Introduction to Multi-Armed Bandits

(preliminary and incomplete draft)

Aleksandrs Slivkins

Microsoft Research NYC

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

First draft: January 2017

This version: September 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

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

[author: chapter revised September 2017, seems pretty polished. In the next round of revision, I will probably add a few paragraphs on "practical aspects", and some more citations.]

This chapter covers bandits with i.i.d rewards, the basic model of multi-arm bandits. We present several algorithms, and analyze their performance in terms of regret. The ideas introduced in this chapter extend far beyond the basic model, and will resurface throughout the book.

2.1 Model and examples

Problem formulation (Bandits with i.i.d. rewards). There is a fixed and finite set of *actions*, a.k.a. *arms*, denoted $\mathcal{A}$. Learning proceeds in rounds, indexed by $t = 1, 2 \ldots$. The number of rounds T, a.k.a. the *time horizon*, is fixed and known in advance. The protocol is as follows:

Multi-armed bandits

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

1. Algorithm picks arm $a_t \in \mathcal{A}$.
2. Algorithm observes reward $r_t \in [0, 1]$ for the chosen arm.

The algorithm observes only the reward for the selected action, and nothing else. In particular, it does not observe rewards for other actions that could have been selected. Such feedback model is called *bandit feedback*.

Per-round rewards are bounded; the restriction to the interval $[0, 1]$ is for simplicity. The algorithm's goal is to maximize total reward over all T rounds.

We make the *i.i.d. assumption*: the reward for each action is i.i.d (independent and identically distributed). More precisely, for each action a, there is a distribution $\mathcal{D}_a$ over reals, called the *reward distribution*. Every time this action is chosen, the reward is sampled independently from this distribution. $\mathcal{D}_a$ is initially unknown to the algorithm.

Perhaps the simplest reward distribution is the Bernoulli distribution, when the reward of each arm a can be either 1 or 0 ("success or failure", "heads or tails"). This reward distribution is fully specified by the mean reward, which in this case is simply the probability of the successful outcome. The problem instance is then fully specified by the time horizon T and the mean rewards.

Our model is a simple abstraction for an essential feature of reality that is present in many application scenarios. We proceed with three motivating examples:

1. **News**: in a very stylized news application, a user visits a news site, the site presents it with a news header, and a user either clicks on this header or not. The goal of the website is to maximize the number of clicks. So each possible header is an arm in a bandit problem, and clicks are the rewards. Note that 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. **Ad selection**: In website advertising, a user visits a webpage, and a learning algorithm selects one of many possible ads to display. If ad a is displayed, the website observes whether the user clicks on the ad, in which case the advertiser pays some amount $v_a \in [0,1]$. So each ad is an arm, and the paid amount is the reward.

   A typical modeling assumption is that the paid amount v_a depends only on the displayed ad, but does not change over time. The click probability for a given ad does not change over time, either.

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. As an idealized assumption, each patient is drawn independently from a fixed distribution over patients, so the effectiveness of a given treatment is i.i.d.

Note that the reward of a given arm can only take two possible values in the first two examples, but could, in principle, take arbitrary values in the third example.

Notation. We use the following conventions in this chapter and (usually) throughout the book. Actions are denoted with a, rounds with t. The number of arms is K, the number of rounds is T. The mean reward of arm a is $\mu(a) := \mathbb{E}[\mathcal{D}_a]$. The best mean reward is denoted $\mu^* := \max_{a \in \mathcal{A}} \mu(a)$. The difference $\Delta(a) := \mu^* - \mu(a)$ describes how bad arm a is compared to μ^*; we call it the *badness* of arm a. An optimal arm is an arm a with $\mu(a) = \mu^*$; note that it is not necessarily unique. We take a^* to denote an optimal arm.

Regret. How do we argue whether an algorithm is doing a good job across different problem instances, when some instances inherently allow higher rewards than others? One standard approach is to compare the algorithm to the best one could possibly achieve on a given problem instance, if one knew the mean rewards. More formally, we consider the first t rounds, and compare the cumulative mean reward of the algorithm against $\mu^* \cdot t$, the expected reward of always playing an optimal arm:

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

This quantity is called *regret* at round t.[1] The quantity $\mu^* \cdot t$ is sometimes called the *best arm benchmark*.

Note that a_t (the arm chosen at round t) is a random quantity, as it may depend on randomness in rewards and/or the algorithm. So, regret $R(t)$ is also a random variable. Hence we will typically talk about expected regret $\mathbb{E}[R(T)]$.

We mainly care about the dependence of regret on the round t and the time horizon T. We also consider the dependence on the number of arms K and the mean rewards μ. We are less interested in the fine-grained dependence on the reward distributions (beyond the mean rewards). We will usually use big-O notation to focus on the asymptotic dependence on the parameters of interests, rather than keep track of the constants.

Remark 2.1 (Terminology). Since our definition of regret sums over all rounds, we sometimes call it *cumulative* regret. When/if we need to highlight the distinction between $R(T)$ and $\mathbb{E}[R(T)]$, we say *realized regret* and *expected regret*; but most of the time we just say "regret" and the meaning is clear from the context. The quantity $\mathbb{E}[R(T)]$ is sometimes called *pseudo-regret* in the literature.

2.2 Simple algorithms: uniform exploration

We start with a simple idea: explore arms uniformly (at the same rate), regardless of what has been observed previously, and pick an empirically best arm for exploitation. A natural incarnation of this idea, known as *Explore-first* algorithm, is to dedicate an initial segment of rounds to exploration, and the remaining rounds to exploitation.

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

Algorithm 1: Explore-First with parameter N.

[1] It is called called "regret" because this is how much the algorithm "regrets" not knowing what is the best arm.

The parameter N is fixed in advance; it will be chosen later as function of the time horizon T and the number of arms K, so as to minimize regret. Let us analyze regret of this algorithm.

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 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:

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

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 2.2. 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 by a proof that argues about probabilities more carefully.

For simplicity, let us start with the case of $K = 2$ arms. Consider 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 the algorithm chooses the other arm $a \neq a^*$. 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

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

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. We derive an upper bound on the regret, which consists of two parts: for the first N rounds of exploration, and then for the remaining $T - 2N$ rounds of exploitation:

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

Recall that we can select any value for N, as long as it is known to the algorithm before the first round. So, we can choose N so as to (approximately) minimize the right-hand side. Noting that the two summands are, resp., monotonically increasing and monotonically decreasing in N, we set N so that they are (approximately) equal. For $N = T^{2/3} (\log T)^{1/3}$, we obtain:

$$R(T) \leq O\left(T^{2/3} (\log T)^{1/3}\right).$$

To complete the proof, we have to analyze the case of the "bad event". 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,

$$\begin{aligned} \mathbb{E}[R(T)] &= \mathbb{E}[R(T)|\text{clean event}] \times \Pr[\text{clean event}] \;+\; \mathbb{E}[R(T)|\text{bad event}] \times \Pr[\text{bad event}] \\ &\leq \mathbb{E}[R(T)|\text{clean event}] + T \times O(T^{-4}) \\ &\leq O(\sqrt{\log T} \times T^{2/3}). \end{aligned} \tag{2.3}$$

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, regret accumulated in exploration phase is now upper-bounded by KN. Working through the proof, we obtain $R(T) \leq NK + O(\sqrt{\frac{\log T}{N}} \times T)$. As before, we approximately minimize it by approximately minimizing the two summands. Specifically, we plug in $N = (T/K)^{2/3} \cdot O(\log T)^{1/3}$. Completing the proof same way as in (2.3), we obtain:

Theorem 2.3. *Explore-first achieves regret $\mathbb{E}[R(T)] \leq T^{2/3} \times O(K \log T)^{1/3}$, where K is the number of arms.*

One problem with Explore-first is that its performance in the exploration phase is just *terrible*. It is usually better to spread exploration more uniformly over time. This is done in the *epsilon-greedy* algorithm:

```
for each round t = 1, 2, ... do
    Toss a coin with success probability ε_t;
    if success then
        explore: choose an arm uniformly at random
    else
        exploit: choose the arm with the highest average reward so far
end
```

Algorithm 2: Epsilon-Greedy with exploration probabilities $(\epsilon_1, \epsilon_2, \ldots)$.

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 exploration is uniform over arms, which is similar to the "round-robin" exploration in the explore-first algorithm. Since exploration is now spread uniformly over time, one can hope to derive meaningful regret bounds even for small t. We focus on exploration probability $\epsilon_t = t^{-1/3}$, so that the expected number of exploration rounds up to round t is $\Theta(t^{2/3})$, same as in explore-first with time horizon $T = t$. We derive the same regret bound as in Theorem 2.3, but now it holds for all rounds t. The proof relies on a more refined clean event which we introduce in the next section, and is left as an exercise (see Exercise 2.2).

Theorem 2.4. *Epsilon-greedy algorithm with exploration probabilities $\epsilon_t = t^{-1/3}$ achieves regret $\mathbb{E}[R(t)] \leq t^{2/3} \times O(K \log t)^{1/3}$ for each round t.*

2.3 Advanced algorithms: adaptive exploration

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 chapter we present two algorithms that implement adaptive exploration and achieve better regret.

Let's start with the case of $K = 2$ arms. One 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 us do some probability with our old friend Hoeffding Inequality.

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

$$\Pr\left(|\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)\right) \geq 1 - \tfrac{2}{T^4}, \tag{2.4}$$

where $r_t(a) = \sqrt{\frac{2 \log T}{n_t(a)}}$ is the confidence radius, and T is the time horizon. Note that we have $n_t(a)$ independent random variables — one per each sample of arm a. Since Hoeffding Inequality requires a fixed number of random

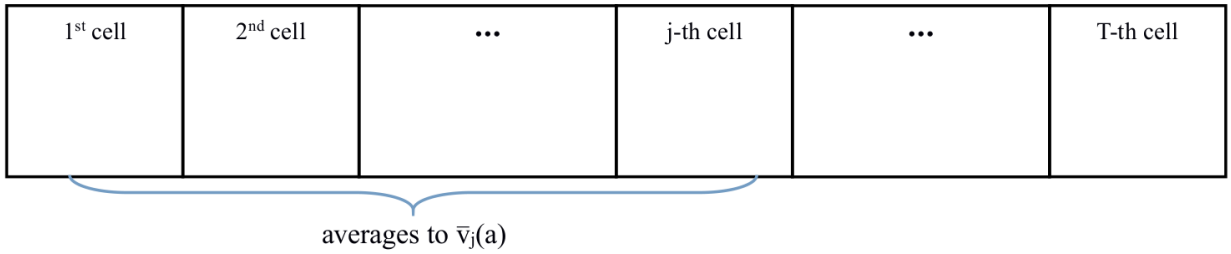


Figure 2.1: the j-th cell contains the reward of the j-th time we pull arm a, *i.e.,* reward of arm a when $n_t(a) = j$

variables, (2.4) would follow immediately if $n_t(a)$ were fixed in advance. However, $n_t(a)$ is itself is 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. Without loss of generality, this table encodes rewards as follows: the j-th time a given arm a is chosen by the algorithm, its reward is taken from the j-th cell in this arm's tape. Let $\bar{v}_j(a)$ represent the average reward at arm a from first j times that arm a is chosen. Now one can use Hoeffding Inequality to derive that

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

Taking a union bound, it follows that (assuming $K = \#\text{arms} \leq T$)

$$\Pr\left(\forall a \forall j \quad |\bar{v}_j(a) - \mu(a)| \leq r_t(a)\right) \geq 1 - \tfrac{2}{T^2}. \tag{2.5}$$

Now, observe that the event in (2.5) implies the event

$$\mathcal{E} := \{\forall a \forall t \quad |\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)\} \tag{2.6}$$

which we are interested in. Therefore, we have proved:

Lemma 2.5. $\Pr[\mathcal{E}] \geq 1 - \frac{2}{T^2}$, *where $\mathcal{E}$ is given by (2.6).*

The event in (2.6) will be the *clean event* for the subsequent analysis.

Motivated by this lemma, we define *upper/lower confidence bounds* (for arm a at round t):

$$\texttt{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a),$$
$$\texttt{LCB}_t(a) = \bar{\mu}_t(a) - r_t(a).$$

The interval $[\texttt{LCB}_t(a); \texttt{UCB}_t(a)]$ is called the *confidence interval*.

2.3.2 Successive Elimination algorithm

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

1 Alternate two arms until $\texttt{UCB}_t(a) < \texttt{LCB}_t(a')$ after some even round t;
2 Then abandon arm a, and use arm a' forever since.

