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: March 2019

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 the strongest possible results. We emphasize accessibility of the material: while exposure to machine learning and probability/statistics would certainly help, a standard undergraduate course on algorithms, *e.g.,* one based on (Kleinberg and Tardos, 2005), should suffice for background.

With the above principles in mind, the choice specific topics and results is based on the author's subjective understanding of what is important and "teachable" (*i.e.,* presentable in a relatively simple manner). Many important results has been deemed too technical or advanced to be presented in detail.

This book (except Chapter 10) is based on a graduate course at University of Maryland, College Park, taught by the author in Fall 2016. Each book chapter corresponds to a week of the course. The first draft of the book evolved from the course's lecture notes. Five of the book chapters were used in a similar manner in a graduate course at Columbia University, co-taught by the author in Fall 2017.

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) and 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.) The author plans to add more material, in addition to the ten chapters already in the manuscript: an introductory chapter on the scope and motivations, and a chapter on connections to incentives and mechanism design. 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 the online lecture notes from (Kleinberg, 2007). The author is grateful to Alekh Agarwal, Bobby Kleinberg, Yishay Mansour, and Rob Schapire for discussions and advice. Chapters 9 and 10 have benefited tremendously from numerous conversations with Karthik Abinav Sankararaman.

Contents

Chapter 1

Bandits with IID Rewards *(rev. Jul'18)*

[TODO: more citations, probably a few paragraphs on practical aspects.]

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.

1.1 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:

Problem protocol: 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:

$$R(t) = \mu^* \cdot t - \sum_{s=1}^{t} \mu(a_s). \tag{1.1}$$

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

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

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 1.1 (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.

1.2 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 1.1: Explore-First with parameter N.

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:

$$\Pr\left\{|\bar{\mu}(a) - \mu(a)| \leq r(a)\right\} \geq 1 - \tfrac{1}{T^4} \tag{1.2}$$

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 (1.2) holds for both arms simultaneously. We will argue separately the clean event, and the "bad event" – the complement of the clean event.

Remark 1.2. 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 (1.2) 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 $O\left(\sqrt{\tfrac{\log T}{N}}\right)$ 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,

$$\begin{aligned} \mathbb{E}[R(T)] &= \mathbb{E}[R(T)|\text{clean event}] \times \Pr[\text{clean event}] \;+\; \mathbb{E}[R(T)|\text{bad event}] \times \Pr[\text{bad event}] \\ &\leq \mathbb{E}[R(T)|\text{clean event}] + T \times O(T^{-4}) \\ &\leq O(\sqrt{\log T} \times T^{2/3}). \end{aligned} \tag{1.3}$$

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

For $K > 2$ arms, we have to apply the union bound for (1.2) 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 $R(T) \leq NK + O(\sqrt{\tfrac{\log T}{N}} \times T)$. 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 (1.3), we obtain:

Theorem 1.3. *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 $t = 1, 2, \ldots$ do
    Toss a coin with success probability $\epsilon_t$;
    if success then
        explore: choose an arm uniformly at random
    else
        exploit: choose the arm with the highest average reward so far
end
```

Algorithm 1.2: 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 $\epsilon_t \sim t^{-1/3}$ (ignoring the dependence on K and $\log t$ for a moment), so that the expected number of exploration rounds up to round t is on the order of $t^{2/3}$, same as in Explore-first with time horizon $T = t$. We derive the same regret bound as in Theorem 1.3, 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 1.2).

Theorem 1.4. *Epsilon-greedy algorithm with exploration probabilities* $\epsilon_t = t^{-1/3} \cdot (K \log t)^{1/3}$ *achieves regret bound* $\mathbb{E}[R(t)] \leq t^{2/3} \cdot O(K \log t)^{1/3}$ *for each round* t.

1.3 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?

1.3.1 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

$$\Pr\left(|\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)\right) \geq 1 - \tfrac{2}{T^4}, \tag{1.4}$$

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 variables, (1.4) 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.

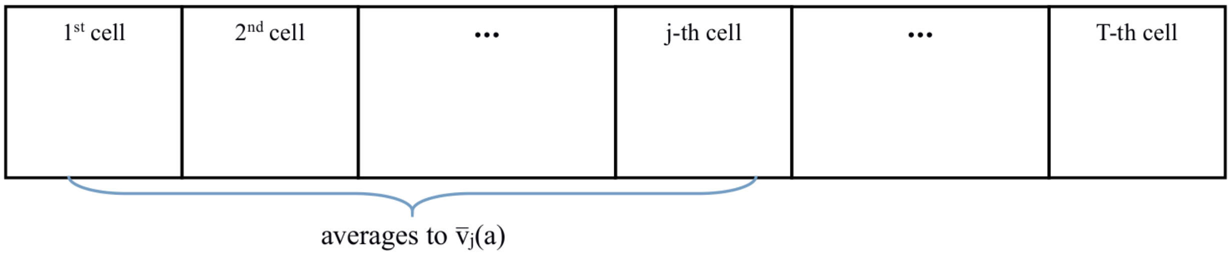


Figure 1.1: the j-th cell contains the reward of the j-th time we pull arm a, *i.e.,* reward of arm a when $n_t(a) = j$

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 1.1. 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 $K = \#\text{arms} \leq T$)

$$\Pr\left(\forall a \forall j \quad |\bar{v}_j(a) - \mu(a)| \leq r_t(a)\right) \geq 1 - \tfrac{2}{T^2}. \tag{1.5}$$

Now, observe that the event in Equation (1.5) implies the event

$$\mathcal{E} := \{\forall a \forall t \quad |\bar{\mu}_t(a) - \mu(a)| \leq r_t(a)\} \tag{1.6}$$

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

Lemma 1.5. $\Pr[\mathcal{E}] \geq 1 - \frac{2}{T^2}$*, where* $\mathcal{E}$ *is given by (1.6).*

The event in (1.6) 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):

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

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

1.3.2 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 1.3: "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?

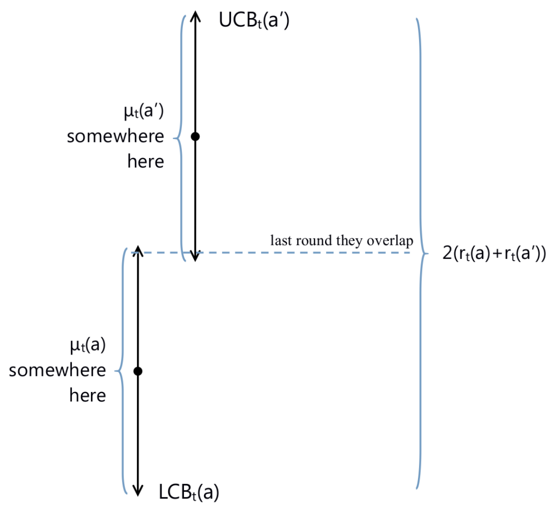


Figure 1.2: t is the last round that the two confidence intervals still overlap

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 1.2). Then

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

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 1.6. *For two arms, Algorithm 1.3 achieves regret* $\mathbb{E}[R(t)] \leq O(\sqrt{t\log T})$ *for each round* $t \leq T$.

Remark 1.7. 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:

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

Algorithm 1.4: Successive Elimination algorithm

To analyze the performance of this algorithm, it suffices to focus on the clean event (1.6); 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:

$$\Delta(a) \leq O(r_T(a)) = O\left(\sqrt{\frac{\log T}{n_T(a)}}\right) \quad \text{for each arm } a \text{ with } \mu(a) < \mu(a^*). \tag{1.7}$$

Informally: an arm played many times cannot be too bad. The rest of the analysis only relies on (1.7). 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 (1.7) 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:

$$R(t) = \sum_{a \in \mathcal{A}^+} R(t; a) = O(\sqrt{\log T}) \sum_{a \in \mathcal{A}^+} \sqrt{n_t(a)} \leq O(\sqrt{\log T}) \sum_{a \in \mathcal{A}} \sqrt{n_t(a)}. \tag{1.8}$$

Since $f(x) = \sqrt{x}$ is a real concave function, and $\sum_{a \in \mathcal{A}} n_t(a) = t$, by Jensen’s Inequality 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 (1.8), we see that $R(t) \leq O(\sqrt{Kt \log T})$. Thus, we have proved:

Theorem 1.8. *Successive Elimination algorithm achieves regret*

$$\mathbb{E}[R(t)] = O(\sqrt{Kt \log T}) \quad \textit{for all rounds } t \leq T. \tag{1.9}$$

We can also use property (1.7) to obtain another regret bound. Rearranging the terms in (1.7), 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:

$$R(T;a) = \Delta(a) \cdot n_T(a) \leq \Delta(a) \cdot O\left(\frac{\log T}{[\Delta(a)]^2}\right) = O\left(\frac{\log T}{\Delta(a)}\right). \tag{1.10}$$

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 1.9. *Successive Elimination algorithm achieves regret*

$$\mathbb{E}[R(T)] \leq O(\log T)\left[\sum_{\text{arms } a \text{ with } \mu(a) < \mu(a^*)} \frac{1}{\mu(a^*) - \mu(a)}\right]. \tag{1.11}$$

Remark 1.10. 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 1.8) 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 1.11. 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 1.12. It is instructive to derive Theorem 1.8 in a different way: starting from the logarithmic regret bound in (1.10). 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 $R(T) \leq O(\sqrt{KT\log T})$.

1.3.3 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 $\underset{a \in \mathcal{A}}{\operatorname{argmax}} \texttt{UCB}_t(a)$, where $\texttt{UCB}_t(a) = \bar{\mu}_t(a) + r_t(a)$;
```

Algorithm 1.5: `UCB1` Algorithm

Remark 1.13. 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 (1.6), 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:

$$\mu(a_t) + 2r_t(a_t) \geq \bar{\mu}_t(a_t) + r_t(a_t) = \texttt{UCB}_t(a_t) \geq \texttt{UCB}_t(a^*) \geq \mu(a^*). \tag{1.12}$$

It follows that

$$\Delta(a_t) := \mu(a^*) - \mu(a_t) \leq 2r_t(a_t) = 2\sqrt{\frac{2\log T}{n_t(a_t)}}. \tag{1.13}$$

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 (1.13) to this round gives us property (1.7). The rest of the analysis follows from that property, as in the analysis of Successive Elimination.

Theorem 1.14. *Algorithm* `UCB1` *satisfies regret bounds in (1.9) and (1.11).*

1.4 Bibliographic remarks and further directions

This chapter introduces several techniques that are broadly useful in multi-armed bandits, beyond the specific setting discussed in this chapter. These are the four algorithmic techniques (Explore-first, Epsilon-greedy, Successive Elimination, and UCB-based arm selection), the 'clean event' technique in the analysis, and the "UCB trick" from (1.12). 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, unclear what are the original references. The original version of `UCB1` has confidence radius

$$r_t(a) = \sqrt{\frac{\alpha \cdot \ln t}{n_t(a)}} \tag{1.14}$$

with $\alpha = 2$; note that $\log T$ is replaced with $\log t$ compared to the exposition in this chapter (see (1.4)). This version allows for the same regret bounds, at the cost of a somewhat more complicated analysis.

Optimality. Regret bounds in (1.9) and (1.11) are near-optimal, according to the lower bounds which we discuss in Chapter 2. The instance-dependent regret bound in (1.9) is optimal up to $O(\log T)$ factors. Audibert and Bubeck (2010) shave off the $\log T$ factor, obtaining an instance dependent regret bound $O(\sqrt{KT})$.

The logarithmic regret bound in (1.11) is optimal up to constant factors. A line of work strived to optimize the multiplicative constant in $O()$. In particular, Auer et al. (2002a); Bubeck (2010); Garivier and Cappé (2011) analyze this constant for `UCB1`, and eventually improve it to $\frac{1}{2\ln 2}$. [2] This factor is the best possible in view of the lower bound in Section 2.5. Further, (Audibert et al., 2009; Honda and Takemura, 2010; Garivier and Cappé, 2011; Maillard et al., 2011) refine the `UCB1` algorithm and obtain improved regret bounds: their regret bounds are at least as good as those for the original algorithm, and get better for some reward distributions.

High-probability regret. In order to upper-bound expected regret $\mathbb{E}[R(T)]$, we actually obtained a high-probability upper bound on $R(T)$. This is common for regret bounds obtained via the "clean event" technique. However, high-probability regret bounds take substantially more work in some of the more advanced bandit scenarios, *e.g.,* for adversarial bandits (see Chapter 6).

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.
- Use `UCB1` with confidence radius $r_t(a) = \sqrt{\frac{2\log t}{n_t(a)}}$, as in (Auer et al., 2002a). This version does not input T, and its regret analysis works for an arbitrary T.
- 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 1.5). 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. See Exercise 1.3.

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 formalize this objective: (i) minimize instantaneous regret $\mu^* - \mu(a_t^*)$, and (ii) maximize the probability of choosing the best arm: $\Pr[a_t^* = a^*]$. The former is often called *pure exploration*, and the latter is called *best-arm identification*. Essentially, good algorithms for cumulative regret, such as Successive Elimination and `UCB1`, are also good for this version (more on this in Exercises 1.3 and 1.4).

[2]More precisely, Garivier and Cappé (2011) derive the constant $\frac{\alpha}{2\ln 2}$, using confidence radius (1.14) with any $\alpha > 1$. The original analysis in Auer et al. (2002a) obtained constant $\frac{8}{\ln 2}$ using $\alpha = 2$.

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). See Exercise 1.4.

1.5 Exercises and Hints

All exercises below are fairly straightforward given the material in this chapter.

Exercise 1.1 (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.

Exercise 1.2 (Epsilon-greedy). Prove Theorem 1.4: derive the $O(t^{2/3}) \cdot (K \log t)^{1/3}$ regret bound for the epsilon-greedy algorithm exploration probabilities $\epsilon_t = t^{-1/3} \cdot (K \log t)^{1/3}$.

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 1.3.1 (treating t as the time horizon), but also include the number of exploration rounds up to time t.

Exercise 1.3 (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

$$\mathbb{E}[\Delta(a_t)] \le \frac{\text{polylog}(T)}{\sqrt{t/K}} \quad \text{for each round } t \in [T]. \tag{1.15}$$

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

Exercise 1.4 (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 $E[R(T)] \le f(T)$, and construct an algorithm for "bandits with predictions" such that $\mathbb{E}[\Delta(y_T)] \le f(T)/T$.

    Note: Surprisingly, taking $y_T = a_t$ does not seem to work in general – definitely not immediately. Taking y_T to be the arm with a maximal empirical 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

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

    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 1.5 (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 2

Lower Bounds *(rev. Jul'18)*

This chapter is about what bandit algorithms *cannot* do. We present two fundamental results which imply that the regret rates in the previous chapter are essentially the best possible.

We are interested in lower bounds on regret which apply to all bandit algorithms, in the sense that no bandit algorithm can achieve better regret. We prove the $\Omega(\sqrt{KT})$ lower bound, which takes most of this chapter. Then we formulate and discuss the instance-dependent $\Omega(\log T)$ lower bound. These lower bounds give us a sense of what are the best possible *upper* bounds that one can hope to achieve.

