \cdot c_t(e)$  and receives feedback.

There are three version of the problem, depending on which feedback is received:

- bandit feedback: only  $c_t(a_t)$  is observed;
- semi-bandit feedback: costs on all edges in  $a_t$  are observed;
- full-feedback: cost on all edges are observed

We will mainly focus on the semi-bandit version, assuming the costs are selected by a deterministic oblivious adversary.

As a preliminary attempt, observe that we can use Exp3 algorithm for this problem, but both the regret as well as the runtime performance will be bad, since the regret in Exp3 is proportional to square root of the number of actions, which in this case may be exponential in  $d$ . Instead, we seek a regret bound of the form:

$$\mathbb{E}[R(T)] \leq \text{poly}(d) \cdot T^\beta, \quad \text{where } \beta < 1.$$

To this end, we will use the reduction (Algorithm 8) with best-expert algorithm Hedge. The “extra information” in this algorithm is the semi-bandit feedback. We will need to specify two other missing steps: the “random exploration” and the “fake costs”.

Without loss of generality, assume that each edge  $e$  belongs to some  $s$ - $t$  path  $a^{(e)}$  (else, we can just remove this edge from the graph). For the “random exploration step”, instead of selecting an action uniformly at random (as we did in Exp4), we select an edge  $e$  uniformly at random, and pick the corresponding path  $a^{(e)}$  as the chosen action.

To define fake costs, let  $\Lambda_{t,e}$  be the event that in round  $t$ , the algorithm chooses “random exploration”, and in random exploration, it chooses edge  $e$ . Note that  $\Pr[\Lambda_{t,e}] = \gamma/d$ . We define fake costs for each edge  $e$  separately:

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{\gamma/d} & \text{if event } \Lambda_{t,e} \text{ happens} \\ 0 & \text{otherwise} \end{cases} \quad (8.2)$$

The fake cost of a path is simply the sum of fake costs over its edges. This completes the specification of an algorithm for the online routing problem with semi-bandit feedback; we will refer to this algorithm as AlgSB.

As in the previous lecture, we prove that fake costs provide unbiased estimates for true costs:

$$\mathbb{E}[\hat{c}_t(e) \mid p_t] = c_t(e) \quad \text{for each round } t \text{ and each edge } e.$$

Since the fake cost for each edge is at most  $d/\gamma$ , it follows that  $c_t(a) \leq d^2/\gamma$  for each action  $a$ . Thus, we can immediately use Theorem 8.1 with Hedge and  $U' = d^2$ . For the number of actions, let us use an upper bound  $K \leq 2^d$ . Then  $U' \log K \leq d^3$ , and so:

**Theorem 8.2.** *Consider the online routing problem with semi-bandit feedback. Assume deterministic oblivious adversary. Algorithm AlgSB achieved regret  $\mathbb{E}[R(T)] \leq O(d T^{2/3})$ .*

*Remark 8.3.* Fake cost  $\hat{c}_t(e)$  is determined by the corresponding true cost  $c_t(e)$  and event  $\Lambda_{t,e}$  which does not depend on algorithm’s actions. Therefore, fake costs are chosen by a (randomized) oblivious adversary. In particular, in order to apply Theorem 8.1 with a different best-experts algorithm ALG, it suffices to have an upper bound on regret against an oblivious adversary.

### 8.3 Combinatorial (semi-)bandits

The online routing problem discussed above is a special case of *combinatorial semi-bandits*, where edges are replaced with  $d$  “atoms”, and  $s$ - $t$  paths are replaced with feasible subsets of atoms. The family of feasible subsets can be arbitrary.<sup>1</sup>

The algorithm and analysis from the previous section does not rely on any special properties of  $s$ - $t$  paths, and carry over word-by-word to combinatorial semi-bandits (replacing edges with atoms, and  $s$ - $t$  paths with feasible subsets). Therefore:

**Theorem 8.4.** *Consider combinatorial semi-bandits with deterministic oblivious adversary. Algorithm AlgSB achieved regret  $\mathbb{E}[R(T)] \leq O(d T^{2/3})$ .*

Let us list a few other notable special cases of combinatorial semi-bandits:

<sup>1</sup>Similarly, *combinatorial bandits* is the same problem, but with bandit feedback.

- *News Articles*: a news site needs to select a subset of articles to display to each user. The user can either click on an article or ignore it. Here, rounds correspond to users, atoms are the news articles, the reward is 1 if it is clicked and 0 otherwise, and feasible subsets here can encode a complicated set of constraints on selecting the articles.
- *Ads*: a website needs select a subset of ads to display to each user. For each displayed ad, we observe whether the user clicked on it, in which case the website receives some payment. The payment may, in principle, depend on an ad and on the user. Again: rounds correspond to users, atoms are the ads, and feasible subset can encode constraints on which ads can or cannot be shown together.
- *Network broadcast*: In each round, we want to transmit a packet from a source to multiple targets in the network. Then the feasible subsets correspond to (feasible) spanning subtrees for sending the packets.
- *A slate of news articles*: Similar to the news articles problem, except that ordering of the articles on the webpage matters. So the news site needs to select a slate (an ordered list) of articles. To represent this problem as an instance of combinatorial semi-bandits, define “atoms” to mean “this news article is chosen for that slot”; a subset of atoms is feasible if it defines a valid slate (*i.e.*, there is exactly one news article assigned to each slot).

Thus, combinatorial semi-bandits is a general setting which captures many motivating examples, and allows for a unified solution. Even better, this solution is itself an application of a more general framework. Such results are valuable even if each of the motivating examples is only a very idealized version of reality (*i.e.*, it captures some features of reality but ignores some others).

*Remark 8.5.* Solving the same problem with bandit feedback, known as *combinatorial bandits*, requires more work. The main challenge is that we need to estimate fake costs for all edges (resp., atoms) in the chosen path, whereas we only observe the total cost for the path. One solution is to construct a *basis*: a subset of feasible paths (called *base paths*) such that each edge can be represented as a linear combination of base paths. Then we can use a version of Algorithm 8 where in the “random exploration” step we chose uniformly among paths in this basis. This gives us fake costs on the base paths. Then fake cost on each edge can be defined as the corresponding linear combination over the base paths. This approach works as long as the linear coefficients are small. For more details, refer to Awerbuch and Kleinberg (2008).

Thus, we have an algorithm for combinatorial semi-bandits that achieves a good regret bound. However, the running time is slow because it relies on Hedge with a very large number of experts. We would like to design a faster algorithm, so that the runtime per time step polynomial in  $d$  (rather than exponential in  $d$ ).

Even if the costs on all atoms were known, to choose the best feasible action the algorithm would need to solve a combinatorial optimization problem: find a feasible subset with minimal cost. This problem is NP-hard in general, but allows polynomial-time solutions in many interesting special cases of combinatorial semi-bandits. For example, in the online routing problem discussed above the corresponding combinatorial optimization problem is a well-known shortest-path problem.

Thus, we should not hope to have a fast algorithm for the full generality of combinatorial semi-bandits. Instead, we assume that we have access to an *oracle*: an algorithm which finds the best feasible action given the costs on all atoms, and express the running time of our algorithm in terms of the number of oracle calls.

We will use this oracle to construct a new algorithm for combinatorial bandits with full feedback. In each round, this algorithm inputs only the costs on the atoms, and makes only one oracle call. It achieves regret  $\mathbb{E}[R(T)] \leq O(U\sqrt{dT})$  against an oblivious adversary, where  $U$  is a known upper bound on the costs of each expert in each round. The algorithm is called *Follow the Perturbed Leader* (FPL).

To solve the combinatorial semi-bandits, we use algorithm AlgSB as before, but replace Hedge with FPL; call the new algorithm AlgSBwithFPL. The same analysis from Section 8.2, applied to AlgSBwithFPL, yields regret

$$\mathbb{E}[R(T)] \leq O(U\sqrt{dT}) + \gamma T,$$

where  $U = d^2/\gamma$  is an upper bound on the fake costs. Optimizing the choice of parameter  $\gamma$ , we immediately obtain the following theorem:



**Theorem 8.6.** Consider combinatorial semi-bandits with deterministic oblivious adversary. Then algorithm AlgSBwithFPL with appropriately chosen parameter  $\gamma$  achieved regret

$$\mathbb{E}[R(T)] \leq O\left(d^{5/4} T^{3/4}\right).$$

*Remark 8.7.* In terms of the running time, it is essential that the fake costs on atoms can be computed *fast*: this is because the normalizing probability in (8.2) is known in advance.

Alternatively, we could have defined fake costs as

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{q_t(e|p_t)} & \text{if } e \in a_t \\ 0 & \text{otherwise,} \end{cases} \quad (8.3)$$

where  $q_t(e|p_t) = \Pr[e \in a_t | p_t]$ . This definition allows for the same regret bound (and, in fact, is somewhat better in terms of regret), but requires computing  $q_t(e|p_t)$ . (At least naively) this computation requires looking at probabilities for all actions, which leads to running times exponential in  $d$ , similar to Hedge.

## 8.4 Best-expert problem with linear costs: Follow the Perturbed Leader

In the second half of this lecture we will study the best-expert problem with linear costs. We will define an algorithm, called Follow the Perturbed Leader (FPL), and prove a regret bound against an oblivious adversary. In particular, this will suffice to complete the proof of Theorem 3.3 from part I of this class.

To be applicable for combinatorial (semi-)bandits, the actions need to be binary vectors. We will posit a more general setting in which the actions can be vectors  $a \in [0, 1]^d$ . More precisely, there is a fixed and known subset  $\mathcal{A} \subset [0, 1]^d$  of feasible actions. The costs are linear, in the sense that  $c_t(a) = a \cdot v_t$  for each round  $t$  and each action  $a$ , where  $v_t$  is the *cost vector* (same for all actions).