Algorithm 3: "High-confidence elimination" algorithm for two arms

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 2.2). Then

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

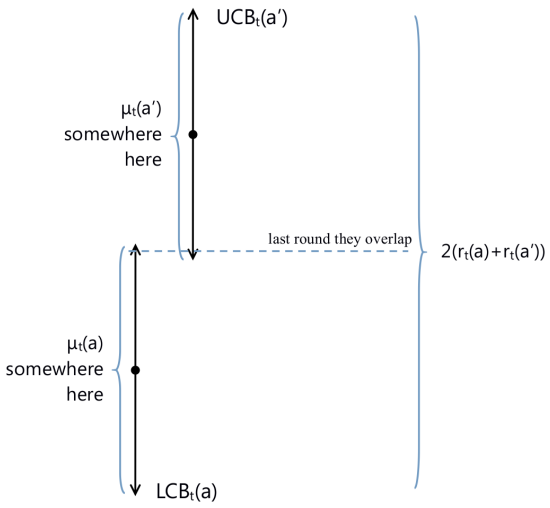


Figure 2.2: t is the last round that the two confidence intervals still overlap

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

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

Since we've chosen the best arm from then on, we have $R(t) \leq O(\sqrt{t \log T})$. To complete the analysis, we need to argue that the "bad event" $\bar{\mathcal{E}}$ contributes a negligible amount to regret, much like we did for Explore-first:

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

We proved the following:

Lemma 2.6. *For two arms, Algorithm 3 achieves regret $\mathbb{E}[R(t)] \leq O(\sqrt{t \log T})$ for each round $t \leq T$.*

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

This approach extends to $K > 2$ arms as follows:

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 $\texttt{UCB}_t(a) < \texttt{LCB}_t(a')$;
5 Repeat until end of rounds.

Algorithm 4: Successive Elimination algorithm

To analyze the performance of this algorithm, it suffices to focus on the clean event (2.6); 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:

$$\Delta(a) \leq O(r_T(a)) = O\left(\sqrt{\frac{\log T}{n_T(a)}}\right) \quad \text{for each arm } a \text{ with } \mu(a) < \mu(a^*). \tag{2.7}$$

Informally: an arm played many times cannot be too bad. The rest of the analysis only relies on (2.7). In other words, it does not matter which algorithm achieves this property.

The contribution of arm a to regret at round t, denoted $R(t;a)$, can be expressed as $\Delta(a)$ for each round this arm is played; by (2.7) we can bound this quantity as

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

Recall that $\mathcal{A}$ denotes 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. Then:

$$R(t) = \sum_{a \in \mathcal{A}^+} R(t;a) = O(\sqrt{\log T}) \sum_{a \in \mathcal{A}^+} \sqrt{n_t(a)} \leq O(\sqrt{\log T}) \sum_{a \in \mathcal{A}} \sqrt{n_t(a)}. \tag{2.8}$$

Since $f(x) = \sqrt{x}$ is a real concave function, and $\sum_{a \in \mathcal{A}} n_t(a) = t$, by Jensen's Inequality (Theorem A.1) 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 (2.8), we see that $R(t) \leq O(\sqrt{Kt \log T})$. Thus, we have proved:

Theorem 2.8. *Successive Elimination algorithm achieves regret*

$$\mathbb{E}[R(t)] = O(\sqrt{Kt \log T}) \quad \textit{for all rounds } t \leq T. \tag{2.9}$$

We can also use property (2.7) to obtain another regret bound. Rearranging the terms in (2.7), we obtain $n_T(a) \leq O\left(\frac{\log T}{[\Delta(a)]^2}\right)$. Informally: a bad arm cannot be played too many times. Therefore, for each arm $a \in \mathcal{A}^+$ we have:

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

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 2.9. *Successive Elimination algorithm achieves regret*

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

Remark 2.10. This regret bound is logarithmic in T, with a constant that can be arbitrarily large depending on a problem instance. The distinction between regret bounds achievable with an absolute constant (as in Theorem 2.8) and regret bounds achievable with an instance-dependent constant is typical for multi-armed bandit problems. The existence of logarithmic regret bounds is another benefit of adaptive exploration compared to non-adaptive exploration.

Remark 2.11. For a more formal terminology, consider a regret bound of the form $C \cdot f(T)$, where $f(\cdot)$ does not depend on the mean rewards μ, and the "constant" C does not depend on T. Such regret bound is called *instance-independent* if C does not depend on μ, and *instance-dependent* otherwise.

Remark 2.12. It is instructive to derive Theorem 2.8 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 ϵ per round, for a total of ϵT;
- 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 ϵ 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, known as *optimism under uncertainty*: 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 $\operatorname{argmax}_{a \in \mathcal{A}} \texttt{UCB}_t(a)$, where $\texttt{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a)$;

Algorithm 5: UCB1 Algorithm

Remark 2.13. Let's see why UCB-based selection rule makes sense. An arm a is chosen in round t because it has a large $\texttt{UCB}_t(a)$, which can happen for two reasons: because the average reward $\bar{\mu}_t(a)$ is large, in which case this arm is likely to have a high reward, and/or because the confidence radius $r_t(a)$ is large, in which case this arm has not been explored much. Either reason makes this arm worth choosing. Further, the $\bar{\mu}_t(a)$ and $r_t(a)$ summands in the UCB represent exploitation and exploration, respectively, and summing them up is a natural way to trade off the two.

To analyze this algorithm, let us focus on the clean event (2.6), as before. Recall that a^* be an optimal arm, and a_t is the arm chosen by the algorithm in round t. According to the algorithm, $\texttt{UCB}_t(a_t) \geq \texttt{UCB}_t(a^*)$. Under the clean event, $\mu(a_t) + r_t(a_t) \geq \bar{\mu}_t(a_t)$ and $\texttt{UCB}_t(a^*) \geq \mu(a^*)$. Therefore:

$$\mu(a_t) + 2r_t(a_t) \geq \bar{\mu}_t(a_t) + r_t(a_t) = \texttt{UCB}_t(a_t) \geq \texttt{UCB}_t(a^*) \geq \mu(a^*). \tag{2.12}$$

It follows that

$$\Delta(a_t) := \mu(a^*) - \mu(a_t) \leq 2r_t(a_t) = 2\sqrt{\frac{2 \log T}{n_t(a_t)}}. \tag{2.13}$$

Remark 2.14. This cute trick resurfaces in the analyses of several UCB-like algorithms for more general settings.

For each arm a consider the last round t when this arm is chosen by the algorithm. Applying (2.13) to this round gives us property (2.7). The rest of the analysis follows from that property, as in the analysis of Successive Elimination.

Theorem 2.15. *Algorithm UCB1 satisfies regret bounds in (2.9) and (2.11).*

2.4 Further remarks

Techniques. This chapter introduces several techniques that are broadly useful in multi-armed bandits, beyond the specific setting discussed in this chapter: the four algorithmic techniques (Explore-first, Epsilon-greedy, Successive Elimination, and UCB-based arm selection), 'clean event' technique in the analysis, and the "UCB trick" from (2.12).

Main references. Successive Elimination is from Even-Dar et al. (2002), and UCB1 is from Auer et al. (2002a). Explore-first and Epsilon-greedy algorithms have been known for a long time, the author is not aware of the original references.

High-probability regret. In order to upper-bound expected regret $\mathbb{E}[R(T)]$, we actually obtained a high-probability upper bound on $R(T)$ itself. This is common for regret bounds obtained via the "clean event" technique, but not to be taken for granted in general.

Regret for all rounds at once. What if the time horizon T is not known in advance? Can we achieve similar regret bounds that hold for all rounds t, not just for all $t \leq T$? Recall that in Successive Elimination and UCB1, knowing T was needed only to define the confidence radius $r_t(a)$. There are several remedies:

- If an upper bound on T is known, one can use it instead of T in the algorithm. Since our regret bounds depend on T only logarithmically, rather significant over-estimates can be tolerated.
- One can use UCB1 with confidence radius $r_t(a) = \sqrt{\frac{2\log t}{n_t(a)}}$. This version achieves the same regret bounds, and with better constants, at the cost of a somewhat more complicated analysis.[2]
- Any algorithm for known time horizon can be converted to an algorithm for an arbitrary time horizon using the *doubling trick*. Here, the new algorithm proceeds in phases of exponential duration. Each phase $i = 1, 2, \ldots$ lasts 2^i rounds, and executes a fresh run of the original algorithm. This approach achieves the "right" theoretical guarantees (see Exercise 2.5). However, forgetting everything after each phase is not very practical.

Instantaneous regret. An alternative notion of regret considers each round separately: *instantaneous regret* at round t (also called *simple regret*) is defined as $\Delta(a_t) = \mu^* - \mu(a_t)$, where a_t is the arm chosen in this round. In addition to having low cumulative regret, it may be desirable to spread the regret more "uniformly" over rounds, so as to avoid spikes in instantaneous regret. Then one would also like an upper bound on instantaneous regret that decreases monotonically over time.

Bandits with predictions. While the standard goal for bandit algorithms is 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. In particular, 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 this in the exercises). However, improvements are possible in some regimes (*e.g.,* Mannor and Tsitsiklis, 2004; Even-Dar et al., 2006; Bubeck et al., 2011a; Audibert et al., 2010).

2.5 Exercises

All exercises below, except Exercise 2.3(b), are fairly straightforward given the material in this chapter.

Exercise 2.1 (rewards from a small interval). Consider a version of the problem in which all the realized rewards are in the interval $[\frac{1}{2}, \frac{1}{2} + \epsilon]$ for some $\epsilon \in (0, \frac{1}{2})$. Define versions of UCB1 and Successive Elimination attain improved regret bounds (both logarithmic and root-T) that depend on the ϵ.

Hint: Use a version of Hoeffding Inequality with ranges.

[2] This is, in fact, the original treatment of UCB1 from (Auer et al., 2002a).

Exercise 2.2 (Epsilon-greedy). Prove Theorem 2.4: derive the $\tilde{O}(t^{2/3})$ regret bound for the epsilon-greedy algorithm exploration probabilities $\epsilon_t = t^{-1/3}$.

Hint: Fix round t and analyze $\mathbb{E}[\Delta(a_t)]$ for this round separately. Set up the "clean event" for rounds $1, \ldots, t$ much like in Section 2.3.1 (treating t as the time horizon), but also include the number of exploration rounds up to time t.

Exercise 2.3 (instantaneous regret). Recall that instantaneous regret at round t is $\Delta(a_t) = \mu^* - \mu(a_t)$.

(a) Prove that Successive Elimination achieves "instance-independent" regret bound of the form

$$\mathbb{E}[\Delta(a_t)] \leq \frac{\text{polylog}(T)}{\sqrt{t/K}} \quad \text{for each round } t \in [T]. \tag{2.14}$$

(b) Let us argue that UCB1 does not achieve the regret bound in (2.14). More precisely, let us consider a version of UCB1 with $\texttt{UCB}_t(a) = \bar{\mu}_t(a) + 2 \cdot r_t(a)$. (It is easy to see that our analysis carries over to this version.) Focus on two arms, and prove that this algorithm cannot achieve a regret bound of the form

$$\mathbb{E}[\Delta(a_t)] \leq \frac{\text{polylog}(T)}{t^\gamma}, \ \gamma > 0 \quad \text{for each round } t \in [T]. \tag{2.15}$$

Hint: Fix mean rewards and focus on the clean event. If (2.15) holds, then the bad arm cannot be played after some time T_0. Consider the last time the bad arm is played, call it $t_0 \leq T_0$. Derive a lower bound on the UCB of the best arm at t_0 (stronger lower bound than the one proved in class). Consider what this lower implies for the UCB of the bad arm at time t_0. Observe that eventually, after some number of plays of the best arm, the bad arm will be chosen again, assuming a large enough time horizon T. Derive a contradiction with (2.15).

Take-away: for "bandits with predictions", the simple solution of predicting the last-played arm to be the best arm does not always work, even for a good algorithm such as UCB1.

(c) Derive a regret bound for Explore-first: an "instance-independent" upper bound on instantaneous regret.

Exercise 2.4 (bandits with predictions). Recall that in "bandits with predictions", after T rounds the algorithm outputs a prediction: a guess y_T for the best arm. We focus on the instantaneous regret $\Delta(y_T)$ for the prediction.

(a) Take any bandit algorithm with an instance-independent regret bound $E[R(T)] \leq f(T)$, and construct an algorithm for "bandits with predictions" such that $\mathbb{E}[\Delta(y_T)] \leq f(T)/T$.

Note: Surprisingly, taking $y_T = a_t$ does not work in general, see Exercise 2.3(b). Taking y_T to be the arm with a maximal reward does not seem to work, either. But there is a simple solution ...

Take-away: We can easily obtain $\mathbb{E}[\Delta(y_T)] = O(\sqrt{K \log(T)/T}$ from standard algorithms such as UCB1 and Successive Elimination. However, as parts (bc) show, one can do much better!

(b) Consider Successive Elimination with $y_T = a_T$. Prove that (with a slightly modified definition of the confidence radius) this algorithm can achieve

$$\mathbb{E}[\Delta(y_T)] \leq T^{-\gamma} \quad \text{if } T > T_{\mu,\gamma},$$

where $T_{\mu,\gamma}$ depends only on the mean rewards $\mu(a) : a \in \mathcal{A}$ and the γ. This holds for an arbitrarily large constant γ, with only a multiplicative-constant increase in regret.