The $\Omega(\sqrt{KT})$ lower bound is stated as follows:

Theorem 2.1. *Consider multi-armed bandits with IID rewards. 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", leaving open the possibility that it has low regret for many/most other problem instances. To prove such a lower bound, one needs to construct a family $\mathcal{F}$ of problem instances that can "fool" any algorithm. Then there are two standard ways to proceed:

(i) prove that any algorithm has high regret on some instance in $\mathcal{F}$,

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

Remark 2.2. 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. Conversely, (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 over the instances, regardless of how large $|\mathcal{F}|$ is.

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:

$$\mathcal{I}_j = \begin{cases} \mu_i = (1+\epsilon)/2 & \text{for arm } i = j \\ \mu_i = 1/2 & \text{for each arm } i \neq j. \end{cases} \tag{2.1}$$

for each $j = 1, 2, \ldots, K$. (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.[1] We will prove that sampling each arm $\Omega(1/\epsilon^2)$ times is *necessary* to determine whether this arm is good or bad. This leads to regret $\Omega(K/\epsilon)$. We complete the proof by plugging in $\epsilon = \Theta(\sqrt{K/T})$. However, the technical details are quite subtle. We present them in several relatively gentle steps.

2.1 Background on KL-divergence

The proof relies on *KL-divergence*, an important tool from Information Theory. This section provides a brief introduction to KL-divergence that suffices for our purposes. This material is usually covered in introductory courses on information theory.

Throughout, consider a finite sample space Ω, and let p, q be two probability distributions 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, KL-divergence is not symmetric and does not satisfy the triangle inequality.

Remark 2.3. KL-divergence is a mathematical construct with amazingly useful properties (see Theorem 2.5 below). The precise definition does not matter for our purposes, as long as these properties are satisfied; in other words, any other construct with the same properties would do just as well. While there are deep reasons as to why KL-divergence should be defined in this specific way, these reasons are beyond the scope of this book. The definition of KL-divergence and the useful properties thereof extend to infinite sample spaces. However, finite sample spaces suffice for our purposes, and is much easier to work with.

Remark 2.4. 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 $n \to \infty$:

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

We present some of the fundamental properties of KL-divergence that will be needed for the rest of this chapter. 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 2.5. *KL-divergence satisfies the following properties:*

(a) **Gibbs' Inequality**: $\mathtt{KL}(p, q) \geq 0$ *for any two distributions p, q, with equality if and only if $p = q$.*

[1]It immediately follows from Equation (1.7) in Chapter 1.

(b) **Chain rule for product distributions***: Let the sample space be a product $\Omega = \Omega_1 \times \Omega_1 \times \cdots \times \Omega_n$. Let p and q be two distributions on Ω such that $p = p_1 \times p_2 \times \cdots \times p_n$ and $q = q_1 \times q_2 \times \cdots \times q_n$, where p_j, q_j are distributions on Ω_j, for each $j \in [n]$. Then $\texttt{KL}(p,q) = \sum_{j=1}^{n} \texttt{KL}(p_j, q_j)$.*

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

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

A typical usage of these properties is as follows. Consider the setting from part (b) with n samples from two random coins: $p_j = \texttt{RC}_\epsilon$ is a biased random coin, and $q_j = \texttt{RC}_0$ is a fair random coin, for each $j \in [n]$. 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:

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

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

We have proved the following:

Lemma 2.6. *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 $\epsilon > 0$.Then $|p(A) - q(A)| \leq \epsilon\sqrt{n}$ for any event $A \subset \Omega$.*

Remark 2.7. The asymmetry in the definition of KL-divergence does not matter in the argument above: 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 2.5 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 completeness.

Proof of Theorem 2.5(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:

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

In the above application of 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 2.5(b). Let $x = (x_1, x_2, \dots, x_n) \in \Omega$ such that $x_i \in \Omega_i$ for all $i = 1, \dots, n$. 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 2.5(c). To prove this property, we first claim the following:

Claim 2.8. *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:

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

Then the claim can be proved as follows:

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

Fix $A \subset \Omega$. Using Claim 2.8 we have the following:

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

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

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

Proof of Theorem 2.5(d).

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

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:

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

2.2 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: 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

$$\texttt{Rule} : \Omega \to \{\texttt{High}, \texttt{Low}\}$$

which satisfies the following two properties:

$$\Pr[\texttt{Rule}(\text{observations}) = \texttt{High} \mid \mu = \mu_1] \geq 0.99, \tag{2.2}$$

$$\Pr[\texttt{Rule}(\text{observations}) = \texttt{Low} \mid \mu = \mu_2] \geq 0.99. \tag{2.3}$$

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 focus on the special case when both μ_1 and μ_2 are close to $\frac{1}{2}$.

Lemma 2.9. *Let $\mu_1 = \frac{1+\epsilon}{2}$ and $\mu_2 = \frac{1}{2}$. Fix a decision rule which satisfies (2.2) and (2.3). Then $T > \frac{1}{4\,\epsilon^2}$.*

Proof. Let $A_0 \subset \Omega$ be the event this rule returns `High`. Then

$$\Pr[A_0 \mid \mu = \mu_1] - \Pr[A_0 \mid \mu = \mu_2] \geq 0.98. \tag{2.4}$$

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 2.6 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 (2.4). □

Remarkably, the proof applies to all decision rules at once!

2.3 Flipping several coins: "bandits with prediction"

Let us extend the previous example to multiple coins. We consider a bandit problem with K arms, where each arm is 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: a prediction for which arm is optimal (has the highest mean reward). We call this version "bandits with predictions". We are only be concerned with the quality of prediction, rather than regret.

As a matter of notation, the set of arms is $[K]$, $\mu(a)$ is the mean reward of arm a, and a problem instance is specified as a tuple $\mathcal{I} = (\mu(a) : a \in [K])$.

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

$$\Pr[\text{prediction } y_T \text{ is correct } \mid \mathcal{I}] \geq 0.99 \tag{2.5}$$

for each problem instance $\mathcal{I}$. We will use the family (2.1) 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 (2.5), 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 2.10. *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*

$$\Pr[y_T = a \mid \mathcal{I}_a] < \tfrac{3}{4}. \tag{2.6}$$

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 2.4.

Proof ($K = 2$ arms). Let us set up the sample space which we will use in the proof. Let

$$(r_t(a) : \; a \in [K], t \in [T])$$

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

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

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

We need to prove that (2.6) holds for at least one of the arms. For the sake of contradiction, assume it fails for both arms. Let $A = \{\omega \subseteq \Omega : y_T = 1\}$ be the event that the algorithm predicts arm 1. Then $P_1(A) \geq \frac{3}{4}$ and $P_2(A) < \frac{1}{4}$, so their difference is $P_1(A) - P_2(A) \geq \frac{1}{2}$.

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

$$\begin{aligned} 2(P_1(A) - P_2(A))^2 &\leq \mathtt{KL}(P_1, P_2) && \textit{(by Pinsker's inequality)} \\ &= \sum_{a=1}^{K}\sum_{t=1}^{T} \mathtt{KL}(P_1^{a,t}, P_2^{a,t}) && \textit{(by Chain Rule)} \\ &\leq 2T \cdot 2\epsilon^2 && \textit{(by Theorem 2.5(d))}. \end{aligned} \tag{2.7}$$

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 2.11. *Assume T is as in Lemma 2.10. 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 2.10 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 2.11 to finish our proof of the $\sqrt{KT}$ lower bound on regret.

Theorem 2.12. *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*

$$\mathbb{E}[R(T)] \geq \Omega(\sqrt{KT}), \tag{2.8}$$

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

Proof. Fix the parameter $\epsilon > 0$ in (2.1), to be adjusted later, and assume that $T \leq \frac{cK}{\epsilon^2}$, where c is the constant from Lemma 2.10.

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 2.11, 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, $\mathbb{E}[R(T)] = \sum_{t=1}^{T} \mathbb{E}[\Delta(a_t)] \geq \epsilon T/24$. Using $\epsilon = \sqrt{\frac{cK}{T}}$, we obtain (2.8). $\square$

2.4 Proof of Lemma 2.10 for $K \geq 24$ arms

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 (2.7) 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:

$$\text{There are at least } \tfrac{2K}{3} \text{ arms } j \text{ such that } \mathbb{E}_0(T_j) \leq \tfrac{3T}{K}, \tag{2.9}$$

$$\text{There are at least } \tfrac{2K}{3} \text{ arms } j \text{ such that } P_0(y_T = j) \leq \tfrac{3}{K}. \tag{2.10}$$

(To prove (2.9), 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. (2.10) 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 (2.9) and (2.10) must overlap on least $\frac{K}{3}$ arms, we conclude:

$$\text{There are at least } \tfrac{K}{3} \text{ arms } j \text{ such that } \Pr[T_j \leq \tfrac{24T}{K}] \geq \tfrac{7}{8} \text{ and } P_0(y_T = j) \leq \tfrac{3}{K}. \tag{2.11}$$

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 $\Omega = \prod_{a \in [K]} \Omega_a^T$.

Fix an arm j satisfying the two properties in (2.11). We will consider a "reduced" sample space in which arm j is played only $m = \frac{24T}{K}$ times:

$$\Omega^* = \Omega_j^m \times \prod_{\text{arms } a \neq j} \Omega_a^T. \tag{2.12}$$

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 $A \subset \Omega^*$:

$$\begin{aligned}
2(P_0^*(A) - P_j^*(A))^2 &\leq \mathtt{KL}(P_0^*, P_j^*) && \textit{(by Pinsker's inequality)} \\
&= \sum_{\text{arms } a} \sum_{t=1}^{T} \mathtt{KL}(P_0^{a,t}, P_j^{a,t}) && \textit{(by Chain Rule)} \\
&= \sum_{\text{arms } a \neq j} \sum_{t=1}^{T} \mathtt{KL}(P_0^{a,t}, P_j^{a,t}) + \sum_{t=1}^{m} \mathtt{KL}(P_0^{j,t}, P_j^{j,t}) \\
&\leq 0 + m \cdot 2\epsilon^2 && \textit{(by Theorem 2.5(d))}.
\end{aligned}$$

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

$$|P_0^*(A) - P_j^*(A)| \leq \epsilon\sqrt{m} < \tfrac{1}{8} \quad \text{for all events } A \subset \Omega^*. \tag{2.13}$$

To apply (2.13), 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 (2.13) twice: to events

$$A = \{y_T = j \text{ and } T_j \leq m\} \text{ and } A' = \{T_j > m\}. \tag{2.14}$$

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:

$$\begin{aligned}
P_j(A) &\leq \tfrac{1}{8} + P_0(A) && \textit{(by (2.13))}\\
&\leq \tfrac{1}{8} + P_0(y_T = j) &&\\
&\leq \tfrac{1}{4} && \textit{(by our choice of arm j).}\\
P_j(A') &\leq \tfrac{1}{8} + P_0(A') && \textit{(by (2.13))}\\
&\leq \tfrac{1}{4} && \textit{(by our choice of arm j).}\\
P_j(Y_T = j) &\leq P_j^*(Y_T = j \textit{ and } T_j \leq m) + P_j^*(T_j > m) &&\\
&= P_j(A) + P_j(A') \leq \tfrac{1}{4}. &&
\end{aligned}$$

This holds for any arm j satisfying the properties in (2.11). Since there are at least $K/3$ such arms, the lemma follows.

2.5 Instance-dependent lower bounds (without proofs)

There is another fundamental lower bound on regret, which asserts $\log(T)$ regret with an instance-dependent constant and (unlike the $\sqrt{KT}$ lower bound) applies to every problem instance. This lower bound complements the $\log(T)$ *upper* bound that we proved for algorithms UCB1 and Successive Elimination. We formulate and explain this lower bound, without a proof.

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

Theorem 2.13. *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 2.14. *Fix K, the number of arms. Consider an algorithm such that*

$$\mathbb{E}[R(t)] \leq O(C_{\mathcal{I},\alpha}\, t^{\alpha}) \quad \textit{for each problem instance } \mathcal{I} \textit{ and each } \alpha > 0. \tag{2.15}$$

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:

$$\textit{There exists time } t_0 \textit{ such that for any } t \geq t_0 \quad \mathbb{E}[R(t)] \geq C_{\mathcal{I}} \ln(t), \tag{2.16}$$

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

Remark 2.15. For example, Assumption (2.15) is satisfied for any algorithm with $\mathbb{E}[R(t)] \leq (\log t)^{1000}$.

Let us refine Theorem 2.14 and specify how the instance-dependent constant $C_{\mathcal{I}}$ in (2.16) can be chosen. In what follows, $\Delta(a) = \mu^* - \mu(a)$ be the "badness" of arm a.

Theorem 2.16. *For each problem instance $\mathcal{I}$ and any algorithm that satisfies (2.15),*

(a) the bound (2.16) holds with

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

(b) for each $\epsilon > 0$, the bound (2.16) holds with

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

Remark 2.17. 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 2.18. Part (b) is a stronger (*i.e.,* larger) lower bound which implies the more familiar form in (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 3) 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.

2.6 Bibliographic remarks and further directions

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 outline and much of the technical details in the present exposition are based on the lecture notes from Kleinberg (2007). That said, we present a substantially simpler proof, in which we replace the general "chain rule" for KL-divergence with the special case of independent distributions (Theorem 2.5(b) in Section 2.1). This special case is much easier to formulate and apply, especially for those not deeply familiar with information theory. The proof of Lemma 2.10 for general K 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 (2.14), rather than a seemingly more natural event $A = \{y_T = j\}$.

The logarithmic lower bound from Section 2.5 is due to Lai and Robbins (1985). Its proof is also based on a KL-divergence technique. The proof can be found in the original paper (Lai and Robbins, 1985), as well as in the survey (Bubeck and Cesa-Bianchi, 2012). Compared to these sources, our exposition is more explicit on “unwrapping” what this lower bound means.

While these two lower bounds essentially resolve the basic version of multi-armed bandits, they do not suffice for many other versions. This is for several reasons:

- some bandit problems posit auxiliary constraints on the problem instances, such as Lipschitzness or linearity (see Interlude A), and the lower-bounding constructions need to respect these constraints.
- these problems often allow a very large or infinite number of actions, and the lower-bounding constructions need to choose the number of actions in the problem instances so as to obtain the strongest possible lower bound.
- in some bandit problems the constraints are on the algorithm, *e.g.,* a dynamic pricing algorithm needs to stop when and if there are no more items to sell; a bandit algorithm that learns click probabilities for an ad auction needs to take into account the advertisers’ incentives. Then much stronger lower bounds may be possible.