Thus, each round  $t$  proceeds as follows:

- the adversary chooses the cost vector  $v_t$ ,
- the algorithm chooses some feasible action  $a_t \in \mathcal{A}$ , receiving cost  $c_t(a_t) = a_t \cdot v_t$ ,
- the cost vector  $v_t$  is revealed.

For formal results, we need to posit an upper bound on the costs. For ease of notation, we assume that  $v_t \in [0, U/d]^d$ , for some known parameter  $U$ . Then  $c_t(a) \leq U$  for each action  $a$ .

We assume there exists an *optimization oracle*: a subroutine which computes the best action for a given cost vector. Formally, we represent this oracle as a function  $M$  from cost vectors to feasible actions such that  $M(v) \in \operatorname{argmin}_{a \in \mathcal{A}} a \cdot v$  (ties can be broken arbitrarily). As explained earlier, while in general the oracle is solving an NP-hard problem, polynomial-time algorithms exist for important special cases such as shortest paths. Thus, the implementation of the oracle is domain-specific, and is irrelevant to our analysis.

We will prove the following theorem:

**Theorem 8.8.** FPL achieves regret  $\mathbb{E}[R(T)] \leq 2U \cdot \sqrt{dT}$ . The running time of FPL in each round is polynomial in  $d$  plus one call to the oracle.

*Remark 8.9.* The set of feasible actions  $\mathcal{A}$  can be infinite (as long as the oracle is provided). For example, if  $\mathcal{A}$  is defined by a finite number of linear constraints, the oracle can be implemented via linear programming. Note that Hedge cannot handle infinitely many actions.

We use shorthand  $v_{i:j} = \sum_{t=i}^j v_t \in \mathbb{R}^d$  to denote the total cost vector between rounds  $i$  and  $j$ .

### 8.4.1 Follow the Perturbed Leader algorithm

**First attempt.** Consider a simple, exploitation-only algorithm called *Follow the Leader*:

$$a_{t+1} = M(v_{1:t}).$$

Equivalently, we play an arm with the lowest average cost, based on the observations so far. As we've discussed in Lecture 7, this approach works for fine IID costs. But it breaks for adversarial costs, as illustrated by the following example:

$$\begin{aligned} \mathcal{A} &= \{(1, 0), (0, 1)\} \\ v_1 &= \left(\frac{1}{3}, \frac{2}{3}\right) \\ v_t &= \begin{cases} (1, 0) & \text{if } t \text{ is even,} \\ (0, 1) & \text{if } t \text{ is odd.} \end{cases} \end{aligned}$$

Then the total cost vector is

$$v_{1:t} = \begin{cases} \left(i + \frac{1}{3}, i - \frac{1}{3}\right) & \text{if } t = 2i, \\ \left(i + \frac{1}{3}, i + \frac{2}{3}\right) & \text{if } t = 2i + 1. \end{cases}$$

Therefore, Follow the Leader picks action  $a_{t+1} = (0, 1)$  if  $t$  is even, and  $a_{t+1} = (1, 0)$  if  $t$  is odd. In both cases, we see that  $c_{t+1}(a_{t+1}) = 1$ . So the total cost for the algorithm is  $T$ , whereas any fixed action achieves total cost at most  $1 + T/2$ , so regret is, essentially,  $T/2$ .

The problem is synchronization: an oblivious adversary can force the algorithm to behave in a particular way, and synchronize its costs with algorithm's actions. In fact, similar issue arises for any deterministic algorithm (see Theorem 2.1 in Lecture 7, part II).

**Perturb the history!** Let us turn to randomized algorithms: let us use randomization to remove the synchronization that turned out so deadly in the example discussed above.

Rather than handing the oracle information provided solely by our adversary  $(v_{1:t-1})$ , we *perturb* the history. Namely, we pretend there was a 0-th round, with cost vector  $v_0 \in \mathbb{R}^d$  sampled from some distribution  $\mathcal{D}$ . We then give the oracle the "perturbed history", as expressed by the total cost vector  $v_{0:t-1}$ . This modified algorithm is known as *Follow the Perturbed Leader* (FPL).

Several choices for distribution  $\mathcal{D}$  lead to meaningful analyses. For ease of exposition, we posit that each coordinate of  $v_0$  is sampled independently and uniformly from the interval  $[-\frac{1}{\epsilon}, \frac{1}{\epsilon}]$ , where  $\epsilon$  is a parameter to be tuned according to  $T, U$ , and  $d$ .

### 8.4.2 Analysis of FPL

As a tool to analyze FPL, we consider a closely related algorithm called *Be the Perturbed Leader* (BPL). Imagine that when we need to choose an action at time  $t$ , we already know the cost vector  $v_t$ , and in each round  $t$  we choose  $a_t = M(v_{0:t})$ . Note that BPL is *not* an algorithm for the best-expert problem (because it uses  $v_t$  to choose  $a_t$ ).

The analysis proceeds in two steps. We first show that BPL comes "close" to the optimal cost

$$\text{OPT} = \min_{a \in \mathcal{A}} \text{cost}(a) = v_{1:t} \cdot M(v_{1:t}),$$

and then we show that FPL comes "close" to BPL. Specifically, we will prove:

**Lemma 8.10.** *For each value of parameter  $\epsilon > 0$ ,*

- (i)  $\text{cost}(\text{BPL}) \leq \text{OPT} + \frac{d}{\epsilon}$
- (ii)  $\mathbb{E}[\text{cost}(\text{FPL})] \leq \mathbb{E}[\text{cost}(\text{BPL})] + \epsilon \cdot U^2 \cdot T$

Then choosing  $\epsilon = \frac{\sqrt{d}}{U\sqrt{T}}$  gives Theorem 8.8. Curiously, note that part (i) makes a statement about realized costs, rather than expected costs.

**Step I: BPL comes close to OPT.** By definition of the oracle  $M$ ,

$$v \cdot M(v) \leq v \cdot a \quad \text{for any cost vector } v \text{ and feasible action } a. \tag{8.4}$$

The main argument proceeds as follows:

$$\begin{aligned}
\text{cost}(\text{BPL}) + v_0 \cdot M(v_0) &= \sum_{t=0}^T v_t \cdot M(v_{0:T}) && \text{(by definition of BPL)} \\
&\leq v_{0:T} \cdot M(v_{0:T}) && \text{(see Claim 8.11 below)} \\
&\leq v_{0:T} \cdot M(v_{1:T}) && \text{(by (8.4) with } a = M(v_{1:T})\text{)} \\
&= v_0 \cdot M(v_{1:T}) + \underbrace{v_{1:T} \cdot M(v_{1:T})}_{\text{OPT}}.
\end{aligned} \tag{8.5}$$

Subtracting  $v_0 \cdot M(v_0)$  from both sides, we obtain Lemma 8.10(i):

$$\text{cost}(\text{BPL}) - \text{OPT} \leq \underbrace{v_0}_{\in [-\frac{1}{\epsilon}, -\frac{1}{\epsilon}]^d} \cdot \underbrace{[M(v_{1:T}) - M(v_0)]}_{\in [-1, 1]^d} \leq \frac{d}{\epsilon}.$$

The missing step (8.5) follows from the following claim, with  $i = 0$  and  $j = T$ .

**Claim 8.11.** For all rounds  $i < j$ ,  $\sum_{t=1}^j v_t \cdot M(v_{i:t}) \leq v_{i:j} \cdot M(v_{i:j})$ .

*Proof.* The proof is by induction on  $j - i$ . The claim is trivially satisfied for the base case  $i = j$ . For the inductive step:

$$\begin{aligned}
\sum_{t=i}^{j-1} v_t \cdot M(v_{i:t}) &\leq v_{i:j-1} \cdot M(v_{i:j-1}) && \text{(by inductive hypothesis)} \\
&\leq v_{i:j-1} \cdot M(v_{i:j}) && \text{(by (8.4) with } a = M(v_{i:j})\text{)}.
\end{aligned}$$

Add  $v_j \cdot M(v_{i:j})$  to both sides to complete the proof.  $\square$

**Step II: FPL comes close to BPL.** We compare the expected costs of FPL and BPL round per round. Specifically, we prove that

$$\mathbb{E}[\underbrace{v_t \cdot M(v_{0:t-1})}_{c_t(a_t) \text{ for FPL}}] \leq \mathbb{E}[\underbrace{v_t \cdot M(v_{0:t})}_{c_t(a_t) \text{ for BPL}}] + \epsilon U^2. \tag{8.6}$$

Summing up over all  $T$  rounds gives Lemma 8.10(ii).

It turns out that for proving (8.6) much of the structure is irrelevant. Specifically, one can denote  $f(u) = v_t \cdot M(u)$  and  $v = v_{1:t-1}$ , and, essentially, prove (8.6) for arbitrary  $f(\cdot)$  and  $v$ .

**Claim 8.12.** For any vectors  $v \in \mathbb{R}^d$  and  $v_t \in [0, U/d]^d$ , and any function  $f : \mathbb{R}^d \rightarrow [0, R]$ ,

$$\left| \mathbb{E}_{v_0 \sim \mathcal{D}} [f(v_0 + v) - f(v_0 + v + v_t)] \right| \leq \epsilon UR.$$

Informally, this claim states that adding  $v_t$  to  $v_0 + v$  imparts a small change the output of the function  $f$ . This precisely captures why including noise in our cost vector is beneficial.