Hint: Put the γ inside the confidence radius, so as to make the "failure probability" sufficiently low.

(c) Prove that alternating the arms (and predicting the best one) achieves, for any fixed $\gamma < 1$:

$$\mathbb{E}[\Delta(y_T)] \leq e^{-\Omega(T^\gamma)} \quad \text{if } T > T_{\mu,\gamma},$$

where $T_{\mu,\gamma}$ depends only on the mean rewards $\mu(a) : a \in \mathcal{A}$ and the γ.

Hint: Consider Hoeffding Inequality with an arbitrary constant α in the confidence radius. Pick α as a function of the time horizon T so that the failure probability is as small as needed.

Exercise 2.5 (Doubling trick). Take any bandit algorithm $\mathcal{A}$ for fixed time horizon T. Convert it to an algorithm $\mathcal{A}_\infty$ which runs forever, in phases $i = 1, 2, 3, ...$ of 2^i rounds each. In each phase i algorithm $\mathcal{A}$ is restarted and run with time horizon 2^i.

(a) State and prove a theorem which converts an instance-independent upper bound on regret for $\mathcal{A}$ into similar bound for $\mathcal{A}_\infty$ (so that this theorem applies to both UCB1 and Explore-first).

(b) Do the same for $\log(T)$ instance-dependent upper bounds on regret. (Then regret increases by a $\log(T)$ factor.)

Chapter 3

Lower Bounds

[author: chapter revised September 2017, seems pretty polished. Would be nice to add another exercise or two.]

This chapter is about *negative* results: about what bandit algorithms *cannot* do. We are interested in lower bounds on regret which apply to all bandit algorithms. In other words, we want to prove that no bandit algorithm can achieve regret better than this lower bound. We prove the $\Omega(\sqrt{KT})$ lower bound, which takes most of this chapter, and state an instance-dependent $\Omega(\log T)$ lower bound without a proof. These lower bounds give us a sense of what are the best possible *upper* bounds that we can hope to prove. The $\Omega(\sqrt{KT})$ lower bound is stated as follows:

Theorem 3.1. *Fix time horizon T and the number of arms K. For any bandit algorithm, there exists a problem instance such that $\mathbb{E}[R(T)] \geq \Omega(\sqrt{KT})$.*

This lower bound is "worst-case" (over problem instances). In particular, it leaves open the possibility that an algorithm has low regret for most problem instances. There are two standard ways of proving such lower bounds:

(i) define a family $\mathcal{F}$ of problem instances and prove that any algorithm has high regret on some instance in $\mathcal{F}$.

(ii) define a distribution over $\mathcal{F}$ and prove that any algorithm has high regret in expectation over this distribution.

Remark 3.2. Note that (ii) implies (i), is because if regret is high in expectation over problem instances, then there exists at least one problem instance with high regret. Also, (i) implies (ii) if $|\mathcal{F}|$ is a constant: indeed, if we have high regret H for some problem instance in $\mathcal{F}$, then in expectation over a uniform distribution over $\mathcal{F}$ regret is least $H/|\mathcal{F}|$. However, this argument breaks if $|\mathcal{F}|$ is large. Yet, a stronger version of (i) which says that regret is high for a *constant fraction* of the instances in $\mathcal{F}$ implies (ii) (with uniform distribution) regardless of whether $|\mathcal{F}|$ is large.

On a very high level, our proof proceeds as follows. We consider 0-1 rewards and the following family of problem instances, with parameter $\epsilon > 0$ to be adjusted in the analysis:

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

(Recall that K is the number of arms.) Recall from the previous chapter that sampling each arm $\tilde{O}(1/\epsilon^2)$ times suffices for our upper bounds on regret. Now we prove that sampling each arm $\Omega(1/\epsilon^2)$ times is *necessary* to determine whether this arm is good or bad. This results in regret $\Omega(K/\epsilon)$. We complete the proof by plugging in $\epsilon = \Theta(\sqrt{K/T})$.

The technical details are quite subtle. We present them in several relatively gentle steps.

3.1 Background on KL-divergence

Our proof relies on *KL-divergence*, an important tool from Information Theory. This section provides a simplified introduction to KL-divergence, which is sufficient for our purposes.

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

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

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

KL-divergence is a mathematical construct with amazingly useful properties, see Theorem 3.5 below. As far as this chapter is concerned, the precise definition does not matter as long as these properties are satisfied; in other words, any other construct with the same properties would do just as well for us. Yet, there are deep reasons as to why KL-divergence is defined in this specific way, which are are beyond the scope of this book. This material is usually covered in introductory courses on information theory.

Remark 3.3. While we are not going to explain why KL-divergence should be defined this way, let us see some intuition why this definition makes sense. Suppose we have data points $x_1, \ldots, x_n \in \Omega$, drawn independently from some fixed, but unknown distribution p^*. Further, suppose we know that this distribution is either p or q, and we wish to use the data to estimate which one is more likely. One standard way to quantify whether distribution p is more likely than q is the *log-likelihood ratio*,

$$\Lambda_n := \sum_{i=1}^{n} \frac{\log p(x_i)}{\log q(x_i)}.$$

KL-divergence is the expectation of this quantity, provided that the true distribution is p, and also the limit as $n \to \infty$:

$$\lim_{n\to\infty} \Lambda_n = \mathbb{E}[\Lambda_n] = \mathtt{KL}(p,q) \quad \text{if } p^* = p.$$

Remark 3.4. 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.

We present several basic properties of KL-divergence that will be needed for the rest of this chapter.[1] Throughout, let $\mathtt{RC}_\epsilon$, $\epsilon \geq 0$, denote a biased random coin with bias $\frac{\epsilon}{2}$, *i.e.,* a distribution over $\{0,1\}$ with expectation $(1+\epsilon)/2$.

Theorem 3.5. *KL-divergence satisfies the following properties:*

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

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

(c) **Pinsker's inequality***: for any event* $A \subset \Omega$ *we have* $2\left(p(A) - q(A)\right)^2 \leq \mathtt{KL}(p,q)$*.*

(d) **Random coins***:* $\mathtt{KL}(\mathtt{RC}_\epsilon, \mathtt{RC}_0) \leq 2\epsilon^2$*, and* $\mathtt{KL}(\mathtt{RC}_0, \mathtt{RC}_\epsilon) \leq \epsilon^2$ *for all* $\epsilon \in (0, \frac{1}{2})$*.*

A typical usage of these properties is as follows. Consider the setting from part (b), where $p_j = \mathtt{RC}_\epsilon$ is a biased random coin, and $q_j = \mathtt{RC}_0$ is a fair random coin for each j. Suppose we are interested in some event $A \subset \Omega$, and we wish to prove that $p(A)$ is not too far from $q(A)$ when ϵ is small enough. Then:

$$\begin{aligned}
2(p(A) - q(A))^2 &\leq \mathtt{KL}(p,q) && \textit{(by Pinsker's inequality)}\\
&= \sum_{j=1}^{n} \mathtt{KL}(p_j, q_j) && \textit{(by Chain Rule)}\\
&\leq n \cdot \mathtt{KL}(\mathtt{RC}_\epsilon, \mathtt{RC}_0) && \textit{(by definition of } p_j, q_j\textit{)}\\
&\leq 2n\epsilon^2. && \textit{(by part (d))}
\end{aligned}$$

[1] We present weaker versions of Chain Rule and Pinsker's inequality which suffice for our purposes.

It follows that $|p(A) - q(A)| \leq \epsilon\sqrt{n}$. In particular, $|p(A) - q(A)| < \frac{1}{2}$ whenever $\epsilon < 1/\sqrt{n}$.

We have proved the following:

Lemma 3.6. *Consider sample space $\Omega = \{0,1\}^n$ and two distributions on Ω, $p = \texttt{RC}_\epsilon^n$ and $q = \texttt{RC}_0^n$, for some $\epsilon > 0$.*[2] *Then $|p(A) - q(A)| \leq \epsilon\sqrt{n}$ for any event $A \subset \Omega$.*

Remark 3.7. The asymmetry in the definition of KL-divergence does not matter for the argument above, in the sense that we could have written $\texttt{KL}(q,p)$ instead of $\texttt{KL}(p,q)$. Likewise, it does not matter throughout this chapter.

The proofs of the properties in Theorem 3.5 are not essential for understanding the rest of this chapter, and can be skipped. However, they are fairly simple, and we include them below for the sake of completeness.

Proof of Theorem 3.5(a). 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:

$$\begin{aligned}
\texttt{KL}(p,q) &= \sum_{x\in\Omega} q(x)\,\frac{p(x)}{q(x)}\ln\frac{p(x)}{q(x)} = \sum_{x\in\Omega} q(x) f\left(\frac{p(x)}{q(x)}\right) \\
&\geq f\left(\sum_{x\in\Omega} q(x)\frac{p(x)}{q(x)}\right) && \textit{(by Jensen's inequality (Theorem A.1))} \\
&= f\left(\sum_{x\in\Omega} p(x)\right) = f(1) = 0,
\end{aligned}$$

By Jensen's inequality, since f is not a linear function, equality holds (*i.e.,* $\texttt{KL}(p,q) = 0$) if and only if $p = q$. □

Proof of Theorem 3.5(b). Let $x = (x_1, x_2, \ldots, x_n) \in \Omega$ st $x_i \in \Omega_i, \forall i = 1,\ldots,n$. Let $h_i(x_i) = \ln\frac{p_i(x_i)}{q_i(x_i)}$. Then:

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

Proof of Theorem 3.5(c). To prove this property, we first claim the following:

Claim 3.8. *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:

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

[2] In other words, p is n independent tosses of a biased coin $\texttt{RC}_\epsilon$, and q is n independent tosses of a fair coin $\texttt{RC}_0$.

Then the claim can be proved as follows:

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

□

Fix $A \subset \Omega$. Using Claim 3.8 we have the following:

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

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

$$\begin{aligned}\mathtt{KL}(p,q) &= a\ln\frac{a}{b} + (1-a)\ln\frac{1-a}{1-b}\\ &= \int_a^b \left(-\frac{a}{x} + \frac{1-a}{1-x}\right)dx\\ &= \int_a^b \frac{x-a}{x(1-x)}dx\\ &\geq \int_a^b 4(x-a)dx = 2(b-a)^2. \qquad \textit{(since } x(1-x) \leq \tfrac{1}{4}\textit{)}\end{aligned}$$

□

Proof of Theorem 3.5(d).

$$\begin{aligned}\mathtt{KL}(\mathtt{RC}_0, \mathtt{RC}_\epsilon) &= \tfrac{1}{2}\ln(\tfrac{1}{1+\epsilon}) + \tfrac{1}{2}\ln(\tfrac{1}{1-\epsilon})\\ &= -\tfrac{1}{2}\ln(1-\epsilon^2)\\ &\leq -\tfrac{1}{2}(-2\epsilon^2) \qquad \textit{(as } \log(1-\epsilon^2) \geq -2\epsilon^2 \textit{ whenever } \epsilon^2 \leq \tfrac{1}{2}\textit{)}\\ &= \epsilon^2.\\ \mathtt{KL}(\mathtt{RC}_\epsilon, \mathtt{RC}_0) &= \tfrac{1+\epsilon}{2}\ln(1+\epsilon) + \tfrac{1-\epsilon}{2}\ln(1-\epsilon)\\ &= \tfrac{1}{2}(\ln(1+\epsilon) + \ln(1-\epsilon)) + \tfrac{\epsilon}{2}(\ln(1+\epsilon) - \ln(1-\epsilon))\\ &= \tfrac{1}{2}\ln(1-\epsilon^2) + \tfrac{\epsilon}{2}\ln\tfrac{1+\epsilon}{1-\epsilon}.\end{aligned}$$

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:

$$\mathtt{KL}(\mathtt{RC}_\epsilon, \mathtt{RC}_0) < \tfrac{\epsilon}{2}\cdot\tfrac{2\epsilon}{1-\epsilon} = \tfrac{\epsilon^2}{1-\epsilon} \leq 2\epsilon^2.$$

□

3.2 A simple example: flipping one coin

We start with a simple application of the KL-divergence technique, which is also interesting as a standalone result. Consider a biased random coin (*i.e.,* a distribution on $\{0,1\}$) with an unknown mean $\mu \in [0,1]$. Assume that $\mu \in \{\mu_1, \mu_2\}$ for two known values $\mu_1 > \mu_2$. The coin is flipped T times. The goal is to identify if $\mu = \mu_1$ or $\mu = \mu_2$ with low probability of error.

Let us make our goal a little more precise. Define $\Omega := \{0,1\}^T$ to be the sample space for the outcomes of T coin tosses. Let us say that we need a decision rule $\texttt{Rule} : \Omega \to \{\texttt{High}, \texttt{Low}\}$ with the following two properties:

$$\Pr[\texttt{Rule}(observations) = \texttt{High} \mid \mu = \mu_1] \geq 0.99, \tag{3.2}$$

$$\Pr[\texttt{Rule}(observations) = \texttt{Low} \mid \mu = \mu_2] \geq 0.99. \tag{3.3}$$

We ask, how large should T be for for such a decision rule to exist? We know that $T \sim (\mu_1 - \mu_2)^{-2}$ is sufficient. What we prove is that it is also necessary. We will focus on the special case when both μ_1 and μ_2 are close to $\frac{1}{2}$.