Therefore, a number of problem-specific lower bounds have been proved over the years. A representative, but likely incomplete, list is below:

- for dynamic pricing (Kleinberg and Leighton, 2003; Babaioff et al., 2015). The algorithm is a seller: in each round it offers some good(s) for sale at fixed and non-negotiable price(s). Algorithm’s actions are the chosen price(s), and its reward is the revenue from sales, if any.
- for Lipschitz bandits (Kleinberg, 2004; Slivkins, 2014; Kleinberg et al., 2019). The algorithm is given a “similarity metric” on actions such that similar actions have similar expected rewards.
- for linear bandits (*e.g.,* Dani et al., 2008; Rusmevichientong and Tsitsiklis, 2010; Shamir, 2015). Each action a corresponds to a known feature vector $x_a \in \mathbb{R}^d$, and its expected reward is linear in x_a.
- for pay-per-click ad auctions (Babaioff et al., 2014; Devanur and Kakade, 2009). Ad auctions are parameterized by click probabilities of ads, which are a priori unknown but can be learned over time by a bandit algorithm. The said algorithm is constrained to take into account advertisers’ incentives.
- for bandits with resource constraints (Badanidiyuru et al., 2018). The algorithm is endowed with some resources, *e.g.,* , inventory for sale or budget for purchase. Each action collects a reward *and* spends some amount of each resource. The algorithm is constrained to stop once some resource is depleted.
- for best-arm identification (*e.g.,* Kaufmann et al., 2016; Carpentier and Locatelli, 2016). The setting is the same as in bandits with i.i.d. rewards, but the goal is to identify the best arm with high probability.

Some of these lower bound are derived from first principles, (*e.g.,* Kleinberg and Leighton, 2003; Kleinberg, 2004; Babaioff et al., 2014; Badanidiyuru et al., 2018; Kleinberg et al., 2019), and some are derived by reduction to a pre-existing regret bound (*e.g.,* Babaioff et al., 2015; Kleinberg et al., 2019). Even when the family of problem instances is simple and intuitive, it usually takes a lengthy and intricate KL-divergence argument to derive the actual regret bound. Reduction to a pre-existing regret bound side-steps the KL-divergence argument, and therefore substantially simplifies exposition.

Some of the lower bounds (*e.g.,* Kleinberg and Leighton, 2003; Kleinberg, 2004; Kleinberg et al., 2019) require one randomized problem instance that applies to many (perhaps, infinitely many) time horizons at once. This problem instance consists of multiple "layers", where each layer is responsible for the lower bound at particular time horizon.

2.7 Exercises and Hints

Exercise 2.1 (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 *(rev. Jan'17)*

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? We survey several approaches suggested in the literature.

Constrained reward functions. Here the "initial information" is that the reward function[2] 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

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

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.

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 3

Thompson Sampling *(rev. Jan'17)*

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 is self-contained, introducing Bayesian concepts as needed.

3.1 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.

3.2 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

$$p_t(a) = \mathbb{P}[a = a^\star \mid H_t] \quad \text{for each arm } a. \tag{3.1}$$

Thompson Sampling is a very simple algorithm:

$$\text{In each round } t, \text{ arm } a_t \text{ is drawn independently from distribution } p_t. \tag{3.2}$$

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 3.1. *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:

$$\mathbb{P}_t(\mu) = \prod_{\text{arms a}} \mathbb{P}_t^a(\mu(a)), \quad \text{where} \quad \mathbb{P}_t^a(x) = \mathbb{P}[\mu(a) = x | H_t]. \tag{3.3}$$

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

$$\mathbb{P}_t^a(x) := \mathbb{P}[\mu(a) = x | H_t] = \mathbb{P}^a \left[\mu(a) = x \mid Z_t^a \text{ is consistent with } S_t^a \right]. \tag{3.4}$$

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).

3.3 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 3.2. *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:

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

and only when this arm is chosen in this round.

3.4 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^a_t. 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 3.3. *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 (3.4): that the random vector Z_t^a is consistent with the summary S_t^a. Then:

$$\begin{aligned}
\mathbb{P}_t^a(x) &= \mathbb{P}[\mu(a)=x|H_t] && \text{By definition of the arm-}a\text{ posterior}\\
&= \mathbb{P}^a[\mu(a)=x \mid A] && \text{by (3.4)}\\
&= \frac{\mathbb{P}^a(\mu(a)=x \text{ and } A)}{\mathbb{P}^a(A)}\\
&= \frac{\mathbb{P}^a[\mu(a)=x]\cdot \Pr(A \mid \mu(a)=x)}{\mathbb{P}^a(A)}\\
&= \frac{\frac{1}{N_a}\cdot x^{\alpha}(1-x)^{n-\alpha}}{\mathbb{P}^a(A)}\\
&= Cx^{\alpha}(1-x)^{n-\alpha}
\end{aligned}$$

for some normalization constant C. $\square$

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

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

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 3.4, in which the prior can be an arbitrary Beta-distribution:

Lemma 3.5. *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 $\mathbb{P}_t^a$ is a Beta distribution $\texttt{Beta}(\alpha+\alpha_0,n+n_0)$, where $\alpha=\alpha_t(a)$ and $n=n_t(a)$.*

Remark 3.6. By Lemma 3.4, 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 α_0-1 1's.)

3.5 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 3.7. *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_0^a and standard deviation σ_0^a. Then the posterior $\mathbb{P}_t^a$ is a Gaussian whose mean and variance are determined by the known parameters ($\mu_0^a,\ \sigma_0^a,\ \hat{\sigma}$), as well as the average reward and the number of samples from this arm so far.*

3.6 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):

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

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:

$$\forall a,\ t,\quad \mathbb{E}[\,[U(a,\ H_t) - \mu(a)\,]^-] \le \frac{\gamma}{T \cdot K} \tag{3.5}$$
$$\forall a,\ t,\quad \mathbb{E}[\,[\mu(a) - L(a,\ H_t)]^-] \le \frac{\gamma}{T \cdot K}. \tag{3.6}$$

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

$$BR(T) \leq 2\gamma + 2\sum_{t=1}^{T} \mathbb{E}[r(a_t,\ H_t)] \tag{3.7}$$

Proof. First note that:

$$\mathbb{E}[\, U(a^*,\ H_t) \mid H_t \,] = \mathbb{E}[\, U(a_t,\ H_t) \mid H_t \,] \tag{3.8}$$

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:

$$\begin{aligned}
BR_t(T) &= \mathbb{E}[R(t)] && \text{expectation over everything} \\
&= \mathbb{E}[\mu(a^*) - \mu(a_t)] && \text{instantaneous regret for round t} \\
&= \mathop{\mathbb{E}}_{H_t}[\, \mathbb{E}[\mu(a^*) - \mu(a_t)] \mid H_t] \,] && \text{bring out expectation over history} \\
&= \mathop{\mathbb{E}}_{H_t}[\, \mathbb{E}[U(a_t, H_t) - \mu(a_t) + \mu(a^*) - U(a^*, H_t)] \mid H_t] \,] && \text{by the equality in equation 3.8 above} \\
&= \underbrace{\mathbb{E}[U(a_t, H_t) - \mu(a_t)]}_{\text{Part 1}} + \underbrace{\mathbb{E}[\mu(a^*) - U(a^*, H_t)]}_{\text{Part 2}}.
\end{aligned}$$

We will use properties (3.5) and (3.6) to bound Part 1 and Part 2. Note that we cannot *immediately* use these properties because they assume a fixed arm a, whereas both a_t and a^* are random variables. (The best arm a^* is a random variable because we take an expectation over everything, including the problem instance.)

Let's handle Part 2:

$$\begin{aligned}
\mathbb{E}[\mu(a^*) - U(a^*, H_t)] &\leq \mathbb{E}[(\mu(a^*) - U(a^*, H_t))^+] \\
&\leq \mathbb{E}[\sum_{\text{arms } a} (\mu(a) - U(a, H_t))^+] \\
&= \mathbb{E}[\sum_{\text{arms } a} (\mu(a) - U(a, H_t))^+] \\
&= \sum_{\text{arms } a} \mathbb{E}[(U(a, H_t) - \mu(a))^-] \\
&\leq k * \frac{\gamma}{T * k} && \text{by 3.5 over all arms} \\
&= \frac{\gamma}{T}
\end{aligned}$$

Let's handle Part 1:

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

Then

$$\begin{aligned}
\mathbb{E}[L(a_t, H_t) - \mu(a_t)] &\leq \mathbb{E}[(L(a_t, H_t) - \mu(a_t))^+] \\
&\leq \mathbb{E}[\sum_{\text{arms } a} ((L(a, H_t) - \mu(a))^+] \\
&= \sum_{\text{arms } a} \mathbb{E}[(\mu(a) - L(a, H_t))^-] \\
&\leq k * \frac{\gamma}{T * k} && \text{by 3.6 over all arms} \\
&= \frac{\gamma}{T}
\end{aligned}$$

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

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

Remark 3.9. Lemma 3.8 holds regardless of what the U and L functions are, so long as they satisfy properties (3.5) and (3.6). Furthermore, Thompson Sampling does not need to know what U and L are, either.

Remark 3.10. While Lemma 3.8 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 3.8 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 3.11. *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. □

3.7 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 3.12. *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 3.13. *Consider IID bandits with no priors. For Thompson sampling with approach (i),*

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

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.

3.8 Bibliographic remarks and further directions

The results in Section 3.6 are from Russo and Roy (2014). (The statement of Lemma 3.8 is a corollary of the result proved in Russo and Roy (2014) which makes the technique a little more transparent.) Russo and Roy

(2014) also use this approach to obtain improved upper bounds for some specific classes of priors, including priors over linear reward functions, priors over "generalized linear" reward functions, and priors given by a Gaussian Process.

The prior-independent results in Section 3.7 are from (Agrawal and Goyal, 2012; Kaufmann et al., 2012; Agrawal and Goyal, 2013). Specifically, the first "prior-independent" regret bound for Thompson Sampling — a weaker version of Theorem 3.13 — has appeared in Agrawal and Goyal (2012). Theorem 3.12 is from Agrawal and Goyal (2013), and Theorem 3.13 is from (Kaufmann et al., 2012; Agrawal and Goyal, 2013).[1] For Thompson Sampling with Gaussian priors, Agrawal and Goyal (2013) achieve a slightly stronger version of Theorem 3.12 in which the $\log(T)$ factor is replaced with $\log(K)$, and prove a matching lower bound for Bayesian regret of this algorithm.

[1]Kaufmann et al. (2012) proves a slightly weaker version in which $\ln(T)$ is replaced with $\ln(T) + \ln\ln(T)$.

Chapter 4

Lipschitz Bandits *(rev. Jul'18)*

We consider bandit problems in which an algorithm has information on similarity between arms, summarized via a Lipschitz condition on the expected rewards.

In various bandit problems an algorithm may have information on similarity between arms, so 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 is the dynamic pricing problem, where arms correspond to prices, and similar prices often correspond to similar expected rewards. Abstractly, we assume that arms map to points in a known metric space, and their expected rewards obey a Lipschitz condition relative to this metric space.

Throughout, we 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 would correspond to "points" on a line or in a metric space. As usual, we assume that realized rewards lie in $[0,1]$.

4.1 Continuum-armed bandits

Let us start with a special case called *continuum-armed bandits* (CAB). Here, the set of arms is $X = [0,1]$, and expected rewards $\mu(\cdot)$ satisfy a *Lipschitz Condition*:

$$|\mu(x) - \mu(y)| \leq L \cdot |x - y| \quad \text{for any two arms } x, y \in X, \tag{4.1}$$

where L is a constant known to the algorithm.[1] An instance of CAB is completely specified by the mean rewards $\mu(\cdot)$, the time horizon T, and the Lipschitz constant L. Note that we have infinitely many arms, and in fact, *continuously* many. While bandit problems with a very large, let alone infinite, number of arms are hopeless in general, CAB is tractable due to the Lipschitz condition.

4.1.1 Simple solution: fixed discretization

A simple but powerful technique, called *fixed discretization*, works as follows. We pick a fixed, finite set of arms $S \subset X$, called *discretization* of 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. Adding more points to S makes it a better approximation of X, but also increases regret of `ALG` on S. Thus, S should be chosen so as to optimize this tradeoff.

[1] A function $\mu : X \to \mathbb{R}$ which satisfies (4.1) is called *Lipschitz-continuous* on X, with *Lipschitz constant* L.

The best arm in S is 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*

$$\texttt{DE}(S) = \mu^*(X) - \mu^*(S). \tag{4.2}$$

More formally, we can represent expected regret of the entire algorithm as a sum

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

where $W(\texttt{ALG})$ is the total reward of the algorithm, and $R_S(T)$ is the regret relative to $\mu^*(S)$. If ALG attains optimal regret $O(\sqrt{KT\log T})$ on any problem instance with time horizon T and K arms, then

$$\mathbb{E}[R(T)] \le O(\sqrt{|S|T\log T}) + \texttt{DE}(S)\cdot T. \tag{4.3}$$

This is a concrete expression for the 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 $\epsilon = \frac{1}{k-1}$, so that S consists of all integer multiples of ϵ. It is easy to see that $\texttt{DE}(S) \le L\epsilon$. Indeed, if x^* is a best arm on X, and y is the closest arm to x^* that lies in S, it follows that $|x^* - y| \le \epsilon$, and therefore $\mu(x^*) - \mu(y) \le L\epsilon$. Picking $\epsilon = (TL^2/\log T)^{-1/3}$, we obtain

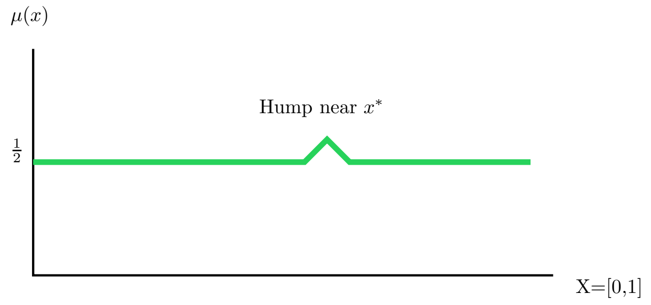
Theorem 4.1. *Consider continuum-armed bandits. Fixed, uniform discretization attains regret*

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

4.1.2 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 from Chapter 2, henceforth called MainLB.

The new lower bound involves problem instances with 0-1 rewards. All arms x have mean reward $\mu(x) = \frac{1}{2}$ except those near the unique best arm x^* with $\mu(x^*) = \frac{1}{2} + \epsilon$. Here $\epsilon > 0$ is a parameter to be adjusted later in the analysis. Due to the Lipschitz condition, we need to ensure a smooth transition between x^* and the arms far away from x^*; hence, we will have a "bump" around x^*.



Formally, we define a problem instance $\mathcal{I}(x^*, \epsilon)$ by

$$\mu(x) = \begin{cases} \frac{1}{2}, & \text{all arms } x \text{ with } |x - x^*| \geq \epsilon/L \\ \frac{1}{2} + \epsilon - L \cdot |x - x^*|, & \text{otherwise.} \end{cases} \tag{4.4}$$

It is easy to see that any such problem instance satisfies the Lipschitz Condition (4.1). We will refer to $\mu(\cdot)$ as the *bump function*. We are ready to state the lower bound:

Theorem 4.2. *Let* ALG *be any algorithm for continuum-armed bandits with time horizon T and Lipschitz constant L. There exists a problem instance $\mathcal{I} = \mathcal{I}(x^*, \epsilon)$, for some $x^* \in [0,1]$ and $\epsilon > 0$, such that*

$$\mathbb{E}\left[R(T) \mid \mathcal{I}\right] \geq \Omega(L^{1/3} \cdot T^{2/3}). \tag{4.5}$$

For simplicity of exposition, assume the Lipschitz constant is $L = 1$; arbitrary L is treated similarly.