The proof of Claim 8.12 involves *coupling*, a general technique from probability theory. As a typical example for what this technique does, let  $X$  and  $Y$  be random variables drawn from some distributions  $P_X$  and  $P_Y$ , resp., and suppose we wish to analyze  $\mathbb{E}[f(X) + f(Y)]$ . Then

$$\mathbb{E}[f(X) + f(Y)] = \mathbb{E}[f(X') + f(Y')]$$

for any jointly distributed random variables  $X', Y'$  that are “consistent” with  $X, Y$  in the sense that the marginal distribution of  $X'$  and  $Y'$  coincide with  $P_X$  and  $P_Y$ . Therefore, instead of analyzing  $X, Y$  directly, it is sometimes easier to consider  $X', Y'$  for an appropriately constructed joint distribution.

Applied to Claim 8.12, this technique works as follows. We take  $X = v_0 + v$ ,  $Y = v_0 + v + v'$ . We construct jointly distributed random variables  $X', Y'$  such that their respective marginals coincide with  $X$  and  $Y$ , and moreover it holds that  $\Pr[X' = Y'] \geq 1 - \epsilon U$ . The claim follows trivially from the existence of such  $X', Y'$ . Constructing the desired  $X', Y'$  is a little tedious, we omit it for now.



## Chapter 9

# Contextual Bandits

### 9.1 Problem statement and examples

In this lecture, we will be talking about a generalization of bandits called *contextual bandits*. Here, each round  $t$  proceeds as follows:

- algorithm observes a “context”  $x_t$ ,
- algorithm picks an arm  $a_t$ ,
- reward  $r_t \in [0, 1]$  is realized.

The reward  $r_t$  depends both on the context  $x_t$  and the chosen action  $a_t$ . Formally, make the IID assumption:  $r_t$  is drawn independently from some distribution that depends on the  $(x_t, a_t)$  pair but not on  $t$ . The expected reward of action  $a$  given context  $x$  is denoted  $\mu(a|x)$ . This setting allows a limited amount of “change over time”, but this change is completely “explained” by the observable contexts. Throughout, we assume contexts  $x_1, x_2, \dots$  are chosen by an oblivious adversary.

The main motivation is that a user with a known “user profile” arrives in each round, and the context is the user profile. Some of the natural scenarios include: choosing which news articles to display to a given reader, which products to recommend to a given customer, and which interface options (*e.g.*, font sizes and colors, page layout, etc.) to use for a given website visitor. Rewards are often determined by user clicks, possibly in conjunction with other observable signals that correlate with revenue and/or user satisfaction. Naturally, rewards for the same action may be different for different users.

Contexts can include other things apart from (and instead of) the user profiles. First, known features of the environment, such as day of the week, time of the day, season (*e.g.*, Summer, pre-Christmas shopping season), proximity to a major event (*e.g.*, Olympics, elections). Second, the set of feasible actions: only a subset of actions may be allowed in a given round and/or for a given user. Third, actions can come with features of their own, and it may be convenient to include this information into the context, esp. if the action features change over time.

For ease of exposition, we assume a fixed and known time horizon  $T$ . The set of actions is denoted by  $\mathcal{A}$ , and  $K = |\mathcal{A}|$  is the number of actions.  $\mathcal{X}$  denotes the set of all contexts. For brevity, we will use CB instead of “contextual bandits”.

The reward of an algorithm ALG is  $\text{REW}(\text{ALG}) = \sum_{t=1}^T r_t$ , so that the expected reward is

$$\mathbb{E}[\text{REW}(\text{ALG})] = \sum_{t=1}^T \mu(a_t|x_t).$$

One natural goal is to compete with the best response:

$$\pi^*(x) = \max_{a \in \mathcal{A}} \mu(a|x) \tag{9.1}$$

Then regret is defined as

$$R(T) = \text{REW}(\pi^*) - \text{REW}(\text{ALG}). \tag{9.2}$$

## 9.2 Small number of contexts

One straightforward approach for CB is to apply a known bandit algorithm such as UCB1: namely, run a separate copy of this algorithm for each context.

**Initialization:** For each context  $x$ , create an instance  $\text{ALG}_x$  of algorithm ALG  
**for each round  $t$  do**  
    invoke algorithm  $\text{ALG}_x$  with  $x = x_t$   
    “play” action  $a_t$  chosen by  $\text{ALG}_x$ , return reward  $r_t$  to  $\text{ALG}_x$ .  
**end**

**Algorithm 9:** Bandit algorithm ALG for each context

Let  $n_x$  be the number of rounds in which context  $x$  arrives. Regret accumulated in such rounds is  $\mathbb{E}[R_x(T)] = O(\sqrt{Kn_x \ln T})$ . The total regret (from all contexts) is

$$\mathbb{E}[R(T)] = \sum_{x \in \mathcal{X}} \mathbb{E}[R_x(T)] = \sum_{x \in \mathcal{X}} O(\sqrt{Kn_x \ln T}) \leq O(\sqrt{KT |\mathcal{X}| \ln T}).$$

**Theorem 9.1.** *Algorithm 9 has regret  $\mathbb{E}[R(T)] = O(\sqrt{KT |\mathcal{X}| \ln T})$ , provided that the bandit algorithm ALG has regret  $\mathbb{E}[R_{\text{ALG}}(T)] = O(\sqrt{KT \log T})$ .*

*Remark 9.2.* The square-root dependence on  $|\mathcal{X}|$  is slightly non-trivial, because a completely naive solution would give linear dependence. However, this regret bound is still very high if  $|\mathcal{X}|$  is large, e.g., if contexts are feature vectors with a large number of features. To handle CB with a large number of contexts, we usually need to assume some structure, as we shall see in the rest of this lecture.

## 9.3 Lipschitz Contextual bandits (a simple example)

As a simple example of how structure allows to handle CB with a large number of contexts, let us consider (a simple example of) CB with Lipschitzness. Namely, we assume that contexts map into the  $[0, 1]$  interval (i.e.,  $\mathcal{X} \subset [0, 1]$ ) so that the expected rewards are Lipschitz w.r.t. the contexts:

$$|\mu(a|x) - \mu(a|x')| \leq L|x - x'| \quad \text{for any arms } a, a' \text{ and contexts } x, x', \quad (9.3)$$

where  $L$  is the Lipschitz constant.

One simple solution for this problem is given by uniform discretization of the context space. The approach is very similar to what we’ve seen for Lipschitz bandits and dynamic pricing; however, we need to be a little careful with some details: particularly, watch out for “discretized best response”.

Let  $S$  be the  $\epsilon$ -uniform mesh on  $[0, 1]$ , i.e., the set of all points in  $[0, 1]$  that are integer multiples of  $\epsilon$ . We take

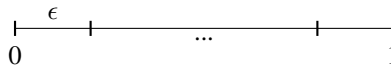


Figure 9.1: Discretization of Context Space

$\epsilon = 1/(d - 1)$ , where the integer  $d$  is the number of points in  $S$ , to be adjusted later in the analysis.

We will use the CB algorithm from Section 9.2, applied to context space  $S$ ; denote this algorithm as  $\text{ALG}_S$ . Let  $f_S(x)$  be a mapping from context  $x$  to the closest point in  $S$ :

$$f_S(x) = \min_{x' \in S} (\text{argmin } |x - x'|)$$

(the min is added just to break ties). The overall algorithm proceeds as follows: in each round  $t$ , we “pre-process” the context  $x_t$  by replacing it with  $f_S(x_t)$ , and call  $\text{ALG}_S$ .

The regret bound will have two summands: regret bound for  $\text{ALG}_S$  and (a suitable notion of) discretization error. Formally, let us define the “discretized best response”  $\pi_S^* : \mathcal{X} \rightarrow \mathcal{A}$ :

$$\pi_S^*(x) = \pi^*(f_S(x)) \quad \text{for each context } x \in \mathcal{X}.$$

Then regret of  $\text{ALG}_S$  and discretization error are defined as, resp.,

$$\begin{aligned} R_S(T) &= \text{REW}(\pi_S^*) - \text{REW}(\text{ALG}_S) \\ \text{DE}(S) &= \text{REW}(\pi^*) - \text{REW}(\pi_S^*). \end{aligned}$$

It follows that the “overall” regret is the sum  $R(T) = R_S(T) + \text{DE}(S)$ , as claimed. We have  $\mathbb{E}[R_S(T)] = O(\sqrt{KT|S|\ln T})$  from Lemma 9.1, so it remains to upper-bound the discretization error and adjust the discretization step  $\epsilon$ .

**Claim 9.3.**  $\mathbb{E}[\text{DE}(S)] \leq \epsilon LT$ .

*Proof.* For each round  $t$  and the respective context  $x = x_t$ ,

$$\begin{aligned} \mu(\pi_S^*(x) \mid f_S(x)) &\geq \mu(\pi^*(x) \mid f_S(x)) && \text{(by optimality of } \pi_S^*) \\ &\geq \mu(\pi^*(x) \mid x) - \epsilon L && \text{(by Lipschitzness).} \end{aligned}$$

Summing this up over all rounds  $t$ , we obtain

$$\mathbb{E}[\text{REW}(\pi_S^*)] \geq \text{REW}[\pi^*] - \epsilon LT. \quad \square$$

Thus, regret is

$$\mathbb{E}[R(T)] \leq \epsilon LT + O\left(\sqrt{\frac{1}{\epsilon} KT \ln T}\right) = O(T^{2/3}(LK \ln T)^{1/3}).$$

where for the last inequality we optimized the choice of  $\epsilon$ .