Lemma 3.9. *Let $\mu_1 = \frac{1+\epsilon}{2}$ and $\mu_2 = \frac{1}{2}$. For any decision rule to satisfy properties (3.2) and (3.3) we need $T > \frac{1}{4\,\epsilon^2}$.*

Proof. Fix a decision rule which satisfies (3.2) and (3.3), and let $A_0 \subset \Omega$ be the event this rule returns `High`. Then

$$\Pr[A_0 \mid \mu = \mu_1] - \Pr[A_0 \mid \mu = \mu_2] \geq 0.98. \tag{3.4}$$

Let $P_i(A) = \Pr[A \mid \mu = \mu_i]$, for each event $A \subset \Omega$ and each $i \in \{1,2\}$. Then $P_i = P_{i,1} \times \ldots \times P_{i,T}$, where $P_{i,t}$ is the distribution of the t^{th} coin toss if $\mu = \mu_i$. Thus, the basic KL-divergence argument summarized in Lemma 3.6 applies to distributions P_1 and P_2. It follows that $|P_1(A) - P_2(A)| \leq \epsilon\sqrt{T}$. Plugging in $A = A_0$ and $T \leq \frac{1}{4\epsilon^2}$, we obtain $|P_1(A_0) - P_2(A_0)| < \frac{1}{2}$, contradicting (3.4). □

Remarkably, the proof does not really consider what a given decision rule is doing, and applies to all rules at once!

3.3 Flipping several coins: "bandits with prediction"

Let us extend the previous example to flipping multiple coins. We consider a bandit problem with K arms, where each arm corresponds to a biased random coin with unknown mean. More formally, the reward of each arm is drawn independently from a fixed but unknown Bernoulli distribution. After T rounds, the algorithm outputs an arm y_T, which is the algorithm's prediction for which arm is optimal (has the highest mean reward). We call this version "bandits with predictions". We will only be concerned with the quality of prediction, rather than regret.

As a matter of notation, recall that with 0-1 rewards, a problem instance can be specified as a tuple $\mathcal{I} = (\mu(a) : \forall a \in \mathcal{A})$, where $\mu(a)$ is the mean reward of arm a and $\mathcal{A}$ is the set of all arms. We will number arms from 1 to K.

For concreteness, let us say that a good algorithm for "bandits with predictions" should satisfy

$$\Pr[\text{prediction } y_T \text{ is correct } \mid \mathcal{I}] \geq 0.99 \tag{3.5}$$

for each problem instance $\mathcal{I}$. We will use the family (3.1) of problem instances, with parameter $\epsilon > 0$, to argue that one needs $T \geq \Omega\left(\frac{K}{\epsilon^2}\right)$ for any algorithm to "work", *i.e.,* satisfy property (3.5), on all instances in this family. This result is of independent interest, regardless of the regret bound that we've set out to prove.

In fact, we prove a stronger statement which will also be the crux in the proof of the regret bound.

Lemma 3.10. *Consider a "bandits with predictions" problem with $T \leq \frac{cK}{\epsilon^2}$, for a small enough absolute constant $c > 0$. Fix any deterministic algorithm for this problem. Then there exists at least $\lceil K/3 \rceil$ arms a such that*

$$\Pr[y_T = a \mid \mathcal{I}_a] < \tfrac{3}{4}. \tag{3.6}$$

The proof for $K = 2$ arms is particularly simple, so we present it first. The general case is somewhat more subtle. We only present a simplified proof for $K \geq 24$, which is deferred to Section 3.4.

Proof ($K = 2$ arms). Let us set up the sample space which we will use 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 $\mu(a)$.[3] 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

[3] We use standard shorthand $[T] = \{1, 2, \ldots, T\}$.

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 Ω as follows:

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

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

Let $A = \{\omega \subseteq \Omega : y_T = 1\}$ be the event that the algorithm predicts arm 1. For the sake of contradiction, assume that (3.6) fails for both arms. Then $P_1(A) \geq \frac{3}{4}$ and $P_2(A) < \frac{1}{4}$, so their difference is at least $\frac{1}{2}$.

To arrive at a contradiction, we use a similar KL-divergence argument as before:

$$\begin{aligned}
2(P_1(A) - P_2(A))^2 &\leq \mathtt{KL}(P_1, P_2) && \textit{(by Pinsker's inequality)}\\
&= \sum_{a=1}^{K}\sum_{t=1}^{T} \mathtt{KL}(P_1^{a,t}, P_2^{a,t}) && \textit{(by Chain Rule)}\\
&\leq 2T\cdot 2\epsilon^2 && \textit{(by Theorem 3.5(d)).}
\end{aligned} \tag{3.7}$$

The last inequality holds because for each arm a and each round t, one of the distributions $P_1^{a,t}$ and $P_2^{a,t}$ is a fair coin $\mathtt{RC}_0$, and another is a biased coin $\mathtt{RC}_\epsilon$. Therefore,

$$P_1(A) - P_2(A) \leq 2\epsilon\sqrt{T} < \tfrac{1}{2} \quad \text{whenever } T \leq (\tfrac{1}{4\epsilon})^2. \qquad \square$$

Corollary 3.11. *Assume T is as in Lemma 3.10. Fix any algorithm for "bandits with predictions". Choose an arm a uniformly at random, and run the algorithm on instance $\mathcal{I}_a$. Then $\Pr[y_T \neq a] \geq \frac{1}{12}$, where the probability is over the choice of arm a and the randomness in rewards and the algorithm.*

Proof. Lemma 3.10 easily implies this corollary for deterministic algorithms, which in turn implies it for randomized algorithms (because any randomized algorithm can be expressed as a distribution over deterministic algorithms). $\square$

Finally, we use Corollary 3.11 to finish our proof of the $\sqrt{KT}$ lower bound on regret. We prove the following:

Theorem 3.12. *Fix time horizon T and the number of arms K. Fix a bandit algorithm. Choose an arm a uniformly at random, and run the algorithm on problem instance $\mathcal{I}_a$. Then*

$$\mathbb{E}[R(T)] \geq \Omega(\sqrt{KT}), \tag{3.8}$$

where the expectation is over the choice of arm a and the randomness in rewards and the algorithm.

Proof. Fix ϵ in (3.1), to be adjusted later, and assume that $T \leq \frac{cK}{\epsilon^2}$, where c is the constant from Lemma 3.10.

Fix round t. Let us interpret the algorithm as a "bandits with predictions" algorithm, where the prediction is simply a_t, the arm chosen in this round. We can apply Corollary 3.11, treating t as the time horizon, to deduce that $\Pr[a_t \neq a] \geq \frac{1}{12}$. In words, the algorithm chooses a non-optimal arm with probability at least $\frac{1}{12}$. Recall that for each problem instances $\mathcal{I}_a$, the "badness" $\Delta(a_t) := \mu^* - \mu(a_t)$ is $\epsilon/2$ whenever a non-optimal arm is chosen. Therefore,

$$\mathbb{E}[\Delta(a_t)] = \Pr[a_t \neq a]\cdot\tfrac{\epsilon}{2} \geq \epsilon/24.$$

Summing up over all rounds, we obtain $\mathbb{E}[R(T)] = \sum_{t=1}^{T}\mathbb{E}[\Delta(a_t)] \geq \epsilon T/24$. Using $\epsilon = \sqrt{\frac{cK}{T}}$, we obtain (3.8). $\square$

3.4 Proof of Lemma 3.10 for $K \geq 24$ arms

This is the crucial technical argument in the proof of our regret bound. Compared to the case of $K = 2$ arms, we need to handle a time horizon that can be larger by a factor of $O(K)$. The crucial improvement is a more delicate version of the KL-divergence argument, which improves the right-hand side of (3.7) to $O(T\epsilon^2/K)$.

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

$$\mathcal{I}_0 = \{\mu_i = \tfrac{1}{2} \text{ for all arms } i\ \},$$

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 the following:

$$\text{there are at least } \tfrac{2K}{3} \text{ arms } j \text{ such that } \mathbb{E}_0(T_j) \leq \tfrac{3T}{K}, \tag{3.9}$$

$$\text{there are at least } \tfrac{2K}{3} \text{ arms } j \text{ such that } P_0(y_T = j) \leq \tfrac{3}{K}. \tag{3.10}$$

(To prove (3.9), 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. (3.10) is proved similarly.) By Markov inequality,

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

Since the sets of arms in (3.9) and (3.10) must overlap on least $\frac{K}{3}$ arms, we conclude:

$$\text{there are at least } \tfrac{K}{3} \text{ arms } j \text{ such that } \Pr[T_j \leq \tfrac{24T}{K}] \geq \tfrac{7}{8} \text{ and } P_0(y_T = j) \leq \tfrac{3}{K}. \tag{3.11}$$

We will now refine our definition of the sample space. 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 (3.11). We will consider a "reduced" sample space in which arm j is played only $m = \frac{24T}{K}$ times:

$$\Omega^* = \Omega_j^m \times \prod_{\text{arms } a \neq j} \Omega_a^T. \tag{3.12}$$

For each problem instance $\mathcal{I}_\ell$, we define distribution P_ℓ^* on Ω^* as follows:

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

In other words, distribution P_ℓ^* is a restriction of P_ℓ to the reduced sample space Ω^*.

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

$$\begin{aligned}
2(P_0^*(A) - P_j^*(A))^2 &\leq \mathtt{KL}(P_0^*, P_j^*) && \textit{(by Pinsker's inequality)} \\
&= \sum_{\text{arms } a} \sum_{t=1}^{T} \mathtt{KL}(P_0^{a,t}, P_j^{a,t}) && \textit{(by Chain Rule)} \\
&= \sum_{\text{arms } a \neq j} \sum_{t=1}^{T} \mathtt{KL}(P_0^{a,t}, P_j^{a,t}) + \sum_{t=1}^{m} \mathtt{KL}(P_0^{j,t}, P_j^{j,t}) \\
&\leq 0 + m \cdot 2\epsilon^2 && \textit{(by Theorem 3.5(d)).}
\end{aligned}$$

The last inequality holds because each arm $a \neq j$ has identical reward distributions under problem instances $\mathcal{I}_0$ and $\mathcal{I}_j$ (namely the fair coin $\mathtt{RC}_0$), and for arm j we only need to sum up over m samples rather than T.

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

$$|P_0^*(A) - P_j^*(A)| \leq \epsilon\sqrt{m} < \tfrac{1}{8} \quad \text{for all events } A \subset \Omega^*. \tag{3.13}$$

To apply (3.13), we need to make sure that the event A is in fact contained in Ω^*, *i.e.,* whether this event 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, because this event may depend on more than m samples of arm j. Instead, we apply (3.13) twice: to events

$$A = \{y_T = j \text{ and } T_j \leq m\} \text{ and } A' = \{T_j > m\}. \tag{3.14}$$

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:

$$\begin{aligned}
P_j(A) &\leq \tfrac{1}{8} + P_0(A) && \text{(by (3.13))}\\
&\leq \tfrac{1}{8} + P_0(y_T = j)\\
&\leq \tfrac{1}{4} && \text{(by our choice of arm } j\text{)}.\\
P_j(A') &\leq \tfrac{1}{8} + P_0(A') && \text{(by (3.13))}\\
&\leq \tfrac{1}{4} && \text{(by our choice of arm } j\text{)}.\\
P_j(Y_T = j) &\leq P_j^*(Y_T = j \text{ and } T_j \leq m) + P_j^*(T_j > m)\\
&= P_j(A) + P_j(A') \leq \tfrac{1}{4}.
\end{aligned}$$

This holds for any arm j satisfying the properties in (3.11). Since there are at least $K/3$ such arms, the lemma follows.

3.5 Instance-dependent lower bounds (without proofs)

There is another fundamental lower bound on regret, which applies to any given problem instance and asserts $\log(T)$ regret with an instance-dependent constant. This lower bound complements the $\log(T)$ *upper* bound that we proved for algorithms UCB1 and Successive Elimination. We present and discuss this lower bound without a proof.

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 3.13. *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.*

This version guarantees at least one problem instance on which a given algorithm has "high" regret. We would like to have a stronger lower bound which guarantees "high" regret for each problem instance. However, such lower bound is impossible because of a trivial counterexample: an algorithm which always plays arm 1, as dumb as it is, nevertheless has 0 regret on any problem instance for which arm 1 is optimal. Therefore, the desired lower bound needs to assume that the algorithm is at least somewhat good, so as to rule out such counterexamples.

Theorem 3.14. *Fix K, the number of arms. Consider an algorithm such that*

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

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

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

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

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

Remark 3.15. For example, Assumption (3.15) is satisfied for any algorithm with $\mathbb{E}[R(t)] \leq (\log t)^{1000}$.