Fix $K \in \mathbb{N}$ and partition arms into K disjoint intervals of length $\frac{1}{K}$. Use bump functions with $\epsilon = \frac{1}{2K}$ such that each interval either contains a bump or is completely flat. More formally, we use problem instances $\mathcal{I}(x^*, \epsilon)$ indexed by $a^* \in [K] := \{1, \ldots, K\}$, where the best arm is $x^* = (2\epsilon - 1) \cdot a^* + \epsilon$.

The intuition for the proof is as follows. We have these K intervals defined above. Whenever an algorithm chooses an arm x in one such interval, choosing the *center* of this interval is best: 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 we restrict to arms that are centers of the intervals, we have a family of problem instances of K-armed bandits, where all arms have mean reward $\frac{1}{2}$ except one with mean reward $\frac{1}{2} + \epsilon$. This is precisely the family of instances from MainLB. Therefore, one can apply the lower bound from MainLB, and tune the parameters to obtain (4.5). To turn this intuition into a proof, the main obstacle is to prove that choosing the center of an interval is really the best option. While this is a trivial statement for the immediate round, we need to argue carefullt that choosing an arm elsewhere would not be advantageous later on.

Let us recap MainLB in a way that is convenient for this proof. Recall that MainLB considered problem instances with 0-1 rewards such that all arms a have mean reward $\frac{1}{2}$, except the best arm a^* whose mean reward is $\frac{1}{2} + \epsilon$. Each instance is parameterized by best arm a^* and $\epsilon > 0$, and denoted $\mathcal{J}(a^*, \epsilon)$.

Theorem 4.3 (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 2.10. Then there exists a problem instance $\mathcal{J} = \mathcal{J}(a^*, \epsilon)$, $a^* \in [K]$, such that*

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

To prove Theorem 4.2, 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 (4.5). On a high level, our plan is as follows: (i) take any problem instance $\mathcal{J}$ from MainLB and "embed" it into CAB, and (ii) show that any algorithm for CAB will, in fact, need to solve $\mathcal{J}$, (iii) tune the parameters to derive (4.5).

Proof of Theorem 4.2 ($L = 1$). We use the problem instances $\mathcal{I}(x^*, \epsilon)$ as described above. More precisely, we fix $K \in \mathbb{N}$, to be specified later, and let $\epsilon = \frac{1}{2K}$. We index the instances by $a^* \in [K]$, so that

$$x^* = f(a^*), \text{ where } f(a^*) := (2\epsilon - 1) \cdot a^* + \epsilon.$$

We use problem instances $\mathcal{J}(a^*, \epsilon)$ from MainLB, with K arms and the same time horizon T. The set of arms in these instances is denoted $[K]$. Each instance $\mathcal{J} = \mathcal{J}(a^*, \epsilon)$ corresponds to an instance $\mathcal{I} = \mathcal{I}(x^*, \epsilon)$

of CAB with $x^* = f(a^*)$. In particular, each arm $a \in [K]$ in $\mathcal{J}$ corresponds to an arm $x = f(a)$ in $\mathcal{I}$. We view $\mathcal{J}$ as a discrete version of $\mathcal{I}$. In particular, we have $\mu_{\mathcal{J}}(a) = \mu(f(a))$, where $\mu(\cdot)$ is the reward function for $\mathcal{I}$, and $\mu_{\mathcal{J}}(\cdot)$ is the reward function for $\mathcal{J}$.

Consider an execution of `ALG` on problem instance $\mathcal{I}$ of CAB, and use it to construct an algorithm `ALG`$'$ which solves an instance $\mathcal{J}$ of K-armed bandits. Each round in algorithm `ALG`$'$ proceeds as follows. `ALG` is called and returns some arm $x \in [0,1]$. This arm falls into the interval $[f(a)-\epsilon,\ f(a)+\epsilon)$ for some $a \in [K]$. Then algorithm `ALG`$'$ chooses arm a. When `ALG`$'$ receives reward r, it uses r and x to compute reward $r_x \in \{0,1\}$ such that $\mathbb{E}[r_x \mid x] = \mu(x)$, and feed it back to `ALG`. We summarize this in a table:

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

To complete the specification of `ALG`$'$, it remains to define reward 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 $\mu_{\mathcal{J}}(a)$ or $\mu(x)$, since they are not know to `ALG`$'$. We define r_x as follows:

$$r_x = \begin{cases} r & \text{with probability } p_x \in [0,1] \\ X & \text{otherwise,} \end{cases} \tag{4.6}$$

where X is an independent toss of a Bernoulli random variable with expectation $\frac{1}{2}$, and probability p_x is to be specified later. Then

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

if we set $p_x = 1 - |x - f(a)|/\epsilon$ so as to match the definition of $\mathcal{I}(x^*,\epsilon)$ in Equation (4.4).

For each round t, let x_t and a_t be the arms chosen by `ALG` and `ALG`$'$, resp. Since $\mu_{\mathcal{J}}(a_t) \geq \frac{1}{2}$, we have

$$\mu(x_t) \leq \mu_{\mathcal{J}}(a_t).$$

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

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

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 $\mathcal{J} = \mathcal{J}(a^*,\epsilon)$. Let us apply Theorem 4.3 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 4.3. Then there exists some instance $\mathcal{J} = \mathcal{J}(a^*,\epsilon)$ such that

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

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

4.2 Lipschitz MAB

A useful interpretation of continuum-armed bandits is that arms lie in a particular metric space $(X, \mathcal{D})$, where $X = [0,1]$ is a ground set and $\mathcal{D}(x,y) = L \cdot |x-y|$ is the metric. In this section we extend this problem and the fixed discretization approach to arbitrary metric spaces.

The general problem, called *Lipschitz MAB Problem*, is a multi-armed bandit problem such that the expected rewards $\mu(\cdot)$ satisfy the Lipschitz condition relative to some known metric $\mathcal{D}$ on arms:

$$|\mu(x) - \mu(y)| \leq \mathcal{D}(x,y) \quad \text{for any two arms } x, y. \tag{4.7}$$

The metric $\mathcal{D}$ can be arbitrary, as far as this problem formulation is concerned. It represents an abstract notion of (known) similarity between arms. Note that w.l.o.g. $\mathcal{D}(x,y) \leq 1$. The set of arms X can be finite or infinite, this distinction is irrelevant to the essence of the problem. While some of the subsequent definitions are more intuitive for infinite X, we state them so that they are meaningful for finite X, too. A problem instance is specified by the metric space $(X, \mathcal{D})$, mean rewards $\mu(\cdot)$, and the time horizon T.

(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:

$$\begin{aligned}
&\mathcal{D}(x,y) \geq 0 && \textit{(non-negativity)}\\
&\mathcal{D}(x,y) = 0 \Leftrightarrow x = y && \textit{(identity of indiscernibles)}\\
&\mathcal{D}(x,y) = \mathcal{D}(y,x) && \textit{(symmetry)}\\
&\mathcal{D}(x,z) \leq \mathcal{D}(x,y) + \mathcal{D}(y,z) && \textit{(triangle inequality).}
\end{aligned}$$

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. A metric space is called finite (resp., infinite) if so is X.

Some notable examples:

- $X = [0,1]^d$, or an arbitrary subset thereof, and the metric 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 are ℓ_1-metric (a.k.a. Manhattan metric) and ℓ_2-metric (a.k.a. Euclidean distance).

- $X = [0,1]$, or an arbitrary subset thereof, and the metric is $\mathcal{D}(x,y) = |x-y|^{1/d}$, $d \geq 1$. In a more succinct notation, this metric space is denoted $\left([0,1],\, \ell_2^{1/d}\right)$.

- The shortest-path metric of a graph: X is the set of nodes of a graph, and $\mathcal{D}(x,y)$ is the length of a shortest path between x and y.

- "Tree distance": X is the set of leaves in a tree with numerical "weights" on internal nodes, and the distance between two leaves is the weight of their least common ancestor (*i.e.,* the root of the smallest subtree containing both leaves). For example, if a node is at depth h (*i.e.,* , h edges away from the root), then its weight could be c^h, for some constant c.

Fixed discretization. The developments in Section 4.1.1 carry over word-by-word, up until (4.3). 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 4.4. 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 4.1.1: 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 (4.3), we obtain

$$\mathbb{E}[R(T)] \leq O\left(\sqrt{(\tfrac{1}{\epsilon})^d \cdot T \log T} + \epsilon T\right) = O\left(T^{\frac{d+1}{d+2}}\, (c \log T)^{\frac{1}{d+2}}\right), \tag{4.8}$$

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 4.5. A subset $S \subset X$ is called an *ϵ-mesh* if every point $x \in X$ is within distance ϵ from some point $y \in S$, in the sense that $\mathcal{D}(x, y) \leq \epsilon$.

It is easy to see that the discretization error of an ϵ-mesh is $\texttt{DE}(S) \leq \epsilon$. In what follows, we characterize the smallest size of an ϵ-mesh, using some notions from the analysis of metric spaces.

Definition 4.6. 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.

Definition 4.7 (covering numbers)**.** An *ϵ-covering* of a metric space is 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* of the metric space and denoted $N_\epsilon(X)$.

The covering number characterizes the "complexity" of a metric space at distance scale ϵ, oin 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 4.8. *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 (4.3). But how do we optimize over all $\epsilon > 0$? For this purpose it is convenient to characterize $N_\epsilon(X)$ for all ϵ simultaneously:

Definition 4.9. The *covering dimension* of X, with multiplier $c > 0$, is

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

In particular, the covering dimension in Example 4.4 is d, with an appropriate multiplier c. Likewise, the covering dimension of metric space $\left([0, 1],\ \ell_2^{1/d}\right)$ is d, for any $d \geq 1$. Note that in the latter example the covering dimension does not need to be integer. The covering dimension of a subset $X' \subset X$ cannot exceed that of X: indeed, $\texttt{COV}_c(X') \leq \texttt{COV}_c(X)$.

Remark 4.10. In the literature, covering dimension is often stated without the multiplier c:

$$\texttt{COV}(X) = \inf_{c>0} \texttt{COV}_c(X).$$

This version is usually meaningful (and more lucid) for infinite metric spaces. However, for finite metric spaces it is trivially 0, so we need to fix the multiplier for a meaningful definition. Furthermore, our definition allows for more precise regret bounds (both for finite and infinite metrics).

We obtain a regret bound by plugging $|S| = N_\epsilon(X) \leq c/\epsilon^d$ into (4.3), and do the same computation as in Example 4.4. Therefore, we obtain the following theorem:

Theorem 4.11 (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*

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

This upper bound is essentially the best possible. A matching lower bound can be achieved for various metric spaces, *e.g.,* for $\left([0,1],\ \ell_2^{1/d}\right)$, $d \geq 1$. It can be proved similarly to Theorem 4.2.

Theorem 4.12. *Consider Lipschitz MAB problem on metric space* $\left([0,1],\ \ell_2^{1/d}\right)$*, for any* $d \geq 1$*, with time horizon* T*. For any algorithm, there exists a problem instance exists a problem instance* $\mathcal{I}$ *such that*

$$\mathbb{E}\left[R(T) \mid \mathcal{I}\right] \geq \Omega(T^{(d+1)/(d+2)}). \tag{4.9}$$

4.3 Adaptive discretization: the Zooming Algorithm

Despite the existence of a matching lower bound, the fixed discretization approach is wasteful. 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. Thinking of arms in S as "probes" that the algorithm places in the metric space. 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 this way if S needs to be chosen in advance.

However, an algorithm could approximately learn the mean rewards 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*. On a very high level, the idea is that we place more probes in regions that could produce better rewards, as far as the algorithm knowns, and less probes in regions which are known to yield only low rewards with high confidence. What can we hope to prove for this algorithm, given the existence of a matching lower bound for fixed discretization? The goal here is to attain the same worst-case regret as in Theorem 4.11, but do "better" on "nice" problem instances. Quantifying what we mean by "better" and "nice" here is an important aspect of the overall research challenge.

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.

4.3.1 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". This is the whole algorithm: we just need to specify the activation rule and the selection rule.

Confidence radius/ball. Fix round t and an arm x that is active at this round. Let $n_t(x)$ is the number of rounds before round t when this arm is chosen, and let $\mu_t(x)$ be the average reward in these rounds. The confidence radius of arm x at time t is defined as

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

Recall that this is essentially the smallest number so as to guarantee with high probability that

$$|\mu(x) - \mu_t(x)| \leq r_t(x).$$

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. We start with some intuition. We will estimate the mean reward $\mu(x)$ of a given active arm x using only the samples from this arm. While samples from "nearby" arms could potentially improve the estimates, this choice simplifies the algorithm and the analysis, and does not appear to worsen the regret bounds. Suppose arm y is not active in round t, and lies very close to some active arm x, in the sense that $\mathcal{D}(x,y) \ll r_t(x)$. Then the algorithm does not have enough samples of x to distinguish x and y. Thus, instead of choosing arm y the algorithm might as well choose arm x. We conclude there is no real need to activate y yet. Going further with this intuition, there is no real need to activate any arm that is covered by the confidence ball of any active arm. We would like to maintain the following 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 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.

Thus, the activation rule is very simple:

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

With this 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 technique from algorithm `UCB1`. If arm x is active at time t, we define

$$\texttt{index}_t(x) = \bar{\mu}_t(x) + 2r_t(x) \tag{4.10}$$

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 upper confidence bound (UCB) on the mean reward, defined as $\texttt{UCB}_t(x) = \mu_t(x) + r_t(x)$. So $\texttt{index}_t(x)$ is very similar, and shares the intuition behind `UCB1`: if $\texttt{index}_t(x)$ is large, then either $\mu_t(x)$ is large, and so x is likely to be a good arm, or $r_t(x)$ is large, so arm x has not been played very often, and should probably be explored more. And the '+' in (4.10) 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.

To summarize, the algorithm is as follows:

```
Initialize: set of active arms $S \leftarrow \emptyset$.
for each round $t = 1, 2, \ldots$ do
    if some arm $y$ is not covered by the confidence balls of active arms then
        pick any such arm $y$ and "activate" it: $S \leftarrow S \cap \{y\}$.
    Play an active arm $x$ with the largest $\texttt{index}_t(x)$.
end
```

Algorithm 4.1: Zooming algorithm for adaptive discretization.

4.3.2 Analysis: clean event

We define a "clean event" $\mathcal{E}$ much like we did in Chapter 1, and prove that it holds with high probability. The proof is more delicate than in Chapter 1, essentially because we cannot immediately take the Union Bound over all of X. The rest of the analysis would simply assume that this event holds.

We consider a $K \times T$ table of realized rewards, with T columns and a row for each arm x. The j-th column for arm x is the reward for the j-th time this arm is chosen by the algorithm. We assume, without loss of generality, that this entire table is chosen before round 1: each cell for a given arm is an independent draw from the reward distribution of this arm. The clean event is defined as a property of this reward table. For each arm x, the clean event is

$$\mathcal{E}_x = \{\ |\mu_t(x) - \mu(x)| \leq r_t(x) \quad \text{for all rounds } t \in [T+1]\ \}.$$

Here $[T] := \{1, 2, \ \ldots\ , T\}$. For convenience, we define $\mu_t(x) = 0$ 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 $\mathcal{E} = \cap_{x \in X} \mathcal{E}_x$.

To simplify the proof of the next claim, we assume that realized rewards take values on a finite set.