**Theorem 9.4.** *Consider the Lipschitz CB problem with contexts in  $[0, 1]$ . The uniform discretization approach yields regret  $\mathbb{E}[R(T)] = O(T^{2/3}(LK \ln T)^{1/3})$ .*

**Further remarks.** We can have a more general Lipschitz condition in which the right-hand side in (9.3) is replaced with  $\mathcal{D}_{\mathcal{X}}(x, x')$ , an arbitrary metric on contexts. Further, we can have a Lipschitz condition on both contexts and arms:

$$|\mu(a|x) - \mu(a'|x')| \leq D_{\mathcal{X}}(x, x') + D_{\mathcal{A}}(a, a') \quad \text{for any arms } a, a' \text{ and contexts } x, x', \quad (9.4)$$

where  $D_{\mathcal{X}}, D_{\mathcal{A}}$  are arbitrary metrics on contexts and arms, respectively.

Uniform discretization approach easily extends to this setting. Choose an  $\epsilon$ -mesh  $S_{\mathcal{X}}$  for  $(\mathcal{X}, \mathcal{D}_{\mathcal{X}})$  and an  $\epsilon$ -mesh  $S_{\mathcal{A}}$  for  $(\mathcal{A}, \mathcal{D}_{\mathcal{A}})$ .<sup>1</sup> Fix a bandit algorithm  $\text{ALG}$ , and run a separate copy  $\text{ALG}_x$  of this algorithm for each context  $x \in S_{\mathcal{X}}$ , with  $S_{\mathcal{A}}$  as a set of arms. In each round  $t$ , “round up” each context  $x_t$  to the nearest point  $x \in S_{\mathcal{X}}$ , and call the corresponding algorithm  $\text{ALG}_x$ .

Further, adaptive discretization approach can improve over uniform discretization, just as it does for (non-contextual) bandits. Here, it is useful to think about  $\mathcal{X} \times \mathcal{A}$ , the space of context-arms pairs. An algorithm in Slivkins (2014) implements adaptive discretization (and improves over the regret bounds for uniform discretization) by discretizing  $\mathcal{X} \times \mathcal{A}$ , rather than  $\mathcal{X}$  and/or  $\mathcal{A}$  separately. In fact, the algorithm and analysis extends to more general setting when the right-hand side of (9.4) is  $\mathcal{D}((x, a), (x', a'))$ , an arbitrary metric on context-arm pairs.

Consider the special case when the context  $x_t$  is simply the time  $t$  and the Lipschitz condition

$$|\mu(a|t) - \mu(a|t')| \leq \mathcal{D}_a(t, t'),$$

<sup>1</sup>As in Lecture 6, an  $\epsilon$ -mesh for a metric space  $(Y, \mathcal{D})$  is a subset  $S \subset Y$  such that for each point  $y \in Y$  there is a point  $y' \in S$  with  $\mathcal{D}(y, y') \leq \epsilon$ .

where  $\mathcal{D}_a$  is a metric on  $[T]$ , possibly parameterized by arm  $a$ . This describes a bandit problem with adversarial rewards and restriction that expected rewards can only change slowly. The paradigmatic special cases are  $\mathcal{D}_a(t, t') = \sigma_a |t - t'|$  (slow change per round), and  $\mathcal{D}_a(t, t') = \sigma_a \sqrt{|t - t'|}$  (e.g., mean reward of each arm  $a$  evolves as a random walk with step  $\sigma_a$ .) Interestingly, Slivkins (2014) solves these problems using a general algorithm for Lipschitz CB, and achieve near-optimal dynamic regret (see Section 7 in Lecture 8 for definition and discussion).

## 9.4 Linear Contextual Bandits (LinUCB algorithm without proofs)

Let us recap the setting of *linear bandits*. One natural formulation is that each arm  $a$  is characterized by a feature vector  $x_a \in [0, 1]^d$ , and the expected reward is linear in this vector:  $\mu(a) = x_a \cdot \theta$ , for some fixed but unknown vector  $\theta \in [0, 1]^d$ . One can think of the tuple

$$x = (x_a \in \{0, 1\}^d : a \in \mathcal{A}) \tag{9.5}$$

as a “static context” (i.e., a context that does not change from one round to another).

In linear *contextual* bandits, contexts are of the form (9.5), and the expected rewards are linear:

$$\mu(a|x) = x_a \cdot \theta_a \quad \text{for all arms } a \text{ and contexts } x, \tag{9.6}$$

for some fixed but unknown vector  $\theta = (\theta_a \in \mathcal{R}^d : a \in \mathcal{A})$ .<sup>2</sup>

This problem can be solved by a version of the UCB technique. Instead of constructing confidence bounds on the mean rewards of each arm, we do that for the  $\theta$  vector. Namely, in each round  $t$  we construct a “confidence region”  $C_t \subset \Theta$  such that  $\theta \in C_t$  with high probability. (Here  $\Theta$  be the set of all possible  $\theta$  vectors.) Then we use  $C_t$  to construct an UCB on the mean reward of each arm given context  $x_t$ , and play an arm with the highest UCB. This algorithm is known as LinUCB (see Algorithm 10).

**for round  $t$  do**

Form a confidence region  $C_t \in \Theta$  s.t.  $\theta \in C_t$  with high probability

For each arm  $a$ , compute  $\text{UCB}_t(a|x_t) = \sup_{\theta \in C_t} x_a \cdot \theta_a$

pick arm  $a$  which maximizes  $\text{UCB}_t(a|x_t)$ .

**end**

**Algorithm 10:** LinUCB: UCB-based algorithm for linear contextual bandits

To completely specify the algorithm, one needs to specify what the confidence region is, and how to compute the UCBs. This is somewhat subtle, and there are multiple ways to do that. Full specification and analysis of LinUCB is a topic for another lecture.

Suitably specified versions of LinUCB allow for rigorous regret bounds, and work well in experiments. The best known regret bound is of the form  $\mathbb{E}[R(T)] = \tilde{O}(\sqrt{dT})$ , and there is a nearly matching lower bound  $\mathbb{E}[R(T)] \geq \Omega(\sqrt{dT})$ . Interestingly, the algorithm is known to work well in practice even for scenarios without linearity.

**Bibliography note.** LinUCB has been introduced in Li et al. (2010), and analyzed in Chu et al. (2011).

## 9.5 Contextual Bandits with a Known Policy Class

We now consider a more general contextual bandit problem where we do not make any assumptions on the mean rewards. Instead, we make the problem tractable by making restricting the benchmark in the definition of regret (i.e., the first term in (9.2)). Specifically, we define a *policy* as a mapping from contexts to actions, and posit a known class of policies  $\Pi$ . Informally, algorithms only need to compete with the best policy in  $\pi$ . A big benefit of this approach is that it allows to make a clear connection to the “traditional” machine learning, and re-use some of its powerful tools.

<sup>2</sup>We allow the  $\theta_a$  in (9.6) to depend on the arm  $a$ . The special case when all  $\theta_a$ ’s are the same is also interesting.



For ease of presentation, we assume that contexts arrive as independent samples from some fixed distribution  $\mathcal{D}$  over contexts, which is known to the algorithm. For a given policy  $\pi$ , the expected reward is defined as

$$\mu(\pi) = \mathbb{E}_{x \in \mathcal{D}} [\mu(\pi(x)) | x] \quad (9.7)$$

The regret of an algorithm ALG w.r.t. policy class  $\Pi$  is defined as

$$R_{\Pi}(T) = T \max_{\pi \in \Pi} \mu(\pi) - \text{REW}(\text{ALG}). \quad (9.8)$$

Note that the definition (9.2) can be seen a special case when  $\Pi$  is the class of all policies.

**A simple but slow solution.** One simple solution is to use a powerful algorithm Exp4 that we’ve seen before, treating policies  $\pi \in \Pi$  as “experts”.

**Theorem 9.5.** *Algorithm Exp4 with expert set  $\Pi$  yields regret  $\mathbb{E}[R_{\Pi}(T)] = O(\sqrt{KT \ln |\Pi|})$ . However, the running time per round is linear in  $|\Pi|$ .*

This is a powerful result: it works for an arbitrary policy class  $\Pi$ , and the logarithmic dependence on  $|\Pi|$  makes the problem (reasonably) tractable, as far as regret bounds are concerned, even if the number of possible contexts is huge. Indeed, while there are  $K^{|\mathcal{X}|}$  possible policies, for many important special cases  $|\Pi| = K^c$ , where  $c$  depends on the problem parameters but not on  $|\mathcal{X}|$ . However, Exp4 has a very bad running time which scales as  $|\Pi|$  rather than  $\log |\Pi|$ , which makes the algorithm prohibitively slow in practice.

**Connection to a classification problem.** We would like to achieve similar regret rates — (poly)logarithmic in  $|\Pi|$  — but with a faster algorithm. To this end, we make a connection to a well-studied classification problem in “traditional” machine learning. This connection will also motivate the choices for the policy class  $\Pi$ .

To build up the motivation, let us consider “CB with predictions”: a contextual analog of the “bandits with prediction” problem that we’ve seen before. In “CB with predictions”, in the end of each round  $t$  the algorithm additionally needs to predict a policy  $\pi_t \in \Pi$ . The prediction is the algorithm’s current guess for the best policy in  $\Pi$ : a policy  $\pi = \pi_{\Pi}^*$  which maximizes  $\mu(\pi)$ .

Let’s focus on a much simpler sub-problem: how to predict the best policy when so much data is collected that the mean rewards  $\mu$  are essentially known. More formally:

$$\text{Given } \mu(a|x) \text{ for all arms } a \text{ and contexts } x \in \mathcal{X}, \text{ find policy } \pi \in \Pi \text{ so as to maximize } \mu(\pi). \quad (9.9)$$

Intuitively, any good solution for “CB with predictions” would need to solve (9.9) as a by-product. A very productive approach for designing CB algorithms goes in the opposite direction: essentially, it assumes that we already have a good algorithm for (9.9), and uses it as a subroutine for CB.