Let us refine Theorem 3.14 and specify how the instance-dependent constant $C_{\mathcal{I}}$ in (3.16) can be chosen. In what follows, $\Delta(a) = \mu^* - \mu(a)$ be the "badness" of arm a.

Theorem 3.16. *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

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

Remark 3.17. 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 μ^* is bounded away from 0 and 1, *e.g.,* when $\mu^* \in [\frac{1}{4}, \frac{3}{4}]$.

Remark 3.18. 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 (another standard algorithm discussed in Chapter 4) 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.

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 (Theorem 3.5(b) in Section 3.1), which is much easier to state and to apply. The proof of Lemma 3.10 (for general K), which in prior work relies on the general "chain rule", is modified accordingly. In particular, we define the "reduced" sample space Ω^* with only a small number of samples from the "bad" arm j, and apply the KL-divergence argument to carefully defined events in (3.14), rather than a seemingly more natural event $A = \{y_T = j\}$.

The proof of the logarithmic lower bound from Section 3.5 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).

3.7 Exercises

Exercise 3.1 (lower bound for non-adaptive exploration). Consider an algorithm such that:

- in the first N rounds ("exploration phase") the choice of arms does not depend on the observed rewards, for some N that is fixed before the algorithm starts;
- in all remaining rounds, the algorithm only uses rewards observed during the exploration phase.

Focus on the case of two arms, and prove that such algorithm must have regret $\mathbb{E}[R(T)] \geq \Omega(T^{2/3})$ in the worst case.

Hint: Regret is a sum of regret from exploration, and regret from exploitation. For "regret from exploration", we can use two instances: $(\mu_1, \mu_2) = (1, 0)$ and $(\mu_1, \mu_2) = (0, 1)$, *i.e.,* one arm is very good and another arm is very bad. For "regret from exploitation" we can invoke the impossibility result for "bandits with predictions" (Corollary 3.11).

Take-away: Regret bound for Explore-First cannot be substantially improved. Further, allowing Explore-first to pick different arms in exploitation does not help.

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[4] 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.

[4] Recall that the *reward function* μ maps arms to its mean rewards. We can also view μ 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 μ is fully specified by two parameters, θ and ω, and we have a prior on θ but nothing is known about ω. Then the hybrid approach would strive to prove a regret bound of the following form:

For each ω, the regret of this algorithm in expectation over θ 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 θ. Another example: the problem instances $\mathcal{I}_j$ in the lower bound are parameterized by two things: the best arm a^* and the number ϵ; so, *e.g.,* we could have a uniform distribution over the a^*, but no information on the ϵ.

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

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

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 μ, 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 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 $H_t = (h_1, h_2, \ldots, h_t)$ 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 $\{1, 2, \ldots, t\}$.

Sample space. Consider a fixed bandit algorithm. While we defined $\mathbb{P}$ as a distribution over reward functions μ, 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 (μ, H_t). This is because the choice of μ 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 μ, such as the best arm for a given μ. 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

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

Thompson Sampling is a very simple algorithm:

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

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 μ_t from the posterior distribution $\mathbb{P}_t(\mu) = \mathbb{P}(\mu|H_t)$.
2. choose the best arm $\tilde{a}_t$ according to μ_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:

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

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

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

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 μ 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 μ.*

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 $\mathbb{P}^a_t$, the t-round posterior for arm a, is given by a simple formula:*

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

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^a_t. Let A denote the event in (4.4): that the random vector Z^a_t is consistent with the summary S^a_t. Then:

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

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 $\mathbb{P}^a_t$ is a distribution with density*

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

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 $\texttt{Beta}(\alpha+1, n+1)$. This is a well-studied distribution, *e.g.,* see the corresponding Wikipedia page. $\texttt{Beta}(1,1)$ 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 $\texttt{Beta}(\alpha_0, n_0)$. Then the posterior $\mathbb{P}^a_t$ is a Beta distribution $\texttt{Beta}(\alpha+\alpha_0, n+n_0)$, where $\alpha = \alpha_t(a)$ and $n = n_t(a)$.*

Remark 4.6. By Lemma 4.4, starting with $\texttt{Beta}(\alpha_0, n_0)$ 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 μ^a_0 and standard deviation σ^a_0. Then the posterior $\mathbb{P}^a_t$ is a Gaussian whose mean and variance are determined by the known parameters $(\mu^a_0,\ \sigma^a_0,\ \hat{\sigma})$, 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:

$$\mathop{\mathbb{E}}_{\mu \sim prior}[\mathop{\mathbb{E}}_{rewards}[\, R(t) \mid \mu \,]\,]$$

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):

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

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:

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

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

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:*

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

Proof. First note that:

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

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:

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

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:

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

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

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

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

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

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,

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

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. □

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* μ, *but not on the* ϵ, *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*:

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

where L is a constant known to the algorithm.[1]

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*

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

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

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

[1] A function $\mu : X \to \mathbb{R}$ 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

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

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

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

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

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

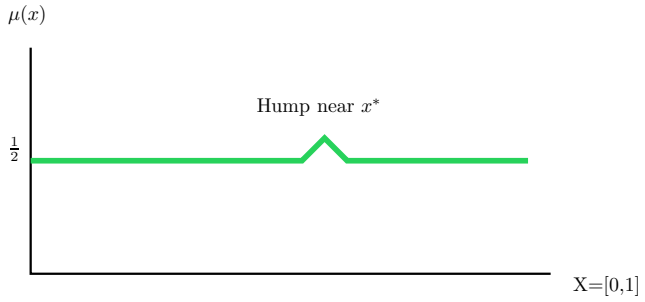
5.1.2 Lower Bound

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

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

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*

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

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 $a^* \in [K] := \{1, \ldots, K\}$. Thus, the set of arms $X = [0,1]$ is partitioned into K disjoint intervals of length 2ϵ, 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 $\mathcal{I}=\mathcal{I}(a^*,\epsilon)$*

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

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 $[K]=\{1,2,\ \ldots\ ,K\}$. 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 $x \in \left[\frac{a}{K}-\epsilon,\ \frac{a}{K}+\epsilon\right)$, $a\in[K]$	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 $\left[\frac{a}{K}-\epsilon,\ \frac{a}{K}+\epsilon\right)$, 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:

$$r_x = \begin{cases} r & \text{with probability } p_x \\ X & \text{otherwise} \end{cases} \qquad \text{where} \quad p_x = 1 - \left|x-\tfrac{a}{K}\right|/\epsilon, \tag{5.6}$$

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

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

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

5.2 Lipschitz MAB

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

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

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 $\mathcal{D} : X \times X \to \mathbb{R}$ 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 ℓ_1-metric (a.k.a. Manhattan metric) and ℓ_2-metric (a.k.a. Euclidean distance).
- X can be any subset of $[0, 1]^d$, finite or infinite, under the same distance function ℓ_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 $\texttt{DE}(S) \leq \epsilon$ and $|S|$ as small as possible.

Example 5.3. Suppose the metric space is $X = [0,1]^d$ under the ℓ_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 $\texttt{DE}(S) \leq c_{p,d} \cdot \epsilon$, 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

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

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 *ϵ-mesh* if $\forall x \in X \ \exists y \in S$ such that $\mathcal{D}(x,y) \leq \epsilon$.

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

Definition 5.5 (ϵ-covering and covering number)**.** A collection of subsets $X_i \subset X$ such that the diameter of each subset is at most ϵ and $X = \bigcup X_i$. The smallest number of subsets in an ϵ-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 ϵ-covering $\{X_1, \ldots, X_N\}$. For each subset X_i, pick an arbitrary representative $x_i \in X_i$. Then $S = \{x_1, \ldots, x_N\}$ is an ϵ-mesh.*

Thus, for each $\epsilon > 0$ there exists an ϵ-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 ϵ simultaneously:

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

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

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, $\texttt{COV}_c(X') \leq \texttt{COV}_c(X)$.

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 $d = \texttt{COV}_c(X)$ 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 $\Omega\left(T^{(d+1)/(d+2)}\right)$. This lower bound can be achieved on $X = [0,1]^d$, with any ℓ_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 "probles": 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^*:

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

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 to 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

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

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 $\texttt{index}_t(x)$ is very similar, and shares the intuition behind UCB1: if $\texttt{index}_t(x)$ 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 $\texttt{index}_t(x)$ 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 $\texttt{index}_t(x)$.

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 $\mathcal{E} = \cap_{x \in X} \mathcal{E}_x$.

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 $[T] := \{1, 2, \ldots, T\}$. 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 $\{y_1, \ldots, y_N\}$ 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:

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

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

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

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

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

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. $\Delta(x) \leq 3\, r_t(x)$ *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

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

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

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

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 $r_t(x) = r_s(x) \geq \Delta(x)/3$. □

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. □

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. □

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 δ per round. Therefore:

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

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 $\delta = (\frac{\log T}{T})^{1/(d+2)}$, 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 chose in the analysis only; the algorithm does not depend on the δ.

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 μ. In general, the zooming dimension is at most as large as the covering dimension,[2] 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 δ, 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., 2011b).

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).

[2]More precisely: if $d = \texttt{COV}_c$, then the zooming dimension with multiplier $3^d \cdot c$ is at most d.

Chapter 6

Full Feedback and Adversarial Costs

[author: chapter revised September 2017, seems pretty polished.]

In this chapter, we shift our focus from bandit feedback to full feedback. Recall that in the full-feedback setting, in the end of each round, we observe the outcome not only for the chosen arm, but for all other arms as well. To be in line with the literature on such problems, we express the outcomes as *costs* rather than *rewards*.

As i.i.d. costs are quite "easy" with full feedback, we consider the other extreme: costs that can arbitrarily change over time. We adopt a rather pessimistic mathematical model, called *adversarial costs*, in which the algorithm is playing a game against an "adversary" who selects costs. More formally, the protocol is as follows:

Bandits with full feedback and adversarial costs

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

1. Adversary chooses costs $c_t(a) \geq 0$ for each arm $a \in [K]$.
2. Algorithm picks arm $a_t \in [K]$.
3. Algorithm incurs cost $c_t(a_t)$ for the chosen arm.
4. The costs of all arms, $c_t(a) : \ a \in [K]$, are revealed.

Remark 6.1. Some results assume bounded costs, *e.g.,* $c_t(a) \in [0, 1]$, but we do not make this assumption by default.

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 feedback, we determine which stock to invest for the next day.

A paradigmatic special case is a prediction problem with experts advice. Suppose we need to predict labels for observations, and we are assisted with a committee of experts. In each round, a new observation arrives, and each expert predicts a correct label for it. We listen to the experts, and pick an answer to respond. We then observe the correct answer and costs/penalties of all other answers. Such a process can be described by the following protocol:

Prediction with expert advice

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

1. Observation x_t arrives.
2. K experts predict labels $z_{1,t}, \ldots, z_{K,t}$.
3. Algorithm picks expert $e \in [K]$.
4. Correct label z_t^* is revealed, along with the costs $c(z_{j,t},\ z_t^*)$, $j \in [K]$ for all submitted predictions.
5. Algorithm incurs cost $c_t = c(z_{e,t}, z_t^*)$.

In the adversarial costs framework, we assume that the (x_t, z_t^*) pair is chosen by an adversary before each round. The penalty function $c(\cdot, \cdot)$ is typically assumed to be fixed and known to the algorithm. The basic case is binary costs: the cost is 0 if the answer is correct, and 1 otherwise. Then the total cost is simply the number of mistakes.

Our goal is to do approximately as well as the best expert. Surprisingly, this can be done without any domain knowledge, as explained in the rest of this chapter.

Remark 6.2. Because of this special case, the general case (bandits with full feedback) is usually called *online learning with experts*, and defined in terms of *costs* (as penalties for incorrect predictions) rather than *rewards*. We will talk about arms, actions and experts interchangeably throughout this chapter.

Remark 6.3 (i.i.d. costs). Consider the special case when the adversary chooses the cost $c_t(a) \in [0,1]$ of each arm a from some fixed distribution $\mathcal{D}_a$, same for all rounds t. With full feedback, this special case is "easy": indeed, there is no need to explore, since costs of all arms are revealed after each round. With a naive strategy such as playing arm with the lowest average cost, one can achieve regret $O\left(\sqrt{T \log(KT)}\right)$. Further, there is a nearly matching lower regret bound $\Omega(\sqrt{T} + \log K)$. The proofs of these results are left as exercise. The upper bound can be proved by a simple application of clean event/confidence radius technique that we've been using since Chapter 2. The $\sqrt{T}$ lower bound follows from the same argument as the bandit lower bound for two arms in Chapter 3, as this argument does not rely on bandit feedback. The $\Omega(\log K)$ lower bound holds for a simple special case, see Theorem 6.5.

6.1 Adversaries and regret

Let us introduce some crucial terminology. An adversary is called *oblivious* if the costs do not depend on the algorithm's choices. In other words, an oblivious adversary chooses all costs before round 1. Otherwise, the adversary is called *adaptive*. Further, an adversary is called *randomized* if in each round t it chooses distribution over the cost vectors $(c_t(a) : a \in [K])$, and then draws the cost vector independently from this distribution. An adversary is called *deterministic* if this distribution picks some cost vector with probability 1.