Claim 4.13. *Assume that realized rewards take values on a finite set. Then* $\Pr[\mathcal{E}] \geq 1 - \frac{1}{T^2}$.

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.

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

Let N be the total number of arms activated by the algorithm. Define arms $y_j \in X_0$, $j \in [T]$, as follows

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

Here N and y_j's are random variables, the randomness coming from the reward realizations. Note that $\{y_1\,,\,\ldots\,,y_T\}$ is precisely the set of arms activated in a given execution of the algorithm. Since the clean event holds trivially for all arms that are not activated, the clean event can be rewritten as $\mathcal{E} = \cap_{j=1}^{T}\mathcal{E}_{y_j}$. In what follows, we prove that the clean event $\mathcal{E}_{y_j}$ happens with high probability for each $j \in [T]$.

Fix an arm $x \in X_0$ and fix $j \in [T]$. Whether 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 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. Therefore, if $\Pr[y_j = x] > 0$ then

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

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

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

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.

4.3.3 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 4.15), and that each "bad" arm cannot be played too often (Corollary 4.16). As usual, let $\mu^* = \sup_{x\in X}\mu(x)$ be the best reward, and 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 4.14. $\Delta(x) \leq 3\,r_t(x)$ *for each arm x and each round t.*

Proof. Suppose arm x is played in this round. By the covering invariant, the best arm x^* was covered by the confidence ball of some active arm y, *i.e.,* $x^* \in \mathbf{B}_t(y)$. 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 4.15. *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 4.14. $\square$

Corollary 4.16. *For each arm x, we have $n(x) \leq O(\log T)\ \Delta^{-2}(x)$.*

Proof. Use Lemma 4.14 for $t = T$, and plug in the definition of the confidence radius. $\square$

4.3.4 Analysis: covering numbers and 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 4.15, 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 4.16, 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:

$$\begin{aligned} R(T) &\leq \delta T + \sum_{i:\ r=2^{-i}>\delta} R_i(T) \\ &\leq \delta T + \sum_{i:\ r=2^{-i}>\delta} \frac{\Theta(\log T)}{r}\ N_{r/3}(Y_i) \\ &\leq \delta T + O(c \cdot \log T) \cdot (\tfrac{1}{\delta})^{d+1} \end{aligned} \tag{4.11}$$

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

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

The smallest such d is called the *zooming dimension*:

Definition 4.17. For an instance of Lipschitz MAB, the *zooming dimension* with multiplier $c > 0$ is

$$\inf_{d \geq 0} \left\{ N_{r/3}(X) \leq c \cdot r^{-d} \quad \forall r > 0 \right\}.$$

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 4.18. *Consider Lipschitz MAB problem with time horizon T. Assume that realized rewards take values on a finite set. 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 mean rewards. 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 4.18 is appealingly simple, a more precise regret bound is given in (4.11). Since the algorithm does not depend on δ, this bound holds for all $\delta > 0$.

4.4 Bibliographic remarks and further directions

Continuum-armed bandits have been introduced in Agrawal (1995), and further studied in (Kleinberg, 2004; Auer et al., 2007). Uniform discretization has been introduced by Kleinberg (2004) for continuum-armed bandits; Kleinberg et al. (2008) observed that this technique easily extends to Lipschitz bandits. Lipschitz MAB have been introduced in Kleinberg et al. (2008) and in a near-simultaneous and independent paper (Bubeck et al., 2011b). The zooming algorithm is from Kleinberg et al. (2008), see Kleinberg et al. (2019) for a definitive journal version. Bubeck et al. (2011b) present a different algorithm that implements adaptive discretization and obtains similar regret bounds. The lower bound in Theorem 4.12 traces back to Kleinberg (2004).[3] The lower bound statement in Theorem 4.2, with explicit dependence on both L and T, is from Bubeck et al. (2011c). Our presentation roughly follows that in Slivkins (2014) and (Bubeck et al., 2011b).

A line of work that pre-dated and inspired Lipschitz MAB posits that the algorithm is given a "taxonomy" on arms: a tree whose leaves are arms, where arms in the same subtree being "similar" to one another (Kocsis and Szepesvari, 2006; Pandey et al., 2007; Munos and Coquelin, 2007). Numerical similarity information is not revealed. While these papers report successful empirical performance of their algorithms on some examples, they do not lead to non-trivial regret bounds. Essentially, regret scales as the number of arms in the worst case, whereas in Lipschitz MAB regret is bounded in terms of covering numbers.

Covering dimension. Covering dimension is closely related to several other "dimensions", such as Haussdorff dimension, capacity dimension, box-counting dimension, and Minkowski-Bouligand Dimension, that characterize the covering properties of a metric space in fractal geometry (Schroeder, 1991). Covering numbers/dimension have been widely used in machine learning to characterize the complexity of the hypothesis space in classification problems; however, we are not aware of a clear technical connection between this usage and ours. Similar but stronger notions of "dimension" of a metric space have been studied in the theoretical computer science literature, *e.g.,* the ball-growth dimension and the doubling dimension.[4] These

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

[3]Kleinberg (2004) proves a much stronger lower bound which implies Theorem 4.12, and, accordingly, the dependence on T in Theorem 4.2. The construction in this result is overly overly complicated for our needs.

[4]A metric has ball-growth dimension d if doubling the radius of a ball increases the number of points by at most $O(2^d)$ (*e.g.,* Karger and Ruhl, 2002; Abraham and Malkhi, 2005; Slivkins, 2007). A metric has doubling dimension d if any ball can be covered with at most $O(2^d)$ balls of half the radius (*e.g.,* Gupta et al., 2003; Talwar, 2004; Kleinberg et al., 2009).

notions allow for (more) efficient algorithms in many different problems: space-efficient distance representations such as metric embeddings, distance labels, and sparse spanners; network primitives such as routing schemes and distributed hash tables; approximation algorithms for optimization problems such as traveling salesman, k-median, and facility location.

More on the zooming algorithm. The analysis of the zooming algorithm, both in Kleinberg et al. (2008) and in this chapter, goes through without some of the assumptions. First, there is no need to assume that the metric satisfies triangle inequality (although this assumption is useful for the intuition). Second, Lipschitz condition (4.7) only needs to hold for pairs (x, y) such that x is the best arm.[5] Third, no need to restrict realized rewards to finitely many possible values (but one needs a slightly more careful analysis of the clean event). Fourth, no need for a fixed time horizon: The zooming algorithm can achieve the same regret bound for all rounds at once, by an easy application of the "doubling trick" from Section 1.4.

The zooming algorithm attains improved regret bounds for several special cases (Kleinberg et al., 2008). First, if the maximal payoff is near 1. Second, when $\mu(x) = 1 - f(\mathcal{D}(x, S))$, where S is a "target set" that is not revealed to the algorithm. Third, if the realized reward from playing each arm x is $\mu(x)$ plus an independent noise, for several noise distributions; in particular, if rewards are deterministic.

The zooming algorithm achieves near-optimal regret bounds, in a very strong sense (Slivkins, 2014). The "raw" upper bound in (4.11) is optimal up to logarithmic factors, for any algorithm, any metric space, and any given value of this upper bound. Consequently, the upper bound in Theorem 4.18 is optimal, up to logarithmic factors, for any algorithm and any given value d of the zooming dimension that does not exceed the covering dimension. This holds for various metric spaces, *e.g.,* $\left([0,1],\, \ell_2^{1/d}\right)$ and $\left([0,1]^d,\, \ell_2\right)$.

The zooming algorithm, with similar upper and lower bounds, can be extended to the "contextual" version of Lipschitz MAB (Slivkins, 2014), see Sections 8.2 and 8.7 for background and some details.

Per-metric optimality. What is the best regret rate that can be obtained for a given metric space, in the worst case over all problem instances? Instance-independent regret bounds are determined by the "uniform" version of covering dimension: the largest $b \geq 0$ such that $N_\epsilon(X) \geq \Omega(\epsilon^{-b})$ for all $\epsilon > 0$. No algorithm can achieve regret better than $\Omega(T^{(b+1)/(b+2)})$, for any metric space (Bubeck et al., 2011b). (This is a fairly straightforward extension of Theorem 4.12 to covering numbers.) In particular, when b is the covering dimension, as in $\left([0,1],\, \ell_2^{1/d}\right)$, the regret rate in Theorem 4.11 is optimal up to logarithmic factors.

For instance-*independent* regret bounds, the situation is much more complex and interesting. We are interested in regret bounds of the form $C_\mathcal{I} \cdot f(t)$ for all times t, where $C_\mathcal{I}$ depends on the problem instance $\mathcal{I}$ but not on the time t; we abbreviate this as $O_\mathcal{I}(f(t))$. Recall that $O_\mathcal{I}(\log t)$ regret is feasible for finitely many arms. Further, the lower bound in Theorem 4.9 holds even if we allow an instance-dependent constant (Kleinberg, 2004). The proof is much more complicated than what we presented in Section 4.1.2 (for the special case of $d = 1$). Essentially, this is because the randomized problem instance needs to "work" for infinitely many times t, and therefore embeds a "bump function" on every distance scale. To extend this lower bound to arbitrary metric spaces, Kleinberg et al. (2008, 2019) introduce a more refined notion of the covering dimension:

$$\texttt{MaxMinCOV}(X) = \sup_{Y \subset X} \left(\inf_{\text{non-empty } U \subset Y:\ U \text{ is open in } (Y, \mathcal{D})} \texttt{COV}(U) \right),$$

and prove matching upper and lower regret bounds relative to this notion. Their algorithm is a version of the zooming algorithm with quotas on the number of active arms in some regions of the metric space.

[5]The analysis of a similar algorithm in Bubeck et al. (2011b) only needs the Lipschitz condition to hold when both arms are in the vicinity of the best arm.

Further, Kleinberg and Slivkins (2010); Kleinberg et al. (2019) prove that the transition from $O_{\mathcal{I}}(\log t)$ to $O_{\mathcal{I}}(\sqrt{t})$ regret is sharp and corresponds to the distinction between countable and uncountable set of arms. The full characterization of optimal regret rates is summarized in Table 4.1. This line of work makes deep connections between bandit algorithms, metric topology, and transfinite ordinal numbers.

If the metric completion of $(X, \mathcal{D})$ is ...	then regret can be ...	but not ...
finite	$O(\log t)$	$o(\log t)$
compact and countable	$\omega(\log t)$	$O(\log t)$
compact and uncountable		
$\texttt{MaxMinCOV} = 0$	$\tilde{O}(t^\gamma), \gamma > \frac{1}{2}$	$o(\sqrt{t})$
$\texttt{MaxMinCOV} = d \in (0, \infty)$	$\tilde{O}(t^\gamma), \gamma > \frac{d+1}{d+2}$	$o(t^\gamma), \gamma < \frac{d+1}{d+2}$
$\texttt{MaxMinCOV} = \infty$	$o(t)$	$O(t^\gamma), \gamma < 1$
non-compact	$O(t)$	$o(t)$

Table 4.1: Per-metric optimal regret bounds for Lipschitz MAB

Kleinberg and Slivkins (2010); Kleinberg et al. (2019) derive a similar characterization for a version of Lipschitz bandits with full feedback (*i.e.,* when the algorithm receives feedback for all arms). $O_{\mathcal{I}}(\sqrt{t})$ regret is feasible for any metric space of finite covering dimension, and one needs an exponentially more "permissive" version of the covering dimension to obtain more "interesting" regret bounds.

Partial similarity information. Numerical similarity information required for the Lipschitz MAB may be difficult to obtain in practice. A canonical example is the "taxonomy bandits" problem mentioned above, where an algorithm is given a taxonomy (a tree) on arms but not a metric which admits the Lipschitz condition (4.7). From the perspective of Lipschitz MAB, the goal here is to obtain regret bounds that are (almost) as good as if the metric were known.

Slivkins (2011) considers the metric implicitly defined by an instance of taxonomy bandits: the distance between any two arms is the "width" of their least common subtree S, where the width of S is defined as $W(S) := \max_{x,y \in S} |\mu(x) - \mu(y)|$. (Note that $W(S)$ is not known to the algorithm.) This is the best possible metric, *i.e.,* a metric with smallest distances, that admits the Lipschitz condition (4.7). Slivkins (2011) puts forward an extension of the zooming algorithm which partially reconstructs the implicit metric, and almost matches the regret bounds of the zooming algorithm for this metric. In doing so, it needs to deal with *another* exploration-exploitation tradeoff: between learning more about the widths and exploiting this knowledge to run the zooming algorithm. The idea is to have "active subtrees" S rather than "active arms", maintain a lower confidence bound (LCB) on $W(S)$, and use it instead of the true width. The LCB can be obtained any two sub-subtrees S_1, S_2 of S. Indeed, if one chooses arms from S at random according to some fixed distribution, then $W(S) \geq |\mu(S_1) - \mu(S_2)|$, where $\mu(S_i)$ is the expected reward when sampling from S_i, and with enough samples the empirical average reward from S_i is close to its expectation. The regret bound depends on the "quality parameter": essentially, how deeply does one need to look in each subtree S in order to find sub-subtrees S_1, S_2 that give a sufficiently good lower bound on $W(S)$. However, the algorithm does not need to know this parameter upfront. Bull (2015) considers a somewhat more general setting where multiple taxonomies on arms are available, and some of them may work better for this problem than others. He carefully traces out the conditions under which one can achieve $\tilde{O}(\sqrt{T})$ regret.

A similar issue arises when arms correspond to points in $[0,1]$ but no Lipschitz condition is given. This setting can be reduced to "taxonomy bandits" by positing a particular taxonomy on arms, *e.g.,* the root corresponds to $[0, 1]$, its children are $[0, \frac{1}{2})$ and $[\frac{1}{2}, 1]$, and so forth splitting each interval into halves.

Ho et al. (2016) consider a very related problem in the context of crowdsourcing markets. Here the algorithm is an employer who offers a quality-contingent contract to each arriving worker, and adjusts the contract over time. On an abstract level, this is a bandit problem in which arms are contracts: essentially, vectors of prices. However, there is no Lipschitz-like assumption. Ho et al. (2016) treat this problem as a version of "taxonomy bandits", and design a version of the zooming algorithm. They estimate the implicit metric in a problem-specific way, taking advantage of the structure provided by the employer-worker interactions, and avoid the dependence on the "quality parameter" from Slivkins (2011).

Another line of work studies the "pure exploration" version of "taxonomy bandits", where the goal is to output a "predicted best arm" with small instantaneous regret (Munos, 2011; Valko et al., 2013; Grill et al., 2015), see Munos (2014) for a survey. The main result essentially recovers the regret bounds for the zooming algorithm as if a suitable distance function were given upfront. The algorithm posits a parameterized family of distance functions, guesses the parameters, and runs a zooming-like algorithm for each guess.

Bubeck et al. (2011c) study a version of continuum-armed bandits with strategy set $[0, 1]^d$ and Lipschitz constant L that is not revealed to the algorithm, and match the regret rate in Theorem 4.2. This result is powered by an assumption that $\mu(\cdot)$ is twice differentiable, and a bound on the second derivative is known to the algorithm. Minsker (2013) considers the same strategy set, under metric $\|x - y\|_\infty^\beta$, where the "smoothness parameter" $\beta \in (0, 1]$ is not known. His algorithm achieves near-optimal instantaneous regret as if the β were known, under some structural assumptions.