*Remark 9.6.* We note in passing that (9.9) is also well-motivated as a stand-alone problem. Of course, if  $\mu(a|x)$  is known for all arms and all contexts, then an algorithm can simply compute the best response (9.1), so computing the best policy seems redundant. The interpretation that makes (9.9) interesting is that  $\mathcal{X}$  is the set of all contexts already seen by the algorithm, and the goal is to choose an action for a new contexts that may arrive in the future.

For our purposes, it suffices to consider a special case of (9.9) with finitely many contexts and uniform distribution  $\mathcal{D}$ . Restating in a more flexible notation, the problem is as follows:

- Input:  $N$  data points  $(x_i, c_i(a) \in \mathcal{A}), i = 1, \dots, N$ , where  $x_i \in \mathcal{X}$  are distinct contexts, and  $c_i(a)$  is a “fake cost” of action  $a$  for context  $x_i$ .
- Output: policy  $\pi \in \Pi$  to approximately minimizes the empirical policy cost

$$c(\pi) = \frac{1}{N} \sum_{i=1}^N c_i(\pi(x_i)). \quad (9.10)$$

To recap, this is (still) a simple sub-problem of “CB with predictions”, which

This happens to be a well-studied problem called “cost-sensitive multi-class classification” for policy class  $\Pi$ . An algorithm for this problem will henceforth be called a *classification oracle* for  $\Pi$ . While the exact optimization problem is NP-hard in the worst case for many natural policy classes, practically efficient algorithms have been designed for several important policy classes such as linear classifiers, decision trees and neural nets.

*Remark 9.7.* W.l.o.g., one can start with non-distinct contexts  $x'_1, \dots, x'_N \in \mathcal{X}$ . Then we can define distinct contexts  $x'_i = (x_i, i)$ ,  $i \in [N]$ . Each policy  $\pi$  is extended to the  $x'_i$ 's by writing  $\pi(x'_i) = \pi(x_i)$ .

**A simple oracle-based algorithm.** We will use a classification oracle  $\mathcal{O}$  as a subroutine for designing computationally efficient CB algorithms. The running time is then expressed in terms of the number of oracle calls (the implicit assumption being that each oracle call is reasonably fast). Crucially, CB algorithms designed with this approach can use any available classification oracle; then the relevant policy class  $\Pi$  is simply the policy class that the oracle optimizes over.

Let us consider a simple explore-then-exploit CB algorithm based on this approach. First, we explore uniformly for the first  $N$  rounds, where  $N$  is a parameter. Each round  $t = 1, \dots, N$  of exploration gives a data point  $(x_t, c_t(a) \in \mathcal{A})$  for the classification oracle, where the “fake costs” are given by inverse propensity scoring:

$$c_t(a) = \begin{cases} -r_t K & \text{if } a = a_t \\ 0, & \text{otherwise.} \end{cases} \quad (9.11)$$

Finally, we call the oracle and use the policy returned by the oracle in the remaining rounds; see Algorithm 11.

**1 Parameter:** exploration duration  $N$ , classification oracle  $\mathcal{O}$

1. Explore uniformly for the first  $N$  rounds: in each round, pick an arm u.a.r.
2. Call the classification oracle with data points  $(x_t, c_t(a) \in \mathcal{A})$ ,  $t \in [N]$  as per (9.11).
3. Exploitation: in each subsequent round, use the policy  $\pi_0$  returned by the oracle.

**Algorithm 11:** Explore-then-exploit with a classification oracle

*Remark 9.8.* Algorithm 11 is modular in two ways: it can take an arbitrary classification oracle, and, in principle, use any other unbiased estimator instead of (9.11). In particular, the proof below only uses the fact that (9.11) is an unbiased estimator with  $c_t(a) \leq K$ .

For a simple analysis, assume that the rewards are in  $[0, 1]$  and that the oracle  $\mathcal{O}$  is *exact*, in the sense that it returns a policy  $\pi \in \Pi$  that exactly maximizes  $\mu(\pi)$ .

**Theorem 9.9.** *Assume rewards lie in  $[0, 1]$ . Let  $\mathcal{O}$  be an exact classification oracle for some policy class  $\Pi$ . Algorithm 11 parameterized with oracle  $\mathcal{O}$  and  $N = T^{2/3}(K \log(|\Pi|T))^{1/3}$  has regret*

$$\mathbb{E}[R_{\Pi}(T)] = O(T^{2/3})(K \log(|\Pi|T))^{1/3}.$$

*Proof.* Let us consider an arbitrary  $N$  for now. For a given policy  $\pi$ , we estimate its expected reward  $\mu(\pi)$  using the empirical policy costs from (9.10), where the action costs  $c_t(\cdot)$  are from (9.11). Let us prove that  $-c(\pi)$  is an unbiased estimate for  $\mu(\pi)$ :

$$\begin{aligned} \mathbb{E}[c_t(a)|x_t] &= -\mu(a|x_t) && \text{(for each action } a \in \mathcal{A}) \\ \mathbb{E}[c_t(\pi(x_t))|x_t] &= -\mu(\pi(x)|x_t) && \text{(plug in } a = \pi(x_t)) \\ \mathbb{E}_{x_t \sim D}[c_t(\pi(x_t))] &= \mathbb{E}_{x_t \sim D}[\mu(\pi(x_t))|x_t] && \text{(take expectation over both } c_t \text{ and } x_t) \\ &= -\mu(\pi), \end{aligned}$$

which implies  $\mathbb{E}[c(\pi)] = -\mu(\pi)$ , as claimed.

Now, let us use this estimate to set up a “clean event”:

$$\{|c(\pi) - \mu(\pi)| \leq \text{conf for all policies } \pi \in \Pi\},$$

where the “confidence radius” is

$$\text{conf}(N) = O\left(\sqrt{\frac{K \log(T|\Pi|)}{N}}\right).$$

We can prove that the clean event does indeed happen with probability at least  $1 - \frac{1}{T}$ , say, as an easy application of Chernoff Bounds. For intuition, the  $K$  is present in the confidence radius is because it upper-bounds the costs  $c_t(\cdot)$ . The  $|\Pi|$  is there (inside the log) because we take a Union Bound across all policies. And the  $T$  is there because we need the “error probability” to be on the order of  $\frac{1}{T}$ .

Let  $\pi^* = \pi_{\Pi}^*$  be an optimal policy. Since we have an exact classification oracle,  $c(\pi_0)$  is maximal among all policies  $\pi \in \Pi$ . In particular,  $c(\pi) \geq c(\pi^*)$ . If the clean event holds, then

$$\mu(\pi^*) - \mu(\pi_0) \leq 2 \text{conf}(N).$$

Thus, each round in exploitation contributes at most  $\text{conf}$  to expected regret. And each round of exploration contributes at most 1. It follows that  $\mathbb{E}[R_{\Pi}(T)] \leq N + 2T \text{conf}(N)$ . Choosing  $N$  so that  $N = O(T \text{conf}(N))$ , we obtain  $N = T^{2/3}(K \log(|\Pi|T))^{1/3}$  and  $\mathbb{E}[R_{\Pi}(T)] = O(N)$ .  $\square$

*Remark 9.10.* Suppose classification oracle in Theorem 9.9 is only approximate: say, it returns a policy  $\pi_0 \in \Pi$  which optimizes  $c(\cdot)$  up to an additive factor of  $\epsilon$ . It is easy to see that expected regret increases by an additive factor of  $\epsilon T$ . In practice, there may be a tradeoff between the approximation guarantee  $\epsilon$  and the running time of the oracle.

**Further discussion.** The oracle-based approach to CB was pioneered in Langford and Zhang (2007), with a slightly more complicated “epsilon-greedy”-style algorithm, and a similar analysis.

The key features of Theorem 9.9 are: polylogarithmic dependence on  $|\Pi|$ ,  $T^{2/3}$  dependence on  $T$  (up to log factors), and a single oracle call per round. Follow-up work investigated the tradeoff between dependence on  $T$  and the number of oracle calls (keeping  $\text{polylog}(|\Pi|)$  dependence as a must). One concrete goal here is to achieve the optimal  $\sqrt{T}$  dependence of regret on  $T$  with only a “small” number of oracle calls. In particular, a recent breakthrough result of (Agarwal et al., 2014) achieved regret bound  $O(\sqrt{KT \log(T|\Pi|)})$  with only  $\tilde{O}(\sqrt{KT/\log|\Pi|})$  oracle calls across all  $T$  rounds. However, this result does not appear to extend to approximate oracles, and calls the oracle on a number of carefully constructed artificial problem instances (so if a given classification oracle performs greatly in practice, this would not necessarily carry over to a great performance on these artificial problem instances). However, their algorithm appears to perform well in simulations.

Another recent break-through (Syrkkanis et al., 2016a; Rakhlin and Sridharan, 2016; Syrkkanis et al., 2016b) extends CB with classification oracles to adversarial rewards. The best current result in this line of work involves  $T^{2/3}$  regret, so there may still be room for improvement.

## 9.6 Contextual bandits as a system (to be written, based on (Agarwal et al., 2016))

This section will be based on Section 3 of Agarwal et al. (2016).



## Chapter 10

# Bandits with Knapsacks

### 10.1 Motivating Example: Dynamic Pricing

The basic version of the dynamic pricing problem is as follows. A seller has  $B$  items for sale: copies of the same product. There are  $T$  rounds. In each round  $t$ , a new customer shows up, and one item is offered for sale. Specifically, the algorithm chooses a price  $p_t \in [0, 1]$ . The customer shows up, having in mind some value  $v_t$  for this item, buys the item if  $v_t \geq p_t$ , and does not buy otherwise. The customers' values are chosen independently from some fixed but unknown distribution. The algorithm earns  $p_t$  if there is a sale, and 0 otherwise. The algorithm stops after  $T$  rounds or after there are no more items to sell, whichever comes first; in the former case, there is no premium or rebate for the left-over items. The algorithm's goal is to maximize revenue.