The total cost of each arm a is defined as $\texttt{cost}(a) = \sum_{t=1}^{T} c_t(a)$. Intuitively, the "best arm" is an arm with a lowest cost. However, defining the "best arm" and "regret" precisely is a little subtle:

Deterministic-oblivious adversary. Then the entire "cost table" $(c_t(a) : a \in [K], t \in [T])$ is chosen before the first round. Then, naturally, the best arm is defined as $a^* \in \operatorname{argmin}_{a \in [K]} \texttt{cost}(a)$, and regret is defined as

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

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

One drawback of considering this adversary is that it does not model i.i.d. costs, simply because there is no randomness in costs, whereas we'd like to think if 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, $\texttt{cost}(a)$, 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":

- $\operatorname{argmin}_a \texttt{cost}(a)$: this is the best arm *in hindsight*, *i.e.,* after all costs have been observed. It is a natural notion if we start from the deterministic-oblivious adversary.
- $\operatorname{argmin}_a \mathbb{E}[\texttt{cost}(a)]$: this is be best arm *in foresight*, *i.e.,* an arm you'd pick 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 if we start from i.i.d. costs.

Accordingly, we distinguish two natural notions of regret: the *hindsight regret*, as in (6.1), and the *foresight regret*[1]

$$R(T) = \texttt{cost}(\texttt{ALG}) - \min_{a \in [K]} \mathbb{E}[\texttt{cost}(a)]. \tag{6.2}$$

[1] 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*.

Recall that we used foresight regret for bandits with i.i.d. rewards/costs. 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 hindsight regret.

Adaptive adversary typcially 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 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.

In game-theoretic applications, adaptive adversary can be used to model a game between an algorithm and a self-interested agent that responds to algorithm's moves and strives to optimize its own utility. In particular, the agent may strive to hurt the algorithm if the game is zero-sum. We will touch upon game-theoretic applications in Chapter 12.

An adaptive adversary is assumed to be randomized by default. In particular, this is 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 again.

Crucially, 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. One can side-step these issues via 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*. Regret guarantees relative to the best observed arm are not always satisfactory, due to many problematic examples such as the one above. However, such guarantees are worth studying for several reasons. First, they *are* meaningful in some scenarios, *e.g.,* when algorithm's actions do not substantially affect the total cost of the best arm. Second, such guarantees may be used as a tool to prove results on oblivious adversaries (*e.g.,* see next chapter). Third, such guarantees often follow from the analysis of oblivious adversary with very little extra work.

Adversary in this chapter. We will consider adaptive adversary, unless specified otherwise. We are interested in "hindsight regret" relative to the best observed arm. For ease of comprehension, one can also interpret the same material as working towards hindsight regret guarantees against a randomized-oblivious adversary.

Notation. Let us recap some notation which we will use throughout this chapter. The total cost of arm a is $\texttt{cost}(a) = \sum_{t=1}^{T} c_t(a)$. The best arm is $a^* \in \operatorname{argmin}_{a\in[K]} \texttt{cost}(a)$, and its cost is $\texttt{cost}^* = \texttt{cost}(a^*)$. Note that a^* and $\texttt{cost}^*$ may depend on randomness in rewards, and (for adaptive adversary) on algorithm's actions. As always, K is the number of actions, and T is the time horizon.

6.2 Initial results: binary prediction with experts advice

We consider *binary prediction with experts advice*, a special case where the expert answers $z_{i,t}$ can only take two possible values. For example: is this image a face or not? Is it going to rain today or not?

Let us assume that there exists a *perfect expert* who never makes a mistake. Consider a simple algorithm that disregards all experts who made a mistake in the past, and follows the majority of the remaining experts:

> In each round t, pick the action chosen by the majority of the experts who did not err in the past.

We call this the *majority vote algorithm*. We obtain a strong guarantee for this algorithm:

Theorem 6.4. *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 round t, and let $W_t = |S_t|$. Note that $W_1 = K$, and $W_t \geq 1$ for all rounds t, because the perfect expert is always in S_t. If the algorithm makes a mistake at round t, then $W_{t+1} \leq W_t/2$ because the majority of experts in S_t is wrong and thus excluded from S_{t+1}. It follows that the algorithm cannot make more than $\log_2 K$ mistakes. □

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 does not increase over time by definition.
- 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".

The guarantee in Theorem 6.4 is in fact optimal (the proof is left as Exercise 6.1):

Theorem 6.5. *Consider binary prediction with experts advice. For any algorithm, any* T *and any* K*, there is a problem instance with a perfect expert such that the algorithm makes at least* $\Omega(\min(T, \log K))$ *mistakes.*

Let us turn to the more realistic case where there is no perfect expert among the committee. The majority vote algorithm breaks as soon as all experts make at least one mistake (which typically happens quite soon).

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).

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

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

Algorithm 6: Weighted Majority Algorithm

To analyze the algorithm, we first introduce some notation. Let $w_t(a)$ be the weight of expert a before round t, and let $W_t = \sum_{a=1}^{K} w_t(a)$ be the total weight before round t. Let S_t be the set of experts that made incorrect prediction at round t. We will use the following fact about logarithms:

Fact 6.6. *For any* $x \in (0, 1)$, $\ln(1 - x) < -x$.

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

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

Since the weights are non-increasing, we must have

$$W_{t+1} \leq W_t \tag{6.4}$$

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:

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

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

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

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

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

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

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

where the last inequality follows from Fact 6.6. Rearranging the terms, we get

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

Where the last step follows from Fact 6.6 with $x = \frac{\epsilon}{1-\epsilon}$. To summarize, we have proved the following theorem.

Theorem 6.7. *The number of mistakes made by WMA with parameter $\epsilon \in (0,1)$ is at most*

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

Remark 6.8. This bound is very meaningful if $\texttt{cost}^*$ is small, but it does not imply sublinear regret guarantees when $\texttt{cost}^* = \Omega(T)$. Interestingly, it recovers the $O(\ln K)$ number of mistakes in the special case with a perfect expert, *i.e.,* when $\texttt{cost}^* = 0$.

6.3 Hedge Algorithm

We extend the results from the previous section in two ways: to the general case, online learning with experts, and to $o(T)$ (and, in fact, optimal) regret bounds. We start with an easy observation that deterministic algorithms are not sufficient for this goal, because they can be easily "fooled" by an oblivious adversary:

Theorem 6.9. *Consider online learning with K experts and 0-1 costs. Any deterministic algorithm has total cost T for some deterministic-oblivious adversary, even if $\texttt{cost}^* \leq T/K$.*

Thus, with a deterministic algorithm, cannot even recover the guarantee from Theorem 6.7 for the general case of online learning with experts, let alone have $o(T)$ regret. The easy proof is left as Exercise 6.3.

We define a randomized algorithm for online learning with experts, called `Hedge`. This algorithm maintains a weight $w_t(a)$ for each arm 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 specification is shown in Algorithm 7:

parameter: $\epsilon \in (0, \frac{1}{2})$

Initialize the weights as $w_1(a) = 1$ for each arm a.
For each round t:
  Let $p_t(a) = \frac{w_t(a)}{\sum_{a'=1}^{K} w_t(a')}$.
  Sample an arm a_t from distribution $p_t(\cdot)$.
  Observe cost $c_t(a)$ for each arm a.
  For each arm a, update its weight $w_{t+1}(a) = w_t(a) \cdot (1-\epsilon)^{c_t(a)}$.

Algorithm 7: `Hedge` algorithm for online learning with experts

Let us analyze regret of `Hedge`. We break the analysis in several steps, for ease of comprehension.

As in the previous section, denote $W_t = \sum_{a=1}^{K} w_t(a)$ for the total weight of all arms at round t. Throughout, $\epsilon \in (0, \frac{1}{2}$ denotes the parameter in the algorithm.

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

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

Hence, the total weight of last round satisfies

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

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 6.10. *There exist $\alpha, \beta \geq 0$, possibly dependent on ϵ, such that*

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

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

In what follows, let us fix some (α, β) as above. Using this fact with $x = c_t(a)$, we obtain the following:

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

where

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

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 $\mathbb{E}[\texttt{cost}(\texttt{ALG})] = \sum_t \mathbb{E}[F_t]$.

A naive attempt. Using the (6.7), we can obtain:

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

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 3: the telescoping product. Taking a logarithm on both sides of (6.7) and using Fact (6.6), we get

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

Inversing 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:

$$\alpha\, \mathbb{E}[\texttt{cost}(\texttt{ALG})] < \beta \sum_{t=1}^{T} \mathbb{E}[G_t] + \ln K - \ln(1-\epsilon)\, \mathbb{E}[\texttt{cost}^*]. \tag{6.8}$$

In what follows, we use this equation in two different ways. Using it with $\alpha = \epsilon$ and $\beta = 0$, we obtain:

$$\mathbb{E}[\texttt{cost}(\texttt{ALG})] < \tfrac{\ln K}{\epsilon} + \underbrace{\tfrac{1}{\epsilon} \ln(\tfrac{1}{1-\epsilon})}_{\le\, 1 + 2\epsilon \text{ if } \epsilon \in (0,\, \frac{1}{2})} \mathbb{E}[\texttt{cost}^*].$$

$$\mathbb{E}[\texttt{cost}(\texttt{ALG}) - \texttt{cost}^*] < \frac{\ln K}{\epsilon} + 2\epsilon\ \mathbb{E}[\texttt{cost}^*].$$

This yields the main regret bound for Hedge:

Theorem 6.11. *Consider an adaptive adversary such that* $\texttt{cost}^* \le U$ *for some number* U *known to the algorithm. Then* Hedge *with parameter* $\epsilon = \sqrt{\ln K/(2U)}$ *satisfies*

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

If all per-round costs are in $[0,1]$ *interval, then one can take* $U = T$.

Step 4: 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 (6.8) with $\alpha = \ln(\frac{1}{1-\epsilon})$ and $\beta = \alpha^2$ to obtain:

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

Dividing both sides by α and moving terms around, we get

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

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

Lemma 6.12. *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*

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

We use this lemma to derive a regret bound for unbounded costs with small expectation and variance. Further, in the next chapter we use this lemma to analyze a bandit algorithm.

Step 5: unbounded costs with small expectation and variance. Consider an *randomized-oblivious* adversary such that the costs are independent across rounds. Instead of bounding the actual costs $c_t(a)$, let us instead bound their expectation and variance:

$$\mathbb{E}[c_t(a)] \leq \mu \text{ and } \text{Var}(c_t(a)) \leq \sigma^2 \text{ for all rounds } t \text{ and all arms } a. \tag{6.9}$$

Then for each round t we have:

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

Thus, we can use Lemma 6.12 with $U = T(\mu^2 + \sigma^2)$. We obtain:

Theorem 6.13. *Consider online learning with experts, with a randomized-oblivious adversary. Assume the costs are independent across rounds. Assume upper bound (6.9) for some μ and σ known to the algorithm. Then* Hedge *with parameter $\epsilon = \sqrt{\ln K/(3T(\mu^2 + \sigma^2))}$ has regret*

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

6.4 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 chapter mostly 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 chapter.

6.5 Exercises

Exercise 6.1. Consider binary prediction with expert advice, with a perfect expert. Prove Theorem 6.5: prove that any algorithm makes at least $\Omega(\min(T, \log K))$ mistakes in the worst case.

Take-away: The majority vote algorithm is worst-case-optimal for instances with a perfect expert.

Hint: For simplicity, let $K = 2^d$ and $T \geq d$, for some integer d. Construct a distribution over problem instances such that each algorithm makes $\Omega(d)$ mistakes in expectation. Recall that each expert x corresponds to a binary sequence $x \in \{0,1\}^T$, where x_t is the prediction for round t. Put experts in 1-1 correspondence with all possible binary sequences for the first d rounds. Pick the "perfect expert" u.a.r. among the experts.

Exercise 6.2. Assume i.i.d. costs, as in Remark 6.3, and let us focus on "hindsight regret".

(a) Prove that $\min_a \mathbb{E}[\texttt{cost}(a)] \leq \mathbb{E}[\min_a \texttt{cost}(a)] + O(\sqrt{T \log(KT)})$.

Take-away: All $\sqrt{T}$-regret bounds from Chapter 2 carry over to "hindsight regret".

Hint: Consider the "clean event": the event in the Hoeffding inequality holds for the cost sequence of each arm.

(b) Prove that there is a problem instance for which any algorithm suffers regret

$$\mathbb{E}[\texttt{cost}(\texttt{ALG}) - \min_{a\in[K]} \texttt{cost}(a)] \geq \Omega(\sqrt{T \log K}).$$

Hint: Assume all arms have 0-1 costs with mean $\frac{1}{2}$. Use the following fact about random walks:

$$\mathbb{E}[\min_{a} \texttt{cost}(a)] \leq \tfrac{T}{2} - \Omega(\sqrt{T \log K}). \tag{6.10}$$

Note: This example does not carry over to "foresight regret". Since each arm has expected reward of $\frac{1}{2}$ in each round, any algorithm trivially achieves 0 "foresight regret" for this problem instance.