Beyond IID rewards. Several papers consider Lipschitz bandits with non-IID rewards. Kleinberg (2004) a version with arbitrary rewards that satisfy the Lipschitz condition, and proves that a suitable version of uniform discretization matches the regret bound in Theorem 4.11.[6] Maillard and Munos (2010) consider the same problem in the Euclidean space $(\mathbb{R}^d, \ell_2)$. Assuming full feedback, they achieve a surprisingly strong regret bound of $O(\sqrt{T})$, for any constant d. Azar et al. (2014) consider a version in which the IID condition is replaced by more sophisticated ergodicity and mixing assumptions, and essentially recover the performance of the zooming algorithm. Slivkins (2014) handles a version of Lipschitz MAB where expected rewards of each arm change slowly over time.

Other structural models of MAB with similarity. One drawback of Lipschitz MAB as a model is that the distance $\mathcal{D}(x, y)$ only gives a "worst-case" notion of similarity between arms x and y. In particular, the distances may need to be very large in order to accommodate a few outliers, which would make $\mathcal{D}$ less informative elsewhere.[7] With this criticism in mind, Srinivas et al. (2010); Krause and Ong (2011); Desautels et al. (2012) define a probabilistic model, called *Gaussian Processes Bandits*, where the expected payoff function is distributed according to a suitable Gaussian Process on X, thus ensuring a notion of "probabilistic smoothness" with respect to X. Amin et al. (2011) take a different approach and consider multi-armed bandits with an *arbitrary* known structure on reward functions $\mu(\cdot)$. However, their results do not subsume any prior work on Lipschitz MAB.

4.5 Exercises and Hints

Exercise 4.1 (Lower bounds). Consider the lower bound for continuum-armed bandits (Theorem 4.2). Extend the construction and analysis in Section 4.1.2:

(a) from Lipschitz constant $L = 1$ to an arbitrary L.

[6] See Chapter 6 for more background on bandits with arbitrary rewards, and Exercise 6.2 for this particular approach.

[7] This concern is partially addressed by relaxing the Lipschitz condition in the analysis of the zooming algorithm.

(b) from continuum-armed bandits to $\left([0,1],\, \ell_2^{1/d}\right)$, *i.e.,* prove Theorem 4.12.

(c) from $\left([0,1],\, \ell_2^{1/d}\right)$ to an arbitrary metric space: prove the lower bound of $\Omega(T^{(b+1)/(b+2)})$ for any algorithm, any metric space, and any $b \geq 0$ that satisfies $N_\epsilon(X) \geq \Omega(\epsilon^{-b})$ for all $\epsilon > 0$.

Exercise 4.2 (Covering dimension and zooming dimension).

(a) Prove that the covering dimension of $\left([0,1]^d, \ell_2\right)$, $d \in \mathbb{N}$ and $\left([0,1],\, \ell_2^{1/d}\right)$, $d \geq 1$ is d.

(b) Prove that the zooming dimension cannot exceed the covering dimension. More precisely: if $d = \mathtt{COV}_c(X)$, then the zooming dimension with multiplier $3^d \cdot c$ is at most d.

(c) Construct an example in which the zooming dimension is substantially smaller than the covering dimension.

Exercise 4.3 (Lipschitz MAB with a target set). Consider Lipschitz MAB on metric space $(X, \mathcal{D})$ with $\mathcal{D}(\cdot,\cdot) \leq \frac{1}{2}$. Fix the best reward $\mu^* \in [\frac{3}{4}, 1]$ and a subset $S \subset X$ and assume that

$$\mu(x) = \mu^* - \mathcal{D}(x, S) \quad \forall x \in X, \qquad \text{where } \mathcal{D}(x,S) := \inf_{y \in S} \mathcal{D}(x,y).$$

In words, the mean reward is determined by the distance to some "target set" S.

(a) Prove that $\mu^* - \mu(x) \leq \mathcal{D}(x,S)$ for all arms x, and that this condition suffices for the analysis of the zooming algorithm, instead of the full Lipschitz condition (4.7).

(b) Assume the metric space is $([0,1]^d, \ell_2)$, for some $d \in \mathbb{N}$. Prove that the zooming dimension of the problem instance (with a suitably chosen multiplier) is at most the covering dimension of S.

Take-away: In part (b), the zooming algorithm achieves regret $\tilde{O}(T^{(b+1)/(b+2)})$, where b is the covering dimension of the target set S. Note that b could be much smaller than d, the covering dimension of the entire metric space. In particular, one achieves regret $\tilde{O}(\sqrt{T})$ if S is finite.

Chapter 5

Full Feedback and Adversarial Costs *(rev. Sep'17)*

We shift our focus from bandit feedback to full feedback. As the IID assumption makes the problem "too easy", we consider the other extreme, when rewards/costs are adversarially chosen.

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 IID costs are quite "easy" with full feedback, we consider the other extreme: costs can arbitrarily change over time, as if they are selected by an adversary.

The protocol for full feedback and adversarial costs is as follows:

Problem protocol: 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 5.1. While some results rely on bounded costs, *e.g.,* $c_t(a) \in [0, 1]$, we do not assume this by default.

One real-life scenario with full feedback is investments on a stock market. For a particularly simple (albeit very stylized) formulation, suppose each morning choose one stock and invest \$1 into it. At the end of the day, we observe not only the price of the chosen stock, but prices of all stocks. Based on this feedback, we determine which stock to invest for the next day.

A paradigmatic special case of bandits with full feedback 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:

Problem 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 5.2. 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 5.3 (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 1. The $\sqrt{T}$ lower bound follows from the same argument as the bandit lower bound for two arms in Chapter 2, as this argument does not rely on bandit feedback. The $\Omega(\log K)$ lower bound holds for a simple special case, see Theorem 5.7.

5.1 Adversaries and regret

Let us introduce a crucial distinction: an adversary is called *oblivious* if the costs do not depend on the algorithm's choices. Then, w.l.o.g., all costs are chosen before round 1. Otherwise, the adversary is called *adaptive*.

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 total 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

$$R(T) = \texttt{cost}(\texttt{ALG}) - \min_{a \in [K]} \texttt{cost}(a), \tag{5.1}$$

where $\texttt{cost}(\texttt{ALG})$ denotes the total cost incurred by the algorithm. One drawback of this adversary is that it does not model i.i.d. costs, even though we think of i.i.d. rewards as a simple special case of adversarial rewards.

Randomized oblivious adversary. The adversary fixes a distribution $\mathcal{D}$ over the cost tables before round one, and then draws a cost table from this distribution. Then i.i.d. costs are indeed a simple special case. Since $\texttt{cost}(a)$ is now a random variable whose distribution is specified by $\mathcal{D}$, 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 (5.1), and the *foresight regret*[1]

$$R(T) = \texttt{cost}(\texttt{ALG}) - \min_{a \in [K]} \mathbb{E}[\texttt{cost}(a)]. \tag{5.2}$$

For i.i.d. costs, this notion coincides with the definition of regret from Chapter 1.

Remark 5.4. Foresight regret cannot exceed hindsight regret, because the best-arm-in-foresight is a weaker benchmark. Some positive results for foresight regret carry over to hindsight regret, and some don't. For i.i.d. rewards/costs, the $\sqrt{T}$ upper regret bounds from Chapter 1 extend to hindsight regret, whereas the $\log(T)$ upper regret bounds do not extend in full generality, see Exercise 5.2 for details.

Adaptive adversary typically 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.

[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*.

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 9.

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, $\operatorname{argmin}_a \texttt{cost}(a)$, 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 are essential in several important applications to game theory, when a bandit algorithm controls a player in a repeated game (see Chapter 9). Finally, such guarantees often follow from the analysis of oblivious adversary with very little extra work.

The 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.

5.2 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 5.5. *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. □

Remark 5.6. This simple proof 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 that by definition, W_1 is upper-bounded, and W_t does not increase over time. Derive a lower bound on W_T from the existence of a "good expert".
- Connect W_t with the behavior of the algorithm: prove that W_t decreases by a constant factor whenever the algorithm makes mistake / incurs a cost.

The guarantee in Theorem 5.5 is in fact optimal (the proof is left as Exercise 5.1):

Theorem 5.7. *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).

Recall that the majority vote algorithm fully trusts each expert until his first mistake, and completely ignores him afterwards. When all experts may make a mistake, we need a more granular notion of trust. We assign a *confidence weight* $w_a \geq 0$ to each expert a: the higher the weight, the larger the confidence. We update the weights over time, decreasing the weight of a given expert whenever he makes a mistake. More specifically, in this case we multiply the weight by a factor $1 - \epsilon$, for some fixed parameter $\epsilon > 0$. We treat each round as a weighted vote among the experts, and we choose a prediction with a largest total weight. 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 5.1: 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 5.8. *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

$$\frac{W_{T+1}}{W_1} > \frac{(1 - \epsilon)^{\texttt{cost}^*}}{K}. \tag{5.3}$$

Since the weights are non-increasing, we must have

$$W_{t+1} \leq W_t \tag{5.4}$$

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 (5.3) and (5.4), 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 5.8. 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 5.8 with $x = \frac{\epsilon}{1-\epsilon}$. To summarize, we have proved the following theorem.

Theorem 5.9. *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 5.10. 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 $\texttt{cost}^* = 0$.

5.3 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 5.11. *Consider online learning with K experts and 0-1 costs. Any deterministic algorithm has total cost T for some deterministic-oblivious adversary, even if* $\texttt{cost}^* \leq T/K$.

The easy proof is left as Exercise 5.3. Essentially, a deterministic-oblivious adversary just knows what the algorithm is going to do, and can rig the prices accordingly.

Remark 5.12. Thus, the special case of binary prediction with experts advice is much easier for deterministic algorithms. Indeed, it allows for an "approximation ratio" arbitrarily close to 2, as in Theorem 5.9, whereas in the general case the "approximation ratio" cannot be better than K.

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 a fairly natural way). We need to use a different rule to select an arm, because (i) we need this rule to be randomized in order to obtain $o(T)$ regret, and (ii) the weighted majority rule is not even well-defined in the general case of online learning with experts. We use another selection rule, which is also very natural: at each round, choose an arm with probability proportional to the weights. The complete specification is shown in Algorithm 5.2:

```
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 5.2: `Hedge` algorithm for online learning with experts

Below we analyze `Hedge`, and prove $O(\sqrt{T \log K})$ bound on expected hindsight regret. We use the same analysis to derive several important extensions. We break the analysis in several distinct steps, for ease of comprehension.

Remark 5.13. The $O(\sqrt{T \log K})$ regret bound is the best possible for hindsight regret. This can be seen on a simple example in which all costs are i.i.d. with mean $\frac{1}{2}$, see Exercise 5.2(b). Recall that we also have a $\Omega(\sqrt{T})$ bound for foresight regret, due to the lower-bound analysis for two arms in Chapter 2.

As in the previous section, we use the technique outlined in Remark 5.6, with $W_t = \sum_{a=1}^{K} w_t(a)$ being 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

$$W_{T+1} > w_{T+1}(a^*) = (1-\epsilon)^{\texttt{cost}^*}. \tag{5.5}$$

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

Step 2: multiplicative decrease in W_t. We will use polynomial upper bounds for $(1-\epsilon)^x$, $x > 0$.

Fact 5.14. *There exist $\alpha, \beta \geq 0$, possibly dependent on ϵ, such that*

$$(1-\epsilon)^x < 1 - \alpha x + \beta x^2 \quad \textit{for all } x > 0. \tag{5.6}$$

In particular, this holds for a first-order upper bound $(\alpha, \beta) = (\epsilon, 0)$, and for a second-order upper bound with $\alpha = \ln(\frac{1}{1-\epsilon})$ and $\beta = \alpha^2$.

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

$$\begin{aligned}
\frac{W_{t+1}}{W_t} &= \sum_{a\in[K]} (1-\epsilon)^{c_t(a)} \cdot \frac{w_t(a)}{W_t} \\
&< \sum_{a\in[K]} (1 - \alpha\, c_t(a) + \beta\, c_t(a)^2) \cdot p_t(a) \\
&= \sum_{a\in[K]} p_t(a) - \alpha \sum_{a\in[K]} p_t(a)\, c_t(a) + \beta \sum_{a\in[K]} p_t(a)\, c_t(a)^2 \\
&= 1 - \alpha F_t + \beta G_t, \qquad (5.7)
\end{aligned}$$

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

A naive attempt. Using the (5.7), we can obtain:

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

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

Step 3: the telescoping product. Taking a logarithm on both sides of (5.7) and using Fact (5.8), we get

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

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

$$\begin{aligned}
\sum_{t\in[T]} (\alpha F_t - \beta G_t) &< -\sum_{t\in[T]} \ln \frac{W_{t+1}}{W_t} \\
&= -\ln \prod_{t\in[T]} \frac{W_{t+1}}{W_t} \\
&= -\ln \frac{W_{T+1}}{W_1} \\
&= \ln W_1 - \ln W_{T+1} \\
&< \ln K - \ln(1-\epsilon) \cdot \text{cost}^*, \qquad (5.8)
\end{aligned}$$

where we used (5.5) in the last step. Taking expectation on both sides, we obtain:

$$\alpha\,\mathbb{E}[\texttt{cost}(\texttt{ALG})] < \beta \sum_{t\in[T]} \mathbb{E}[G_t] + \ln K - \ln(1-\epsilon)\,\mathbb{E}[\texttt{cost}^*]. \tag{5.9}$$

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

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

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

This yields the main regret bound for Hedge:

Theorem 5.15. *Consider an adaptive adversary such that* $\texttt{cost}^* \leq U$ *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* $U = T$.

Remark 5.16. The same analysis also yields a meaningful performance guarantee which holds with probability 1. Take Equation (5.8) with the same parameters as above: $\alpha = \epsilon = \sqrt{\ln K/2T}$ and $\beta = 0$. If all per-round costs are in $[0,1]$ interval, then Hedge satisfies

$$\textstyle\sum_{t\in[T]} p_t\cdot c_t - \texttt{cost}^* < 2\sqrt{2T\ln K}. \tag{5.10}$$

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 Equation (5.9) with $\alpha = \ln(\frac{1}{1-\epsilon})$ and $\beta = \alpha^2$ to obtain:

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

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

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

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

Lemma 5.17. *Assume we have* $\sum_{t\in[T]}\ \mathbb{E}[G_t] \leq U$ *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:

$$\mathbb{E}[c_t(a)] \leq \mu \text{ and } \mathrm{Var}(c_t(a)) \leq \sigma^2 \text{ for all rounds } t \text{ and all arms } a. \tag{5.11}$$

Then for each round t we have:

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

Thus, we can use Lemma 5.17 with $U = T(\mu^2+\sigma^2)$. We obtain:

Theorem 5.18. *Consider online learning with experts, with a randomized-oblivious adversary. Assume the costs are independent across rounds. Assume upper bound (5.11) for some μ and σ known to the algorithm. Then* `Hedge` *with parameter $\epsilon = \sqrt{\ln K/(3T(\mu^2+\sigma^2))}$ has regret*

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

5.4 Bibliographic remarks and further directions

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.

5.5 Exercises and Hints

Exercise 5.1. Consider binary prediction with expert advice, with a perfect expert. Prove Theorem 5.7: 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 e corresponds to a binary sequence $e \in \{0,1\}^T$, where e_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 5.2 (i.i.d. costs and hindsight regret). Assume i.i.d. costs, as in Remark 5.3.

(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 1 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) Construct a problem instance with a deterministic adversary 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:

$$\mathbb{E}[\min_{a} \texttt{cost}(a)] \leq \tfrac{T}{2} - \Omega(\sqrt{T\,\log K}). \tag{5.12}$$

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.

Take-away: The $O(\sqrt{T\log K})$ regret bound for Hedge is the best possible for hindsight regret. Further, $\log(T)$ upper regret bounds from Chapter 1 do not carry over to hindsight regret in full generality.

(c) Prove that algorithms UCB and Successive Elimination achieve logarithmic regret bound (1.11) even for hindsight regret, assuming that the best-in-foresight arm a^* is unique.

Hint: Under the "clean event", $\texttt{cost}(a) < T\cdot\mu(a)+O(\sqrt{T\log T})$ for each arm $a\neq a^*$, where $\mu(a)$ is the mean cost. It follows that a^* is also the best-in-hindsight arm, unless $\mu(a)-\mu(a^*) < O(\sqrt{T\log T})$ for some arm $a\neq a^*$ (in which case the claimed regret bound holds trivially).

Exercise 5.3. Prove Theorem 5.11: 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 5.9 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 6

Adversarial Bandits *(rev. Jun'18)*

This chapter is concerned with *adversarial bandits*: multi-armed bandits with adversarially chosen costs. In fact, we solve a more general formulation that explicitly includes expert advice. Our algorithm is based on a reduction to the full-feedback problem studied in the previous chapter: it uses algorithm `Hedge` for the full-feedback problem as a subroutine, and its analysis relies on regret bounds that we proved for `Hedge`. We achieve regret bound $\mathbb{E}[R(T)] \leq O\left(\sqrt{KT \log K}\right)$.

For ease of exposition, we focus on deterministic, oblivious adversary: that is, the costs for all arms and all rounds are chosen in advance. Accordingly, we are interested in "hindsight regret", as defined in Equation (5.1). We assume bounded per-round costs: $c_t(a) \leq 1$ for all rounds t and all arms a.

Remark 6.1. Curiously, this upper regret bound not only matches our result for IID bandits (Theorem 1.8), but in fact improves it a little bit, replacing the $\log T$ term with $\log K$. This regret bound is essentially optimal: recall the $\Omega(\sqrt{KT})$ lower bound on regret, derived in Chapter 2.

Recap from Chapter 5. Let us recap the material on the full-feedback problem, reframing it slightly for this chapter. Recall that in the full-feedback problem, the cost of each arm is revealed after every round. A common interpretation 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 *online learning with experts*. We considered a particular algorithm for this problem, called `Hedge`. In each round t, it computes a distribution p_t over experts, and samples an expert from this distribution. We obtained the following regret bound (adapted from Theorem 5.15 and Lemma 5.17):

Theorem 6.2 (`Hedge`)**.** *Consider online learning with N experts. Focus on adaptive adversary and "hindsight regret" $R(T)$ relative to the best observed expert. Algorithm* `Hedge` *with parameter $\epsilon = \epsilon_U := \sqrt{\ln K/(3U)}$ 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) $c_t(e) \leq U$ for all experts e and all rounds t,

(b) $\mathbb{E}[G_t] \leq U$ for all rounds t, where $G_t = \sum_{\text{experts } e} p_t(e)\ c_t^2(e)$.

We will need to distinguish between "experts" in the full-feedback problem and "actions" in the bandit problem. Therefore, we will consistently use "experts" for the former and "actions/arms" for the latter.

6.1 Reduction from bandit feedback to full feedback

Our algorithm for adversarial bandits is a reduction to the full-feedback setting. 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 t, we use the expert e_t chosen by `Hedge` to pick an arm a_t, and define "fake costs" $\hat{c}_t(\cdot)$ on all experts in order to provide `Hedge` with valid inputs. This generic reduction is given below:

```
Given: set $\mathcal{E}$ of experts, parameter $\epsilon \in (0, \frac{1}{2})$ for Hedge.
In each round $t$,
    1. Call Hedge, receive the probability distribution $p_t$ over $\mathcal{E}$.
    2. Draw an expert $e_t$ independently from $p_t$.
    3. Selection rule: use $e_t$ to pick arm $a_t$ (TBD).
    4. Observe the cost $c_t(a_t)$ of the chosen arm.
    5. Define "fake costs" $\hat{c}_t(e)$ for all experts $x \in \mathcal{E}$ (TBD).
    6. Return the "fake costs" to Hedge.
```

Algorithm 6.1: Reduction from bandit feedback to full feedback

Later in this chapter we specify *how* to select arm a_t using expert e_t, and *how* to define fake costs. The former will provide for sufficient exploration, and the latter will ensure that fake costs are unbiased estimates of the true costs.

6.2 Adversarial bandits with expert advice

The reduction defined above suggests a more general problem: what if experts can predict different arms in different rounds? This problem, called *bandits with expert advice*, is one that we will actually solve. 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 separating experts from actions makes the solution clearer. Formally, the problem is defined as follows:

Problem protocol: Adversarial bandits with expert advice

In each round $t \in [T]$:
1. adversary picks cost $c_t(a)$ for each arm a,
2. each expert e recommends an arm $a_{t,e}$,
3. algorithm picks arm a_t and receives the corresponding cost $c_t(a_t)$.

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

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

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 $a_{t,e}$ are also chosen in advance; in other words, experts cannot learn over time. We will have K actions and N experts. The set of all experts is denoted $\mathcal{E}$.

We use the reduction in Algorithm 6.2 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.

This regret bound is essentially the best possible. Specifically, there is a nearly matching lower bound on regret that holds for any given triple of parameters K, T, N:

$$\mathbb{E}[R(T)] \geq \min\left(T,\ \Omega\left(\sqrt{KT\log(N)/\log(K)}\right)\right). \tag{6.1}$$

This lower bound can be proved by an ingenious (yet simple) reduction to the basic $\Omega(\sqrt{KT})$ lower regret bound for bandits, see Exercise 6.1.

6.3 Preliminary analysis: unbiased estimates

We have two notions of "cost" on experts. For each expert e at round t, we have the true cost $c_t(e) = c_t(a_{t,e})$ determined by the predicted arm $a_{t,e}$, and the *fake cost* $\hat{c}_t(e)$ that is computed inside the algorithm and then fed to Hedge. Thus, our regret bounds for Hedge refer to the *fake regret* defined relative to the fake costs:

$$\widehat{R}_{\texttt{Hedge}}(T) = \widehat{\texttt{cost}}(\texttt{Hedge}) - \min_{e\in\mathcal{E}} \widehat{\texttt{cost}}(e),$$

where $\widehat{\texttt{cost}}(\texttt{Hedge})$ and $\widehat{\texttt{cost}}(e)$ are the total fake costs for Hedge and expert e, respectively.

We want the 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 ensure that

$$\mathbb{E}[\hat{c}_t(e) \mid \vec{p}_t] = c_t(e) \quad \text{for all experts } e, \tag{6.2}$$

where $\vec{p}_t = (p_t(e):\ \text{all experts } e)$. We use this as follows:

Claim 6.3. *Assuming Equation (6.2), it holds that* $\mathbb{E}[R_{\texttt{Hedge}}(T)] \leq \mathbb{E}[\widehat{R}_{\texttt{Hedge}}(T)]$.

Proof. First, we connect true costs of Hedge with the corresponding fake costs.

$$\begin{aligned}\mathbb{E}[\hat{c}_t(e_t) \mid \vec{p}_t] &= \textstyle\sum_{e\in\mathcal{E}} \Pr[e_t = e \mid \vec{p}_t]\ \mathbb{E}[\hat{c}_t(e) \mid \vec{p}_t] \\ &= \textstyle\sum_{e\in\mathcal{E}} p_t(e)\ c_t(e) \qquad \textit{(use definition of } p_t(e) \textit{ and Equation (6.2))} \\ &= \mathbb{E}[c_t(e_t) \mid \vec{p}_t].\end{aligned}$$

Taking expectation of both sides, $\mathbb{E}[\hat{c}_t(e_t)] = \mathbb{E}[c_t(e_t)]$. Summing over all rounds, it follows that

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

To complete the proof, we deal with the benchmark:

$$\mathbb{E}[\min_{e\in\mathcal{E}} \widehat{\texttt{cost}}(e)] \leq \min_{e\in\mathcal{E}} \mathbb{E}[\widehat{\texttt{cost}}(e)] = \min_{e\in\mathcal{E}} \mathbb{E}[\texttt{cost}(e)] = \min_{e\in\mathcal{E}} \texttt{cost}(e).$$

The first equality holds by (6.2), and the second equality holds because true costs $c_t(e)$ are deterministic. □

Remark 6.4. This proof used the "full power" of assumption (6.2). A weaker assumption $\mathbb{E}[\hat{c}_t(e)] = \mathbb{E}[c_t(e)]$ would not have sufficed to argue about true vs. fake costs of Hedge.

6.4 Algorithm `Exp4` and crude analysis

To complete the specification of Algorithm 6.2, we need to define fake costs $\hat{c}_t(\cdot)$ and specify how to choose an arm a_t. For fake costs, we will use a standard trick in statistics called *Inverse Propensity Score* (IPS). Whichever way arm a_t is chosen in each round t given the probability distribution $\vec{p}_t$ over experts, this defines distribution q_t over arms:

$$q_t(a) := \Pr[a_t = a \mid \vec{p}_t] \quad \text{for each arm } a.$$

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

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

The fake cost on each expert e is defined as the fake cost of the arm chosen by this expert: $\hat{c}_t(e) = \hat{c}_t(a_{t,e})$.

Remark 6.5. Algorithm 6.2 can use fake costs as defined above as long as it can compute probability $q_t(a_t)$.

Claim 6.6. *Equation (6.2) holds if $q_t(a_{t,e}) > 0$ for each expert e.*

Proof. Let us argue about each arm a separately. If $q_t(a) > 0$ then

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

For a given expert e plug in arm $a = a_{t,e}$, its choice in round t. □

So, if an arm a is selected by some expert in a given round t, the selection rule needs to choose this arm with non-zero probability, regardless of which expert is actually chosen by `Hedge` and what is this expert's recommendation. Further, if probability $q_t(a)$ is sufficiently large, then one can upper-bound fake costs, and consequently apply Theorem 6.2(a). On the other hand, we would like to follow the chosen expert e_t most of the time, so as to ensure low costs. A simple and natural way to achieve both objectives is to follow e_t with probability $1 - \gamma$, for some small $\gamma > 0$, and with the remaining probability choose an arm uniformly at random. This completes the specification of our algorithm, which is known as `Exp4`. For clarity, we recap the full specification in Algorithm 6.2.

Note that $q_t(a) \geq \gamma/K > 0$ for each arm a. According to Claim 6.6 and Claim 6.3, the expected true regret of `Hedge` is upper-bounded by its expected fake regret: $\mathbb{E}[R_{\texttt{Hedge}}(T)] \leq \mathbb{E}[\widehat{R}_{\texttt{Hedge}}(T)]$.

Remark 6.7. Fake costs $\hat{c}_t(\cdot)$ depend on the probability distribution $\hat{p}_t$ chosen by `Hedge`. This distribution depends on the actions selected by `Exp4` in the past, and these actions in turn depend on the experts chosen by `Hedge` in the past. To summarize, fake costs depend on the experts chosen by `Hedge` in the past. So, fake costs do not form an oblivious adversary, as far as `Hedge` is concerned. Thus, we need regret guarantees for `Hedge` against an adaptive adversary, even though the true costs are chosen by an oblivious adversary.

In each round t, our algorithm accumulates cost at most 1 from the low-probability exploration, and cost $c_t(e_t)$ from the chosen expert e_t. So the expected cost in this round is $\mathbb{E}[c_t(a_t)] \leq \gamma + \mathbb{E}[c_t(e_t)]$. Summing over all rounds, we obtain:

$$\begin{aligned} \mathbb{E}[\texttt{cost}(\texttt{Exp4})] &\leq \mathbb{E}[\texttt{cost}(\texttt{Hedge})] + \gamma T. \\ \mathbb{E}[R_{\texttt{Exp4}}(T)] &\leq \mathbb{E}[R_{\texttt{Hedge}}(T)] + \gamma T \leq \mathbb{E}[\widehat{R}_{\texttt{Hedge}}(T)] + \gamma T. \end{aligned} \tag{6.3}$$