The simplest version  $B = T$  (i.e., unlimited supply of items) is a special case of bandits with IID rewards, where arms corresponds to prices. However, with  $B < T$  we have a “global” constraint: a constraint that binds across all rounds and all actions.

In more general versions of dynamic pricing, we may have multiple products for sale, with a limited supply of each product. For example, in each round the algorithm may offer one copy of each product for sale, and assign a price for each product, and the customer chooses a subset of products to buy. What makes this generalization interesting is that customers may have valuations over *subsets* of products that are not necessarily additive: e.g., a pair of shoes is usually more valuable than two copies of the left shoe. Here “actions” correspond to price vectors, and we have a separate “global constraint” on each product.

### 10.2 General Framework: Bandits with Knapsacks (BwK)

We introduce a general framework for bandit problems with “global constraints” such as supply constraints in dynamic pricing. We call this framework “Bandits with Knapsacks” because of an analogy with the well-known *knapsack problem* in algorithms. In that problem, one has a knapsack of limited size, and multiple items each of which has a value and takes a space in the knapsack. The goal is to assemble the knapsack: choose a subset of items that fits in the knapsack so as to maximize the total value of these items. Similarly, in dynamic pricing each action  $p_t$  has “value” (the revenue from this action) and “size in the knapsack” (namely, number of items sold). However, in BwK the “value” and “size” of a given action are not known in advance.

The framework of BwK is as follows. There are  $k$  arms and  $d$  resources, where each “resource” represents a global constraint such as limited supply of a given product. There are  $B_i$  units of each resource  $i$ . There are  $T$  rounds, where  $T$  is a known time horizon. In each round, algorithm chooses an arm, receives a reward, and also consumes some amount of each resource. Thus, the outcome of choosing an arm is now a  $(d + 1)$ -dimensional vector rather than a scalar: the first component of this vector is the reward, and each of the remaining  $d$  components describe the consumption of the corresponding resource. We have the “IID assumption”, which now states that for each arm  $a$  the outcome vector is sampled independently from a fixed distribution over outcome vectors. The algorithm stops as soon as we are out of time or any of the resources; the algorithm's goal is to maximize the total reward in all preceding rounds.

We think of time (*i.e.*, the number of rounds) as one of the  $d$  resources: this resource has supply  $T$  and is consumed at the unit rate by each action. As a technical assumption, we assume that the reward and consumption of each resource in each round lies in  $[0, 1]$ . Formally, an instance of BwK is specified by parameters  $(d, k; B_1, \dots, B_d)$  and a mapping from arms to distributions over outcome vectors.

### 10.2.1 Examples

We illustrate the generality of BwK with several basic examples.

**Dynamic pricing.** Dynamic pricing with a single product is a special case of BwK with two resources: time (*i.e.*, the number of customers) and supply of the product. Actions correspond to chosen prices  $p$ . If the price is accepted, reward is  $p$  and resource consumption is 1. Thus, the outcome vector is

$$\begin{cases} (p, 1) & \text{if price } p \text{ is accepted} \\ (0, 0) & \text{otherwise.} \end{cases}$$

**Dynamic pricing for hiring.** A contractor on a crowdsourcing market has a large number of similar tasks, and a fixed amount of money, and wants to hire some workers to perform these tasks. In each round  $t$ , a worker shows up, the algorithm chooses a price  $p_t$ , and offers a contract for one task at this price. The worker has a value  $v_t$  in mind, and accepts the offer (and performs the task) if and only if  $p_t \geq v_t$ . The goal is to maximize the number of completed tasks.

This problem is a special case of BwK with two resources: time (*i.e.*, the number of workers) and contractor's budget. Actions correspond to prices  $p$ ; if the offer is accepted, the reward is 1 and the resource consumption is  $p$ . So, the outcome vector is

$$\begin{cases} (1, p) & \text{if price } p \text{ is accepted} \\ (0, 0) & \text{otherwise.} \end{cases}$$

**PPC ads with budgets.** There is an advertising platform with pay-per-click ads (advertisers pay only when their ad is clicked). For any ad  $a$  there is a known per-click reward  $r_a$ : the amount an advertiser would pay to the platform for each click on this ad. If shown, each ad  $a$  is clicked with some fixed but unknown probability  $q_a$ . Each advertiser has a limited budget of money that he is allowed to spend on her ads. In each round, a user shows up, and the algorithm chooses an ad. The algorithm's goal is to maximize the total reward.

This problem is a special case of BwK with one resource for each advertiser (her budget) and the "time" resource (*i.e.*, the number of users). Actions correspond to ads. Each ad  $a$  generates reward  $r_a$  if clicked, in which case the corresponding advertiser spends  $r_a$  from her budget. In particular, for the special case of one advertiser the outcome vector is:

$$\begin{cases} (r_a, r_a) & \text{if ad } a \text{ is clicked} \\ (0, 0) & \text{otherwise.} \end{cases}$$

**Repeated auction.** An auction platform such as eBay runs many instances of the same auction to sell  $B$  copies of the same product. At each round, a new set of bidders arrives, and the platform runs a new auction to sell an item. The auction is parameterized by some parameter  $\theta$ : *e.g.*, the second price auction with the reserve price  $\theta$ . In each round  $t$ , the algorithm chooses a value  $\theta = \theta_t$  for this parameter, and announces it to the bidders. Each bidder is characterized by the value for the item being sold; in each round, the tuple of bidders' values is drawn from some fixed but unknown distribution over such tuples. Algorithm's goal is to maximize the total profit from sales; there is no reward for the remaining items.

This is a special case of BwK with two resources: time (*i.e.*, the number of auctions) and the limited supply of the product. Arms correspond to feasible values of parameter  $\theta$ . The outcome vector in round  $t$  is:

$$\begin{cases} (p_t, 1) & \text{if an item is sold at price } p_t \\ (0, 0) & \text{if an item is not sold.} \end{cases}$$

The price  $p_t$  is determined by the parameter  $\theta$  and the bids in this round.

**Repeated bidding on a budget.** Let’s look at a repeated auction from a bidder’s perspective. It may be a complicated auction that the bidder does not fully understand. In particular, the bidder often not know the best bidding strategy, but may hope to learn it over time. Accordingly, we consider the following setting. In each round  $t$ , one item is offered for sale. An algorithm chooses a bid  $b_t$  and observes whether it receives an item and at which price. The outcome (whether we win an item and at which price) is drawn from a fixed but unknown distribution. The algorithm has a limited budget and aims to maximize the number of items bought.

This is a special case of BwK with two resources: time (*i.e.*, the number of auctions) and the bidder’s budget. The outcome vector in round  $t$  is:

$$\begin{cases} (1, p_t) & \text{if the bidder wins the item and pays } p_t \\ (0, 0) & \text{otherwise.} \end{cases}$$

The payment  $p_t$  is determined by the chosen bid  $b_t$ , other bids, and the rules of the auction.

### 10.2.2 BwK compared to the “usual” bandits

BwK is complicated in three different ways:

1. In bandits with IID rewards, one thing that we can almost always do is the explore-first algorithm. However, Explore-First does not work for BwK. Indeed, suppose we have an exploration phase of a fixed length. After this phase we learn something, but what if we are now out of supply? Explore-First provably does not work in the worst case if the budgets are small enough: less than a constant fraction of the time horizon.
2. In bandits with IID rewards, we usually care about per-round expected rewards: essentially, we want to find an arm with the best per-round expected reward. But in BwK, this is not the right thing to look at, because an arm with high per-round expected reward may consume too much resource(s). Instead, we need to think about the *total* expected reward over the entire time horizon.
3. Regardless of the distinction between per-round rewards and total rewards, we usually want to learn the best arm. But for BwK, the best arm is not the right thing to learn! Instead, the right thing to learn is the best *distribution* of arms. More precisely, a “fixed-distribution policy” — an algorithm that samples an arm independently from a fixed distribution in each round — may be much better than any “fixed-arm policy”.

The following example demonstrates this point. Assume we have two resources: a “horizontal” resource and a “vertical” resource, both with budget  $B$ . We have two actions: the “horizontal” action which spends one unit of the “horizontal” resource and no “vertical” resource, and the vice versa for the “vertical” action. Each action brings reward of 1. Then best fixed action gives us the total reward of  $B$ , but alternating the two actions gives the total reward of  $2 \times B$ . And the uniform distribution over the two actions gives essentially the same expected total reward as alternating them, up to a low-order error term. Thus, the best fixed distribution performs better by a factor of 2 in this example.

## 10.3 Regret bounds

Algorithms for BwK compete with a very strong benchmark: the best algorithm for a given problem instance  $\mathcal{I}$ . Formally, the benchmark is defined as

$$\text{OPT} \triangleq \text{OPT}(\mathcal{I}) \triangleq \sup_{\text{algorithms ALG}} \text{REW}(\text{ALG}|\mathcal{I}),$$

where  $\text{REW}(\text{ALG}|\mathcal{I})$  is the expected total reward of algorithm ALG on problem instance  $\mathcal{I}$ .

It can be proved that competing with OPT is essentially the same as competing with the best fixed distribution over actions. This simplifies the problem considerably, but still, there are infinitely many distributions even for two actions.