Exercise 6.3. Prove Theorem 6.9: prove that any deterministic algorithm for the online learning problem with K experts and 0-1 costs can suffer total cost T for some deterministic-oblivious adversary, even if $\texttt{cost}^* \leq T/K$.

Take-away: With a deterministic algorithm, cannot even recover the guarantee from Theorem 6.7 for the general case of online learning with experts, let alone have $o(T)$ regret.

Hint: Fix the algorithm. Construct the problem instance by induction on round t, so that the chosen arm has cost 1 and all other arms have cost 0.

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: $\texttt{cost}(a) = \Sigma_{t=1}^{T} c_t(x)$. Regret is defined as the difference between the total cost of the algorithm and the minimum total cost of an action:

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

(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*

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

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 $\min_x \texttt{cost}(x)$ rather than the "foresight" benchmark $\min_x \mathbb{E}[\texttt{cost}(x)]$. (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.

Given: set X of experts.
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 8: 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 $\texttt{cost}(x) = \sum_{t\in[T]} c_t(a_{t,x})$. The goal is to minimize regret w.r.t. best expert (rather than w.r.t. best action):

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

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 8 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:

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

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" $\widehat{R}_{\texttt{Hedge}}(T)$ into a statement about the true costs accumulated by Hedge. Formally, we will ensure that

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

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

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

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

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

Indeed,

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

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:

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

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. □

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 8. 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:

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

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 $\mathbb{E}[R_{\texttt{Hedge}}(T)] \leq \mathbb{E}[\widehat{R}_{\texttt{Hedge}}(T)]$.

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 $\texttt{cost}(\texttt{Exp4}) \leq \texttt{cost}(\texttt{Hedge}) + \gamma T$.

To summarize, we have made sure that:

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

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

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

To minimize regret, chose γ 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

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

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:

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

Each realization of G_t satisfies:

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

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

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:

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

(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 γ to be an arbitrary strictly positive number. So if we make γ 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 $\texttt{Exp4}(\gamma)$ denote the algorithm with parameter γ. Then:

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

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

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

Theorem 7.10. *Algorithm* 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:

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

  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

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

- *Slowly changing costs:* Consider a randomized oblivious adversary such that the expected cost of each arm can change by at most ϵ 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

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

  (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 $\texttt{cost}(a)/T$: namely, the regret bound is of the form $\tilde{O}(\sqrt{V})$, where $V = \sum_{t,a} \left(c_t(a) - \frac{\texttt{cost}(a)}{T} \right)^2$ 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 stating 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:

$$\texttt{cost}(a) = \Sigma_{t=1}^{T} c_t(a).$$

Regret is defined as the difference between the total cost of the algorithm and the minimum total cost of an action:

$$R(T) = \texttt{cost}(\texttt{ALG}) - \min_a \texttt{cost}(a).$$

(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 9.

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.*

Given: best-experts algorithm ALG and parameter $\gamma \in (0, \frac{1}{2})$.
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 9: 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 9) with best-experts algorithm* Hedge *and an appropriate choice of parameter* γ *(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* γT*.*

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 $a_t \subset \texttt{Edges}(G)$.
3. Incurs cost $c_t(a_t) = \Sigma_{e \in a_t} a_e \cdot c_t(e)$ 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:

$$\mathbb{E}[R(T)] \leq \texttt{poly}(d) \cdot T^{\beta}, \quad \text{where } \beta < 1.$$

To this end, we will use the reduction (Algorithm 9) 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:

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{\gamma/d} & \text{if event } \Lambda_{t,e} \text{ happens} \\ 0 & \text{otherwise} \end{cases} \tag{8.2}$$

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/γ, 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.[1]

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:

[1] 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 9 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 boun. 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 γ, we immediately obtain the following theorem:

Theorem 8.6. *Consider combinatorial semi-bandits with deterministic oblivious adversary. Then algorithm* `AlgSBwithFPL` *with appropriately chosen parameter γ 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

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{q_t(e|p_t)} & \text{if } e \in a_t \\ 0 & \text{otherwise,} \end{cases} \tag{8.3}$$

where $q_t(e \mid p_t) = \Pr[e \in a_t \mid p_t]$. This definition allows for the same regret bound (and, in fact, is somewhat better in terms of regret), but requires computing $q_t(e \mid p_t)$. (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:

$$\begin{aligned}
&\mathcal{A} = \{(1,0),\ (0,1)\}\\
&v_1 = (\tfrac{1}{3}, \tfrac{2}{3})\\
&v_t = \begin{cases} (1,0) & \text{if } t \text{ is even,}\\ (0,1) & \text{if } t \text{ is odd.}\end{cases}
\end{aligned}$$

Then the total cost vector is

$$v_{1:t} = \begin{cases} (i+\frac{1}{3}, i-\frac{1}{3}) & \text{if } t = 2i,\\ (i+\frac{1}{3}, i+\frac{2}{3}) & \text{if } t = 2i+1.\end{cases}$$

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 each coordinate of v_0 is sampled independently and uniformly from the interval $[-\frac{1}{\epsilon}, \frac{1}{\epsilon}]$, where ϵ 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

$$\texttt{OPT} = \min_{a \in \mathcal{A}} \texttt{cost}(a) = v_{1:t} \cdot M(v_{1:t}),$$

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) $\texttt{cost}(\texttt{BPL}) \leq \texttt{OPT} + \frac{d}{\epsilon}$
(ii) $\mathbb{E}[\texttt{cost}(\texttt{FPL})] \leq \mathbb{E}[\texttt{cost}(\texttt{BPL})] + \epsilon \cdot U^2 \cdot T$

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:

$$\begin{aligned}\texttt{cost}(\texttt{BPL}) + v_0 \cdot M(v_0) &= \sum_{t=0}^{T} v_t \cdot M(v_{0:T}) \textit{(by definition of BPL)} \\ &\le v_{0:T} \cdot M(v_{0:T}) \textit{(see Claim 8.11 below)} \\ &\le v_{0:T} \cdot M(v_{1:T}) \textit{(by (8.4) with } a = M(v_{1:T})\textit{)} \\ &= v_0 \cdot M(v_{1:T}) + \underbrace{v_{1:T} \cdot M(v_{1:T})}_{\texttt{OPT}}.\end{aligned} \tag{8.5}$$

Subtracting $v_0 \cdot M(v_0)$ from both sides, we obtain Lemma 8.10(i):

$$\texttt{cost}(\texttt{BPL}) - \texttt{OPT} \le \underbrace{v_0}_{\in[-\frac{1}{\epsilon},-\frac{1}{\epsilon}]^d} \cdot \underbrace{[M(v_{1:T}) - M(v_0)]}_{\in[-1,1]^d} \le \frac{d}{\epsilon}.$$

The missing step (8.5) follows from the following claim, with $i = 0$ and $j = T$.

Claim 8.11. *For all rounds $i < j$, $\sum_{t=1}^{j} v_t \cdot M(v_{i:t}) \le v_{i:j} \cdot M(v_{i:j})$.*

Proof. The proof is by induction on $j - i$. The claim is trivially satisfied for the base case $i = j$. For the inductive step:

$$\begin{aligned}\sum_{t=i}^{j-1} v_t \cdot M(v_{i:t}) &\le v_{i:j-1} \cdot M(v_{i:j-1}) \textit{(by inductive hypothesis)} \\ &\le v_{i:j-1} \cdot M(v_{i:j}) \textit{(by (8.4) with } a = M(v_{i:j})\textit{)}.\end{aligned}$$

Add $v_j \cdot M(v_{i:j})$ to both sides to complete the proof. □

Step II: FPL comes close to BPL. We compare the expected costs of FPL and BPL round per round. Specifically, we prove that

$$\mathbb{E}[\underbrace{v_t \cdot M(v_{0:t-1})}_{c_t(a_t) \text{ for FPL}}] \le \mathbb{E}[\underbrace{v_t \cdot M(v_{0:t})}_{c_t(a_t) \text{ for BPL}}] + \epsilon U^2. \tag{8.6}$$

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 $f()$ and v.

Claim 8.12. *For any vectors $v \in R^d$ and $v_t \in [0, U/d]^d$, and any function $f : \mathbb{R}^d \to [0, R]$,*

$$\left| \mathop{\mathbb{E}}_{v_0 \sim \mathcal{D}} [f(v_0 + v) - f(v_0 + v + v_t)] \right| \le \epsilon U R.$$

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 coinside with X and Y, and moreover it holds that $\Pr[X' = Y'] \ge 1 - \epsilon U$. 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 $x_1, x_2, \ldots$ 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 $\texttt{REW}(\texttt{ALG}) = \sum_{t=1}^{T} r_t$, so that the expected reward is

$$\mathbb{E}[\texttt{REW}(\texttt{ALG})] = \sum_{t=1}^{T} \mu(a_t|x_t).$$

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

$$R(T) = \texttt{REW}(\pi^*) - \texttt{REW}(\texttt{ALG}). \tag{9.2}$$

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 $\texttt{ALG}_x$ of algorithm ALG
for *each round* t **do**
  invoke algorithm $\texttt{ALG}_x$ with $x = x_t$
  "play" action a_t chosen by $\texttt{ALG}_x$, return reward r_t to $\texttt{ALG}_x$.
end

Algorithm 10: 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 10 has regret* $\mathbb{E}[R(T)] = O(\sqrt{KT\,|\mathcal{X}|\,\ln T})$*, provided that the bandit algorithm* ALG *has regret* $\mathbb{E}[R_{\texttt{ALG}}(T)] = O(\sqrt{KT \log T})$.

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 Lipshitz 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 Lipschtzness. Namely, we assume that contexts map into the $[0, 1]$ interval (*i.e.,* $\mathcal{X} \subset [0, 1]$) so that the expected rewards are lipshitz w.r.t. the contexts:

$$|\mu(a|x) - \mu(a|x')| \leq L|x - x'| \quad \text{for any arms } a, a' \text{ and contexts } x, x', \tag{9.3}$$

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 ϵ-*uniform mesh* on $[0, 1]$, *i.e.,* the set of all points in $[0, 1]$ that are integer multiples of ϵ. We take

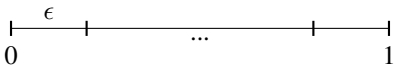


Figure 9.1: Discretization of Context Space

$\epsilon = 1/(d-1)$, 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 $\texttt{ALG}_S$. Let $f_S(x)$ be a mapping from context x to the closest point in S:

$$f_S(x) = \min(\operatorname*{argmin}_{x' \in S} |x - x'|)$$

(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 $\texttt{ALG}_S$.

The regret bound will have two summands: regret bound for $\texttt{ALG}_S$ and (a suitable notion of) discretization error. Formally, let us define the "discretized best response" $\pi^*_S : \mathcal{X} \to \mathcal{A}$:

$$\pi^*_S(x) = \pi^*(f_S(x)) \quad \text{for each context } x \in \mathcal{X}.$$

Then regret of $\texttt{ALG}_S$ and discretization error are defined as, resp.,

$$R_S(T) = \texttt{REW}(\pi^*_S) - \texttt{REW}(\texttt{ALG}_S)$$
$$\texttt{DE}(S) = \texttt{REW}(\pi^*) - \texttt{REW}(\pi^*_S).$$

It follows that the "overall" regret is the sum $R(T) = R_S(T) + \texttt{DE}(S)$, 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 ϵ.

Claim 9.3. $\mathbb{E}[\texttt{DE}(S)] \leq \epsilon LT$.

Proof. For each round t and the respective context $x = x_t$,

$$\begin{aligned}\mu(\pi^*_S(x) \mid f_S(x)) &\geq \mu(\pi^*(x) \mid f_S(x)) \text{(by optimality of } \pi^*_S\text{)}\\ &\geq \mu(\pi^*(x) \mid x) - \epsilon L \text{(by Lipschitzness)}.\end{aligned}$$

Summing this up over all rounds t, we obtain

$$\mathbb{E}[\texttt{REW}(\pi^*_S)] \geq \texttt{REW}[\pi^*] - \epsilon LT. \qquad \square$$

Thus, regret is

$$\mathbb{E}[R(T)] \leq \epsilon LT + O(\sqrt{\tfrac{1}{\epsilon} KT \ln T}) = O(T^{2/3}(LK \ln T)^{1/3}).$$

where for the last inequality we optimized the choice of ϵ.

Theorem 9.4. *Consider the Lispchitz* 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 Lipshitiz condition on both contexts and arms:

$$|\mu(a|x) - \mu(a'|x')| \leq D_\mathcal{X}(x, x') + D_\mathcal{A}(a, a') \quad \text{for any arms } a, a' \text{ and contexts } x, x', \tag{9.4}$$

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 ϵ-mesh $S_\mathcal{X}$ for $(\mathcal{X}, \mathcal{D}_\mathcal{X})$ and an ϵ-mesh $S_\mathcal{A}$ for $(\mathcal{A}, \mathcal{D}_\mathcal{A})$,[1] Fix a bandit algorithm ALG, and run a separate copy $\texttt{ALG}_x$ 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 $\texttt{ALG}_x$.