There algorithms have been proposed, all with essentially the same regret bound:

$$\text{OPT} - \text{REW}(\text{ALG}) \leq \tilde{O} \left( \sqrt{k \cdot \text{OPT}} + \text{OPT} \sqrt{\frac{k}{B}} \right), \quad (10.1)$$

where  $B = \min_i B_i$  is the smallest budget. The first summand is essentially regret from bandits with IID rewards, and the second summand is really due to the presence of budgets.

This regret bound is worst-case optimal in a very strong sense: for any algorithm and any given triple  $(k, B, T)$  there is a problem instance of BwK with  $k$  arms, smallest budget  $B$ , and time horizon  $T$  such that this algorithm suffers regret

$$\text{OPT} - \text{REW}(\text{ALG}) \geq \Omega \left( \min \left( \text{OPT}, \sqrt{k \cdot \text{OPT}} + \text{OPT} \sqrt{\frac{k}{B}} \right) \right). \quad (10.2)$$

However, the lower bound is proved for a particular family of problem instances, designed specifically for the purpose of proving this lower bound. So it does not rule out better regret bounds for some interesting special cases.

## 10.4 Fractional relaxation

While in BwK time is discrete and outcomes are randomized, it is very useful to consider a “fractional relaxation”: a version of the problem in which time is fractional and everything happens exactly according to the expectation. We use the fractional relaxation to approximate the expected total reward from a fixed distribution over arms, and upper-bound OPT in terms of the best distribution.

To make this more formal, let  $r(\mathcal{D})$  and  $c_i(\mathcal{D})$  be, resp., the expected per-round reward and expected per-round consumption of resource  $i$  if an arm is sampled from a given distribution  $\mathcal{D}$  over arms. The fractional relaxation is a version of BwK where:

1. each round has a (possibly fractional) “duration”
2. in each round  $t$ , the algorithm chooses duration  $\tau = \tau_t$  and distribution  $\mathcal{D} = \mathcal{D}_t$  over arms,
3. the reward and consumption of each resource  $i$  are, resp.,  $\tau r(\mathcal{D})$  and  $\tau c_i(\mathcal{D})$ ,
4. there can be arbitrarily many rounds, but the total duration cannot exceed  $T$ .

As as shorthand, we will say that the expected total reward of  $\mathcal{D}$  is the expected total reward of the fixed-distribution policy which samples an arm from  $\mathcal{D}$  in each round. We approximate the expected total reward of  $\mathcal{D}$  in the original problem instance of BwK with that in the fractional relaxation. Indeed, in the fractional relaxation one can continue using  $\mathcal{D}$  precisely until some resource is completely exhausted, for the total duration of  $\min_i B_i / c_i(\mathcal{D})$ ; here the minimum is over all resources  $i$ . Thus, the expected total reward of  $\mathcal{D}$  in the fractional relaxation is

$$\text{FR}(\mathcal{D}) = r(\mathcal{D}) \min_{\text{resources } i} \frac{B_i}{c_i(\mathcal{D})}.$$

Note that  $\text{FR}(\mathcal{D})$  is not known to the algorithm, but can be approximately learned over time.

Further, one can prove that

$$\text{OPT} \leq \text{OPT}_{\text{FR}} \triangleq \sup_{\text{distributions } \mathcal{D} \text{ over arms}} \text{FR}(\mathcal{D}).$$

[as: **TODO: insert a proof via primal feasibility.**]

In fact, the algorithms for BwK compete with the “relaxed benchmark”  $\text{OPT}_{\text{FR}}$  rather than with the original benchmark OPT.

Informally, we are interested in distributions  $\mathcal{D}$  such that  $\text{FR}(\mathcal{D}) = \text{OPT}_{\text{FR}}(\mathcal{D})$ ; we call them “fractionally optimal”. Interestingly, one can prove (using some linear algebra) that there exists a fractionally optimal distribution  $\mathcal{D}$  with two additional nice properties:

- $\mathcal{D}$  randomizes over at most  $d$  arms,
- $c_i(\mathcal{D}) \leq B_i/T$  for each resource  $i$  (*i.e.*, in the fractional relaxation, we run out of all resources simultaneously).



## 10.5 “Clean event” and confidence regions

Another essential preliminary step is to specify the high-probability event (*clean event*) and the associated confidence intervals. As for bandits with IID rewards, the clean event specifies the high-confidence interval for the expected reward of each action  $a$ :

$$|\bar{r}_t(a) - r(a)| \leq \text{conf}_t(a) \quad \text{for all arms } a \text{ and rounds } t,$$

where  $\bar{r}_t(a)$  is the average reward from arm  $a$  by round  $t$ , and  $\text{conf}_t(a)$  is the confidence radius for arm  $a$ . Likewise, the clean event specifies the high-confidence interval for consumption of each resource  $i$ :

$$|\bar{c}_{i,t}(a) - c(a)| \leq \text{conf}_t(a) \quad \text{for all arms } a, \text{ rounds } t, \text{ and resources } i,$$

where  $\bar{c}_{i,t}(a)$  is the average resource- $i$  consumption from arm  $a$  so far.

Jointly, these confidence intervals define the “confidence region” on the matrix

$$\mu = (r(a); c_1(a), \dots, c_d(a)) : \quad \text{for all arms } a$$

such that  $\mu$  belongs to this confidence region with high probability. Specifically, the confidence region at time  $t$ , denoted  $\text{ConfRegion}_t$ , is simply the product of the corresponding confidence intervals for each entry of  $\mu$ . We call  $\mu$  the *latent structure* of the problem instance.

**Confidence radius.** How should we define the confidence radius? The standard definition is

$$\text{conf}_t(a) = O\left(\sqrt{\frac{\log T}{n_t(a)}}\right),$$

where  $n_t(a)$  is the number of samples from arm  $a$  by round  $t$ . Using this definition would result in a meaningful regret bound, albeit not as good as (10.1).

In order to arrive at the optimal regret bound (10.1), it is essential to use a more advanced version:

$$\text{conf}_t(a) = O\left(\sqrt{\frac{\nu \log(T)}{n_t(a)}} + \frac{1}{n_t(a)}\right), \quad (10.3)$$

where the parameter  $\nu$  is the average value for the quantity being approximated:  $\nu = \bar{r}(a)$  for the reward and  $\nu = \bar{c}_{i,t}$  for the consumption of resource  $i$ . This version features improved scaling with  $n = n_t(a)$ : indeed, it is  $1/\sqrt{n}$  in the worst case, and essentially  $1/n$  when  $\nu$  is very small. The analysis relies on a technical lemma that (10.3) indeed defines a confidence radius, in the sense that the clean event happens with high probability.

## 10.6 Three algorithms for BwK (no proofs)

Three different algorithms have been designed for BwK, all achieving the optimal regret bound (10.1). All three algorithms build on the common foundation described above, but then proceed via very different techniques. In the remainder we describe these algorithms and the associated intuition; the analysis of any one of these algorithms is both too complicated and too long for this lecture.

We present the first two algorithms in detail (albeit with some simplification for ease of presentation), and only give a rough intuition for the third one.

**Algorithm I: balanced exploration.** This algorithm can be seen as an extension of Successive Elimination. Recall that in Successive Elimination, we start with all arms being “active” and permanently de-activate a given arm  $a$  once we have high-confidence evidence that some other arm is better. The idea is that each arm that is currently active can potentially be an optimal arm given the evidence collected so far. In each round we choose among arms that are still “potentially optimal”, which suffices for the purpose of exploitation. And choosing *uniformly* (or round-robin) among the potentially optimal arms suffices for the purpose of exploration.

In BwK, we look for optimal *distributions* over arms. Each distribution  $\mathcal{D}$  is called *potentially optimal* if it optimizes  $\text{FR}(\mathcal{D})$  for some latent structure  $\mu$  in the current confidence region  $\text{ConfRegion}_t$ . In each round, we choose a potentially optimal distribution, which suffices for exploitation. But *which* potentially optimal distribution to choose so as to ensure sufficient exploration? Intuitively, we would like to explore each arm as much as possible, given the constraint that we can only use potentially optimal distributions. So, we settle for something almost as good: we choose an arm uniformly at random, and then explore it as much as possible, see Algorithm 12.

[as: TODO: better formatting for the pseudocode ]

- 1 In each round  $t$ ,
- $S_t \leftarrow$  the set of all potentially optimal distributions over arms.
  - Pick arm  $b_t$  uniformly at random, and pick a distribution  $\mathcal{D} = \mathcal{D}_t$  over arms so as to maximize  $\mathcal{D}(b_t)$ , the probability of choosing arm  $b_t$ , among all potentially optimal distributions  $\mathcal{D}$ .
  - pick arm  $a_t \sim \mathcal{D}$ .

**Algorithm 12:** Balanced exploration

While this algorithm is well-defined as a mapping from histories to action, we do not provide an efficient implementation for the general case of BwK.

**Algorithm II: optimism under uncertainty.** For each latent structure  $\mu$  and each distribution  $\mathcal{D}$  we have a fractional value  $\text{FR}(\mathcal{D}|\mu)$  determined by  $\mathcal{D}$  and  $\mu$ . Using confidence region  $\text{ConfRegion}_t$ , we can define the Upper Confidence Bound for  $\text{FR}(\mathcal{D})$ :

$$\text{UCB}_t(\mathcal{D}) = \sup_{\mu \in \text{ConfRegion}_t} \text{FR}(\mathcal{D}|\mu). \quad (10.4)$$

In each round, the algorithm picks distribution  $\mathcal{D}$  with the highest UCB. An additional trick is to pretend that all budgets are scaled down by the same factor  $1 - \epsilon$ , for an appropriately chosen parameter  $\epsilon$ , and redefine  $\text{FR}(\mathcal{D}|\mu)$  accordingly. Thus, the algorithm is as follows:

- Rescale the budgets:  $B_i \leftarrow (1 - \epsilon) \times B_i$  for each resource  $i$
- In each round  $t$ , pick distribution  $\mathcal{D} = \mathcal{D}_t$  with highest  $\text{UCB}_t(\mathcal{D})$
- pick arm  $a_t \sim \mathcal{D}$ .

**Algorithm 13:** UCB for BwK

The rescaling trick is essential: it ensures that we do not run out of resources too soon due to randomness in the outcomes or to the fact that the distributions  $\mathcal{D}_t$  do not quite achieve the optimal value for  $\text{FR}(\mathcal{D})$ .

Choosing a distribution with maximal UCB can be implemented by a linear program. Since the confidence region is a product set, it is easy to specify the latent structure  $\mu \in \text{ConfRegion}_t$  which attains the supremum in (10.4). Indeed, re-write the definition of  $\text{FR}(\mathcal{D})$  more explicitly:

$$\text{FR}(\mathcal{D}) = \left( \sum_a \mathcal{D}(a) r(a) \right) \left( \min_{\text{resources } i} \frac{B_i}{\sum_a \mathcal{D}(a) c_i(a)} \right).$$

Then  $\text{UCB}_t(\mathcal{D})$  is obtained by replacing the expected reward  $r(a)$  of each arm  $a$  with the corresponding upper confidence bound, and the expected resource consumption  $c_i(a)$  with the corresponding lower confidence bound. Denote the resulting UCB on  $r(\mathcal{D})$  with  $r^{\text{UCB}}(\mathcal{D})$ , and the resulting LCB on  $c_i(\mathcal{D})$  with  $c_i^{\text{LCB}}(\mathcal{D})$ . Then the linear program is:

$$\begin{aligned} & \text{maximize} && \tau \times r^{\text{UCB}}(\mathcal{D}) \\ & \text{subject to} && \tau \times c_i^{\text{LCB}}(\mathcal{D}) \leq B_i(1 - \epsilon) \\ & && \tau \leq T \\ & && \sum_a \mathcal{D}(a) = 1. \end{aligned}$$

**Algorithm III: resource costs and Hedge.** The key idea is to pretend that instead of budgets on resources we have *costs* for using them. That is, we pretend that each resource  $i$  can be used at cost  $v_i^*$  per unit. The costs  $v_i^*$  have a mathematical definition in terms of the latent structure  $\mu$  (namely, they arise as the dual solution of a certain linear program), but they are not known to the algorithm. Then we can define the “resource utilization cost” of each arm  $a$  as

$$v^*(a) = \sum_i v_i^* c_i(a).$$

We want to pick an arm which generates more reward at less cost. One natural way to formalize this intuition is to seek an arm which maximizes the *bang-per-buck ratio*

$$\Lambda(a) = r(a)/v^*(a).$$

The trouble is, we do not know  $r(a)$ , and we *really* do not know  $v^*(a)$ .

Instead, we approximate the bang-per-buck ratio  $\Lambda(a)$  using the principle of optimism under uncertainty. As usual, in each round  $t$  we have upper confidence bound  $U_t(a)$  on the expected reward  $r(a)$ , and lower confidence bound  $L_{t,i}(a)$  on the expected consumption  $c_i(a)$  of each resource  $i$ . Then, given our current estimates  $v_{t,i}$  for the resource costs  $v_i^*$ , we can optimistically estimate  $\Lambda(a)$  with

$$\Lambda_t^{\text{UCB}}(a) = r(a) / \sum_i v_{t,i} L_{t,i}(a),$$

and choose an arm  $a = a_t$  which maximizes  $\Lambda_t^{\text{UCB}}(a)$ . The tricky part is to maintain meaningful estimates  $v_{t,i}$ . Long story short, they are maintained using a version of Hedge.

One benefit of this algorithm compared to the other two is that the final pseudocode is very simple and *elementary*, in the sense that it does not invoke any subroutine such as a linear program solver. Also, the algorithm happens to be extremely fast computationally.

[as: **TODO: insert pseudocode, add a bit more intuition**]

## 10.7 Bibliographic notes

The general setting of BwK is introduced in Badanidiyuru et al. (2013), along with the first and third algorithms and the lower bound. The UCB-based algorithm is from Agrawal and Devanur (2014). A more thorough discussion of the motivational examples, as well as an up-to-date discussion of related work, can be found in Badanidiyuru et al. (2013).



## Chapter 11

# Bandits and Incentives (lecture to be written up)

[as: To be added. For now, see hand-written lecture notes at  
<http://www.cs.umd.edu/~slivkins/CMSC858G-fall16/myLN-L12.pdf>]



## Chapter 12

# Bandits and Games (lecture to be written up)

[as: To be added. For now, see hand-written lecture notes at  
<http://www.cs.umd.edu/~slivkins/CMSC858G-fall16/myLN-L13.pdf>]





## **Chapter 13**

### **MDP bandits and Gittins Algorithm (to be written)**



# Bibliography

- Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert Schapire. Taming the monster: A fast and simple algorithm for contextual bandits. In *31st Intl. Conf. on Machine Learning (ICML)*, 2014.
- Alekh Agarwal, Sarah Bird, Markus Cozowicz, Miro Dudik, John Langford, Lihong Li, Luong Hoang, Dan Melamed, Siddhartha Sen, Robert Schapire, and Alex Slivkins. Multiworld testing: A system for experimentation, learning, and decision-making, 2016. A white paper, available at <https://github.com/Microsoft/mwt-ds/raw/master/images/MWT-WhitePaper.pdf>.
- Shipra Agrawal and Nikhil R. Devanur. Bandits with concave rewards and convex knapsacks. In *15th ACM Conf. on Economics and Computation (ACM EC)*, 2014.
- J.Y. Audibert and S. Bubeck. Regret Bounds and Minimax Policies under Partial Monitoring. *J. of Machine Learning Research (JMLR)*, 11:2785–2836, 2010.
- Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002a.
- Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, 2002b. Preliminary version in *36th IEEE FOCS*, 1995.
- Baruch Awerbuch and Robert Kleinberg. Online linear optimization and adaptive routing. *J. of Computer and System Sciences*, 74(1):97–114, February 2008. Preliminary version in *36th ACM STOC*, 2004.
- Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. In *54th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2013.
- Dirk Bergemann and Juuso Välimäki. Bandit Problems. In Steven Durlauf and Larry Blume, editors, *The New Palgrave Dictionary of Economics*, 2nd ed. Macmillan Press, 2006.
- Donald Berry and Bert Fristedt. *Bandit problems: sequential allocation of experiments*. Chapman&Hall, 1985.
- Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends in Machine Learning*, 5(1), 2012.
- Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: stochastic and adversarial bandits. In *25th Conf. on Learning Theory (COLT)*, 2012.
- Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvari. Online Optimization in X-Armed Bandits. *J. of Machine Learning Research (JMLR)*, 12:1587–1627, 2011.
- Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge Univ. Press, 2006.
- Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual Bandits with Linear Payoff Functions. In *14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.
- Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.
- Ofer Dekel, Ambuj Tewari, and Raman Arora. Online bandit learning against an adaptive adversary: from regret to policy regret. In *29th Intl. Conf. on Machine Learning (ICML)*, 2012.

- Eyal Even-Dar, Shie Mannor, and Yishay Mansour. PAC bounds for multi-armed bandit and Markov decision processes. In *15th Conf. on Learning Theory (COLT)*, pages 255–270, 2002.
- John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, 2011.
- Elad Hazan. Introduction to Online Convex Optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2015.
- Elad Hazan and Satyen Kale. Better algorithms for benign bandits. In *20th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 38–47, 2009.
- Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison Wesley, 2005.
- Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. In *18th Advances in Neural Information Processing Systems (NIPS)*, 2004.
- Robert Kleinberg. *CS683: Learning, Games, and Electronic Markets*, a class at Cornell University. Lecture notes, available at <http://www.cs.cornell.edu/courses/cs683/2007sp/>, Spring 2007.
- Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *40th ACM Symp. on Theory of Computing (STOC)*, pages 681–690, 2008.
- Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Bandits and experts in metric spaces. Working paper, published at <http://arxiv.org/abs/1312.1277>, 2015. Merged and revised version of conference papers in *ACM STOC 2008* and *ACM-SIAM SODA 2010*.
- Tze Leung Lai and Herbert Robbins. Asymptotically efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6: 4–22, 1985.
- John Langford and Tong Zhang. The Epoch-Greedy Algorithm for Contextual Multi-armed Bandits. In *21st Advances in Neural Information Processing Systems (NIPS)*, 2007.
- Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *19th Intl. World Wide Web Conf. (WWW)*, 2010.
- Alexander Rakhlin and Karthik Sridharan. BISTRO: an efficient relaxation-based method for contextual bandits. In *33rd Intl. Conf. on Machine Learning (ICML)*, 2016.
- Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *31st Intl. Conf. on Machine Learning (ICML)*, 2014.
- Aleksandrs Slivkins. Contextual bandits with similarity information. *J. of Machine Learning Research (JMLR)*, 15(1):2533–2568, 2014. Preliminary version in *COLT 2011*.
- Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the Brownian restless bandits. In *21st Conf. on Learning Theory (COLT)*, pages 343–354, 2008.
- Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E. Schapire. Efficient algorithms for adversarial contextual learning. In *33rd Intl. Conf. on Machine Learning (ICML)*, 2016a.
- Vasilis Syrgkanis, Haipeng Luo, Akshay Krishnamurthy, and Robert E. Schapire. Improved regret bounds for oracle-based adversarial contextual bandits. In *29th Advances in Neural Information Processing Systems (NIPS)*, 2016b.
- Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010.
- William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285294, 1933.