Further, adaptive discretization approach can improves 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

$$|\mu(a|t) - \mu(a|t')| \leq \mathcal{D}_a(t, t'),$$

[1] As in Lecture 6, an ϵ-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 $\mathcal{D}(y, y') \leq y$.

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 σ_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 $\theta = (\theta_a \in \mathcal{R}^d : a \in \mathcal{A})$.[2]

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 θ 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 Θ be the set of all possible θ 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 11).

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 $\texttt{UCB}_t(a|x_t) = \sup_{\theta \in C_t} x_a \cdot \theta_a$
  pick arm a which maximizes $\texttt{UCB}_t(a|x_t)$.
end

Algorithm 11: 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 Π. Informally, algorithms only need to compete with the best policy in π. 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.

[2] We allow the θ_a in (9.6) to depend on the arm a. The special case when all θ_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 π, the expected reward is defined as

$$\mu(\pi) = \mathop{\mathbb{E}}_{x\in\mathcal{D}}[\mu(\pi(x))|x] \tag{9.7}$$

The regret of an algorithm ALG w.r.t. policy class Π is defined as

$$R_\Pi(T) = T \max_{\pi\in\Pi} \mu(\pi) - \texttt{REW}(\texttt{ALG}). \tag{9.8}$$

Note that the definition (9.2) can be seen a special case when Π 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* Π *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 Π, 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 Π.

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 Π: 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 μ are essentially known. More formally:

$$\text{Given } \mu(a|x) \text{ for all arms } a \text{ and contexts } x \in \mathcal{X}, \text{ find policy } \pi \in \Pi \text{ so as to maximize } \mu(\pi). \tag{9.9}$$

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 $(x_i, c_i(a) \in \mathcal{A})$, $i = 1, \ldots, N$, 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

$$c(\pi) = \frac{1}{N}\sum_{i=1}^{N} c_i(\pi(x_i)). \tag{9.10}$$

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 Π. An algorithm for this problem will henceforth be called a *classification oracle* for Π. 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 $x'_1, \ldots, x'_N \in \mathcal{X}$. Then we can define distinct contexts $x'_i = (x_i, i)$, $i \in [N]$. Each policy π 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 Π 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 $t = 1, \ldots, N$ 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:

$$c_t(a) = \begin{cases} -r_t K & \text{if } a = a_t \\ 0, & \text{otherwise.} \end{cases} \tag{9.11}$$

Finally, we call the oracle and use the policy returned by the oracle in the remaining rounds; see Algorithm 12.

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 π_0 returned by the oracle.

Algorithm 12: Explore-then-exploit with a classification oracle

Remark 9.8. Algorithm 12 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* Π. *Algorithm 12 parameterized with oracle* $\mathcal{O}$ *and* $N = T^{2/3}(K \log(|\Pi|T))^{\frac{1}{3}}$ *has regret*

$$\mathbb{E}[R_\Pi(T)] = O(T^{2/3})(K \log(|\Pi|T))^{\frac{1}{3}}.$$

Proof. Let us consider an arbitrary N for now. For a given policy π, 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 $\mu(\pi)$:

$$\begin{aligned} \mathbb{E}[c_t(a)|x_t] &= -\mu(a|x_t) \text{ (for each action } a \in \mathcal{A}) \\ \mathbb{E}[c_t(\pi(x_t))|x_t] &= -\mu(\pi(x)|x_t) \text{ (plug in } a = \pi(x_t)) \\ \mathop{\mathbb{E}}_{x_t \sim D}[c_t(\pi(x_t))] &= \mathop{\mathbb{E}}_{x_t \sim D}[\mu(\pi(x_t))|x_t] \text{ (take expectation over both } c_t \text{ and } x_t) \\ &= -\mu(\pi), \end{aligned}$$

which implies $\mathbb{E}[c(\pi)] = -\mu(\pi)$, as claimed.

Now, let us use this estimate to set up a "clean event":

$$\{|c(\pi) - \mu(\pi)| \leq \texttt{conf} \text{ for all policies } \pi \in \Pi\},$$

where the "confidence radius" is

$$\texttt{conf}(N) = O\left(\sqrt{\frac{K\log(T|\Pi|)}{N}}\right).$$

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

$$\mu(\pi^\star) - \mu(\pi_0) \leq 2\,\texttt{conf}(N).$$

Thus, each round in exploitation contributes at most conf to expected regret. And each round of exploration contributes at most 1. It follows that $\mathbb{E}[R_\Pi(T)] \leq N + 2T\,\texttt{conf}(N)$. Choosing N so that $N = O(T\,\texttt{conf}(N))$, we obtain $N = T^{2/3}(K\log(|\Pi|T))^{\frac{1}{3}}$ and $\mathbb{E}[R_\Pi(T)] = O(N)$. □

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 ϵ. It is easy to see that expected regret increases by an additive factor of ϵT. In practice, there may be a tradeoff between the approximation guarantee ϵ 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 $\mathrm{polylog}(|\Pi|)$ 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 break-through 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 (Syrgkanis et al., 2016a; Rakhlin and Sridharan, 2016; Syrgkanis 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 knapsacks 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 $(d, k; B_1, \ldots, B_d)$ 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 as 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 budgects. 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 s parameterized by some parameter θ: *e.g.,* the second price auction with the reserve price θ. 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 θ. 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 θ 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

$$\texttt{OPT} \triangleq \texttt{OPT}(\mathcal{I}) \triangleq \sup_{\text{algorithms ALG}} \texttt{REW}(\texttt{ALG}|\mathcal{I}),$$

where $\texttt{REW}(\texttt{ALG}|\mathcal{I})$ 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:

$$\texttt{OPT} - \texttt{REW}(\texttt{ALG}) \leq \tilde{O}\left(\sqrt{k \cdot \texttt{OPT}} + \texttt{OPT}\sqrt{\frac{k}{B}}\right), \tag{10.1}$$

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

$$\texttt{OPT} - \texttt{REW}(\texttt{ALG}) \geq \Omega\left(\min\left(\texttt{OPT},\ \sqrt{k \cdot \texttt{OPT}} + \texttt{OPT}\sqrt{\frac{k}{B}}\right)\right). \tag{10.2}$$

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 $\min_i B_i/c_i(D)$; here the minimum is over all resources i. Thus, the expected total reward of $\mathcal{D}$ in the fractional relaxation is

$$\texttt{FR}(\mathcal{D}) = r(\mathcal{D}) \min_{\text{resources } i} \frac{B_i}{c_i(\mathcal{D})}.$$

Note that $\texttt{FR}(\mathcal{D})$ is not known to the algorithm, but can be approximately learned over time.

Further, one can prove that

$$\texttt{OPT} \leq \texttt{OPT}_{\texttt{FR}} \triangleq \sup_{\text{distributions } \mathcal{D} \text{ over arms}} \texttt{FR}(\mathcal{D}).$$

[as: TODO: insert a proof via primal feasibility.]

In fact, the algorithms for BwK compete with the "relaxed benchmark" $\texttt{OPT}_{\texttt{FR}}$ rather than with the original benchmark OPT.

Informally, we are interested in distributions $\mathcal{D}$ such that $\texttt{FR}(\mathcal{D}) = \texttt{OPT}_{\texttt{FR}}(\mathcal{D})$; 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:

$$|\bar{r}_t(a) - r(a)| \leq \texttt{conf}_t(a) \quad \text{for all arms } a \text{ and rounds } t,$$

where $\bar{r}_t(a)$ is the average reward from arm a by round t, and $\texttt{conf}_t(a)$ is the confidence radius for arm a. Likewise, the clean event specifies the high-confidence interval for consumption of each resource i:

$$|\bar{c}_{i,t}(a) - c(a)| \leq \texttt{conf}_t(a) \quad \text{for all arms } a, \text{ rounds } t, \text{ and resources } i,$$

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

$$\mu = (\quad (r(a); c_1(a), \ \ldots \ , c_d(a)) : \quad \text{for all arms } a)$$

such that μ belongs to this confidence region with high probability. Specifically, the confidence region at time t, denoted $\texttt{ConfRegion}_t$, is simply the product of the corresponding confidence intervals for each entry of μ. We call μ the *latent structure* of the problem instance.

Confidence radius. How should we define the confidence radius? The standard definition is

$$\texttt{conf}_t(a) = O\left(\sqrt{\frac{\log T}{n_t(a)}}\right),$$

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:

$$\texttt{conf}_t(a) = O\left(\sqrt{\frac{\nu \log(T)}{n_t(a)}} + \frac{1}{n_t(a)}\right), \tag{10.3}$$

where the parameter ν 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 improves scaling with $n = n_t(a)$: indeed, it is $1/\sqrt{n}$ in the worst case, and essentially $1/n$ when ν 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 $\texttt{FR}(\mathcal{D})$ for some latent structure μ in the current confidence region $\texttt{ConfRegion}_t$. 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 13.

[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 $a_t \sim D$.

Algorithm 13: 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 μ and each distribution $\mathcal{D}$ we have a fractional value $\texttt{FR}(\mathcal{D}|\mu)$ determined by $\mathcal{D}$ and μ. Using confidence region $\texttt{ConfRegion}_t$, we can define the Upper Confidence Bound for $\texttt{FR}(\mathcal{D})$:

$$\texttt{UCB}_t(\mathcal{D}) = \sup_{\mu \in \texttt{ConfRegion}_t} \texttt{FR}(\mathcal{D}|\mu). \tag{10.4}$$

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 ϵ, and redefine $\texttt{FR}(\mathcal{D}|\mu)$ 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 $\texttt{UCB}_t(\mathcal{D})$
- pick arm $a_t \sim D$.

Algorithm 14: 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 $\texttt{FR}(\mathcal{D})$.

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 $\mu \in \texttt{ConfRegion}_t$ which attains the supremum in (10.4). Indeed, re-write the definition of $\texttt{FR}(D)$ more explicitly:

$$\texttt{FR}(\mathcal{D}) = \left(\sum_a \mathcal{D}(a)\, r(a) \right) \left(\min_{\text{resources } i} \frac{B_i}{\sum_a \mathcal{D}(a)\, c_i(a)} \right).$$

Then $\texttt{UCB}_t(\mathcal{D})$ 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 $r^{\texttt{UCB}}(\mathcal{D})$, and the resulting LCB on $c_i(\mathcal{D})$ with $c_i^{\texttt{LCB}}(\mathcal{D})$. Then the linear program is:

$$\begin{array}{ll} \text{maximize} & \tau \times r^{\texttt{UCB}}(\mathcal{D}) \\ \text{subject to} & \tau \times c_i^{\texttt{LCB}}(\mathcal{D}) \leq B_i(1-\epsilon) \\ & \tau \leq T \\ & \sum_a D(a) = 1. \end{array}$$

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 μ (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

$$\Lambda_t^{\texttt{UCB}}(a) = r(a)/\textstyle\sum_i v_{t,i} \; L_{t,i}(a),$$

and choose an arm $a = a_t$ which maximizes $\Lambda_t^{\texttt{UCB}}(a)$. 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)

Appendix A

Tools

We recap several standard tools that we use throughout the book.

Theorem A.1 (Jensen's inequality)**.** *Fix function $f : \mathbb{R} \to \mathbb{R}$, numbers $x_1, \ldots, x_n$, and positive numbers $\alpha_1, \ldots, \alpha_n$ which sum up to 1. Then*

$$f\left(\sum_i \alpha_i x_i\right) \leq \sum_i \alpha_i f(x_i) \quad \text{if } f \text{ is convex,}$$
$$f\left(\sum_i \alpha_i x_i\right) \geq \sum_i \alpha_i f(x_i) \quad \text{if } f \text{ is concave.}$$

In both cases, equality holds if and only if $x_1 = x_2 = \ldots = x_n$ or f is linear.

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.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *23rd Conf. on Learning Theory (COLT)*, pages 41–53, 2010.

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 Nicolo 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, and Gilles Stoltz. Pure Exploration in Multi-Armed Bandit Problems. *Theoretical Computer Science*, 412(19):1832–1852, 2011a.

Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvari. Online Optimization in X-Armed Bandits. *J. of Machine Learning Research (JMLR)*, 12:1587–1627, 2011b.

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.

Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *J. of Machine Learning Research (JMLR)*, 7:1079–1105, 2006.

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.

Shie Mannor and John N. Tsitsiklis. The sample complexity of exploration in the multi-armed bandit problem. *J. of Machine Learning Research (JMLR)*, 5:623–648, 2004.

Alexander Rakhlin and Karthik Sridharan. BISTRO: an efficient relaxation-based method for contextual bandits. In *33nd Intl. Conf. on Machine Learning (ICML)*, 2016.

Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *31th 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 *33nd 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.