```
Given: set $\mathcal{E}$ of experts, parameter $\epsilon \in (0, \frac{1}{2})$ for Hedge, exploration parameter $\gamma \in [0, \frac{1}{2})$.
In each round $t$,
    1. Call Hedge, receive the probability distribution $p_t$ over $\mathcal{E}$.
    2. Draw an expert $e_t$ independently from $p_t$.
    3. Selection rule: with probability $1 - \gamma$ follow expert $e_t$; else pick an arm $a_t$ uniformly at random.
    4. Observe the cost $c_t(a_t)$ of the chosen arm.
    5. Define fake costs for all experts $e$:
           $\hat{c}_t(e) = \begin{cases} \frac{c_t(a_t)}{\Pr[a_t = a_{t,e} | \vec{p}_t]} & a_t = a_{t,e}, \\ 0 & \text{otherwise.} \end{cases}$
    6. Return the "fake costs" $\hat{c}(\cdot)$ to Hedge.
```
Algorithm 6.2: Algorithm Exp4 for adversarial bandits with experts advice

Equation (6.3) quantifies the sense in which the regret bound for Exp4 reduces to the regret bound for Hedge.

We can immediately derive a crude regret bound via Theorem 6.2(a). Indeed, observe that $\hat{c}_t(a) \leq 1/q_t(a) \leq K/\gamma$. So we can take Theorem 6.2(a) with $U = K/\gamma$, and deduce that

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

To minimize expected regret, chose γ so as to approximately equalize the two summands. We obtain the following theorem:

Theorem 6.8. *Consider adversarial bandits with expert advice, with a deterministic-oblivious adversary. Algorithm* Exp4 *with parameters $\gamma = T^{-1/3}\; (K \log N)^{1/3}$ and $\epsilon = \epsilon_U$, $U = K/\gamma$, achieves regret*

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

Remark 6.9. We did not use any property of Hedge other than the regret bound in Theorem 6.2(a). Therefore, Hedge can be replaced with any other full-feedback algorithm with the same regret bound.

6.5 Improved analysis of Exp4

We obtain a better regret bound by analyzing the quantity

$$\widehat{G}_t := \sum_{e \in \mathcal{E}} p_t(e)\; \hat{c}_t^2(e).$$

We prove an upper bound $\mathbb{E}[G_t] \leq \frac{K}{1-\gamma}$, and use the corresponding regret bound for Hedge, Theorem 6.2(b) with $U = \frac{K}{1-\gamma}$. Whereas the crude analysis presented above used Theorem 6.2 with $U = \frac{K}{\gamma}$.

Remark 6.10. This analysis extends to $\gamma = 0$. In other words, the uniform exploration step in the algorithm is not necessary. While we previously used $\gamma > 0$ to guarantee that $q_t(a_{t,e}) > 0$ for each expert e, the same conclusion also follows from the fact that Hedge chooses each expert with a non-zero probability.

Lemma 6.11. *Fix parameter $\gamma \in [0, \frac{1}{2})$ and round t. Then $\mathbb{E}[G_t] \leq \frac{K}{1-\gamma}$.*

Proof. For each arm a, let $\mathcal{E}_a = \{e \in \mathcal{E} : a_{t,e} = a\}$ be the set of all experts that recommended this arm. Let

$$p_t(a) := \sum_{e \in \mathcal{E}_a} p_t(e)$$

be the probability that the expert chosen by `Hedge` recommends arm a. Then

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

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

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

Each realization of $\widehat{G}_t$ satisfies:

$$\begin{aligned}
\widehat{G}_t &:= \sum_{e \in \mathcal{E}} p_t(e)\, \hat{c}_t^2(e) \\
&= \sum_a \sum_{e \in \mathcal{E}_a} p_t(e) \cdot \hat{c}_t(e) \cdot \hat{c}_t(e) && \textit{(re-write as a sum over arms)} \\
&\leq \sum_a \sum_{e \in \mathcal{E}_a} \frac{p_t(e)}{(1-\gamma)\, p_t(a)}\, \hat{c}_t(a) && \textit{(replace one } \hat{c}_t(a) \textit{ with an upper bound)} \\
&= \frac{1}{1-\gamma} \sum_a \frac{\hat{c}_t(a)}{p_t(a)} \sum_{e \in \mathcal{E}_a} p_t(e) && \textit{(move “constant terms” out of the inner sum)} \\
&= \frac{1}{1-\gamma} \sum_a \hat{c}_t(a) && \textit{(the inner sum is just } p_t(a))
\end{aligned}$$

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

Let us complete the analysis, being slightly careful with the multiplicative constant in the regret bound:

$$\begin{aligned}
\mathbb{E}[\widehat{R}_{\texttt{Hedge}}(T)] &\leq 2\sqrt{3/(1-\gamma)} \cdot \sqrt{TK \log N} \\
\mathbb{E}[R_{\texttt{Exp4}}(T)] &\leq 2\sqrt{3/(1-\gamma)} \cdot \sqrt{TK \log N} + \gamma T && \textit{(by Equation (6.3))} \\
&\leq 2\sqrt{3} \cdot \sqrt{TK \log N} + 2\gamma T && \textit{(since } \sqrt{1/(1-\gamma)} \leq 1+\gamma) \qquad (6.4)
\end{aligned}$$

(To derive (6.4), 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 6.12. *Consider adversarial bandits with expert advice, with a deterministic-oblivious adversary. Algorithm* `Exp4` *with parameters $\gamma \in [0, \frac{1}{2T})$ and $\epsilon = \epsilon_U$, $U = \frac{K}{1-\gamma}$, achieves regret*

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

6.6 Bibliographic remarks and further directions

`Exp4` stands for **exp**loration, **exp**loitation, **exp**onentiation, and **exp**erts. The specialization to 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. Their analysis is presented in various books and courses on online learning (e.g., Cesa-Bianchi and Lugosi, 2006; Bubeck and Cesa-Bianchi, 2012). Our presentation was most influenced by (Kleinberg, 2007, Week 8), but the reduction to `Hedge` is made more explicit.

The lower bound (6.1) for adversarial bandits with expert advice is due to Agarwal et al. (2012). We used a slightly simplified construction from Seldin and Lugosi (2016) in the hint for Exercise 6.1.

Running time. 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. Good regret *and* good running time can be obtained for some important special cases with a large N. It suffices to replace `Hedge` with a different algorithm for online learning with experts which satisfies one or both regret bounds in Theorem 6.2. We follow this approach in the next chapter.

The amazing power of `Exp4`. `Exp4` can be applied in many different settings:

- *contextual bandits:* we will see this application in Chapter 8.
- *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 6.12, we obtain
$$\mathbb{E}[R_S(T)] = O(\sqrt{KST \cdot \log(KT)}). \tag{6.5}$$
- *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 compete with the (cost of) the best *current* arm: $c_t^* = \min_a c_t(a)$. More formally, we are interested in *dynamic regret*, defined as
$$R^*(T) = \min(\texttt{ALG}) - \textstyle\sum_{t\in[T]} c_t^*.$$
(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.

Extensions. Much research has been done on various extensions of adversarial bandits. Let us briefly discuss some of these extensions:

- an algorithm that obtains a similar regret bound for adaptive adversaries (against the best observed arm), and high probability (Algorithm `EXP3.P.1` in the original paper Auer et al. (2002b)).

- an algorithm with $O(\sqrt{KT})$ regret – shaving off the $\sqrt{\log K}$ 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 (6.5) via a modification of Exp3 (and a more involved analysis), with essentially the same running time as Exp3.

- 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, one improve over a "naive" application of Exp4 or S-shifting regret algorithms. Slivkins and Upfal (2008); Slivkins (2014) provide algorithms with better bounds on dynamic regret and fast running times. These algorithms 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.

- *Data-dependent* regret bounds are near-optimal in the worst case, and get better if the realized costs are, in some sense, "nice". Hazan and Kale (2011) obtain an improved regret bound when the realized cost of each arm a does not change too much compared to its average $\texttt{cost}(a)/T$. Their 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. However, a seemingly simple special case of i.i.d. 0-1 costs is essentially the worst case for this regret bound. Wei and Luo (2018); Bubeck et al. (2018) obtain further data-dependent regret bounds that take advantage of, respectively, small path-lengths $\sum_t |c_t(a) - c_{t-1}(a)|$ and sparse costs.

- A related but different direction concerns algorithms that work well for both adversarial bandits and bandits with i.i.d. costs. Bubeck and Slivkins (2012) achieved the *best-of-both-worlds* result: an algorithm that essentially matches the regret of Exp3 in the worst case, and achieves logarithmic regret, like UCB1, if the costs are actually i.i.d. This direction has been continued in (Seldin and Slivkins, 2014; Auer and Chiang, 2016; Seldin and Lugosi, 2017; Lykouris et al., 2018; Wei and Luo, 2018). The goals in this line of work included refining and optimizing the regret bounds, obtaining the same high-level results with a more practical algorithm, and improving the regret bound for the adversarial case if the adversary only "corrupts" a small number of rounds.

6.7 Exercises and Hints

Exercise 6.1 (lower bound). Consider adversarial bandits with experts advice. Prove the lower bound in (6.1) for any given (K, N, T). More precisely: construct a randomized problem instance for which any algorithm satisfies (6.1).

Hint: Split the time interval $1..T$ into $M = \frac{\ln N}{\ln K}$ non-overlapping sub-intervals of duration T/M . For each sub-interval, construct the randomized problem instance from Chapter 2 (independently across the sub-intervals). Each expert recommends the same arm within any given sub-interval; the set of experts includes all experts of this form.

Exercise 6.2 (fixed discretization). Let us extend the fixed discretization approach from Chapter 4 to adversarial bandits. Consider adversarial bandits with the set of arms $\mathcal{A} = [0, 1]$. Fix $\epsilon > 0$ and let S_ϵ be the ϵ-uniform mesh over $\mathcal{A}$, *i.e.,* the set of all points in $[0, 1]$ that are integer multiples of ϵ. For a subset $S \subset \mathcal{A}$, the optimal total cost is $\texttt{cost}^*(S) := \min_{a\in S} \texttt{cost}(a)$, and the (time-averaged) discretization error is defined as

$$\texttt{DE}(S_\epsilon) = \frac{\texttt{cost}^*(S)-\texttt{cost}^*(\mathcal{A})}{T}.$$

(a) Prove that $\texttt{DE}(S_\epsilon) \leq L\epsilon$, assuming Lipschitz property:

$$|c_t(a) - c_t(a')| \leq L \cdot |a - a'| \quad \text{for all arms } a, a' \in \mathcal{A} \text{ and all rounds } t. \tag{6.6}$$

(b) Consider dynamic pricing,[1] where the values $v_1 , \ldots , v_T$ are chosen by a deterministic, oblivious adversary. Prove that $\texttt{DE}(S_\epsilon) \leq \epsilon$.

Note: It is a special case of adversarial bandits, with some extra structure. Due to this extra structure, we can bound discretization error *without assuming Lipschitzness*.

(c) Assume that $\texttt{DE}(S_\epsilon) \leq \epsilon$ for all $\epsilon > 0$. Obtain an algorithm with regret $\mathbb{E}[R(T)] \leq O(T^{2/3} \log T)$.

Hint: Use algorithm Exp3 with arms $S \subset \mathcal{A}$, for a well-chosen subset S.

Exercise 6.3 (slowly changing costs). Consider a randomized oblivious adversary such that the expected cost of each arm changes by at most ϵ from one round to another, for some fixed and known $\epsilon > 0$. Use algorithm Exp4 to obtain dynamic regret

$$\mathbb{E}[R^*(T)] \leq O(T) \cdot (\epsilon K \log KT)^{1/3}. \tag{6.7}$$

Note: Regret bounds for dynamic regret are typically of the form $\mathbb{E}[R^*(T)] \leq C \cdot T$, where C is a "constant" determined by K and the parameter(s). The intuition here is that the algorithm pays a constant per-round "price" for keeping up with the changing costs. The goal here is to make C smaller, as a function of K and ϵ.

Hint: Recall the application of Exp4 to n-shifting regret, denote it Exp4(n). Let $\texttt{OPT}_n = \min \texttt{cost}(\pi)$, where the min is over all n-shifting policies π, be the benchmark in n-shifting regret. Analyze the "discretization error": the difference between $\texttt{OPT}_n$ and $\texttt{OPT}^* = \sum_{t=1}^{T} \min_a c_t(a)$, the benchmark in dynamic regret. Namely: prove that $\texttt{OPT}_n - \texttt{OPT}^* \leq O(\epsilon T^2/n)$. Derive an upper bound on dynamic regret that is in terms of n. Optimize the choice of n.

[1]In each round t, algorithm chooses a price $a_t \in [0, 1]$, and offers one unit of good for this price. A customer arrives having a value $v_t \in [0, 1]$ in mind, and buys if and only if $v_t \geq p_t$. Cost is $-p_t$ if there is a sale, 0 otherwise. (This problem is more naturally stated in terms of rewards rather than costs, but we go with costs for consistency.)

Chapter 7

Linear Costs and Combinatorial Actions *(rev. Jun'18)*

[TODO: add bibnotes.]

We study bandit problems with linear costs: actions are represented by vectors in a low-dimensional real space, and action costs are linear in this representation. 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*. We start with an important special case, online routing problem, and its generalization, combinatorial semi-bandits. We solve both using a version of the bandits-to-Hedge reduction from the previous chapter. Then we 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. It plugs nicely into the same reduction, yielding a substantial improvement over Hedge.

Throughout this chapter, the setting is as follows. There are K actions and a fixed time horizon T. Each action a yields cost $c_t(a)$ at each round t. 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.

7.1 Bandits-to-experts reduction, revisited

Let us recap some material from the previous two chapters, in a shape that is convenient for this chapter. Regret is defined as the total cost of the algorithm minus the minimum total cost of an action:

$$R(T) = \texttt{cost}(\texttt{ALG}) - \min_a \texttt{cost}(a), \qquad \text{where } \texttt{cost}(a) = \Sigma_{t=1}^{T} c_t(a).$$

We are interested in regret bounds when all action costs are upper-bounded by a given number U: $c_t(a) \leq U$ for all actions a and all rounds t. We call such scenario a *U-bounded adversary*. For full feedback, algorithm Hedge achieves the following regret bound against an adaptive, U-bounded adversary:

$$\mathbb{E}[R(T)] \leq \theta(\sqrt{UT \log K}). \tag{7.1}$$

This holds for any $U > 0$ that is known to the algorithm.

We also defined a "reduction" from bandit feedback to full feedback. This reduction takes an arbitrary full-feedback algorithm (*e.g.,* `Hedge`), and uses it as a subroutine to construct a bandit algorithm. The reduction creates an expert for each arm, so that this expert always recommends this arm. We present this reduction in a slightly more abstract way than in the previous chapter, as shown in Algorithm 7.1. 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: an algorithm `ALG` for online learning with experts, and parameter $\gamma \in (0, \frac{1}{2})$.
In each round t:

1. call `ALG`, receive an expert x_t chosen for this round
   (x_t is an independent draw 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 7.1: Reduction from bandit feedback to full feedback.

Theorem 7.1. *Consider Algorithm 7.1 with algorithm* `ALG` *that achieves regret bound* $\mathbb{E}[R(T)] \leq f(T, K, U)$ *against adaptive, U-bounded adversary, for any given $U > 0$ that is known to the algorithm.*

Consider adversarial bandits with a deterministic, oblivious adversary. Assume "fake costs" satisfy

$$\mathbb{E}[\hat{c}_t(x)|p_t] = c_t(x) \text{ and } \hat{c}_t(x) \leq U/\gamma \qquad \text{for all experts } x \text{ and all rounds } t,$$

where U is some number that is known to the algorithm. Then Algorithm 7.1 achieves regret

$$\mathbb{E}[R(T)] \leq f(T, K, U/\gamma) + \gamma T.$$

In particular, if `ALG` *is* `Hedge` *and $\gamma = T^{-1/3} \cdot (U \log K)^{1/3}$, Algorithm 7.1 achieves regret*

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

We instantiate this algorithm, *i.e.,* specify the missing pieces, to obtain a solution for some special cases of linear bandits that we define below.

7.2 Online routing problem

Let us consider an important special case of linear bandits called the *online routing problem*, a.k.a. *online shortest paths*. We are given a graph G with d edges, a source node u, and a destination node v. 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. In each round, an algorithm chooses among "actions" that correspond to u-v paths in the graph. Informally, the algorithm's goal in each round is to find the "best route" from u to v: an u-v path with minimal cost (*i.e.,* minimal travel time). More formally, the problem is as follows:

Problem protocol: Online routing problem

Given: graph G, source node u, destination node v.
For each round $t \in [T]$:

1. Adversary chooses costs $c_t(e) \in [0,1]$ for all edges e.
2. Algorithm chooses u-v-path $a_t \subset \texttt{Edges}(G)$.
3. Algorithm incurs cost $c_t(a_t) = \sum_{e \in a_t} a_e \cdot c_t(e)$ and receives feedback.

To cast this problem as a special case of "linear bandits", ntoe 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 $c_t(a) = a \cdot v_t = \sum_{e \in [d]} a_e \, c_t(e)$.

There are three versions of the problem, depending on which feedback is received:

bandit feedback only $c_t(a_t)$ is observed;

semi-bandit feedback costs $c_t(e)$ for all edges $e \in a_t$ are observed;

full feedback costs $c_t(e)$ for all edges e are observed.

The full-feedback version can be solved with the Hedge algorithm. Applying regret bound (7.1) with a trivial upper bound $U = d$ on the action costs, we obtain regret $\mathbb{E}[R(T)] \leq O(d\sqrt{T})$. This regret bound is optimal up to constant factors (Koolen et al., 2010). That the root-T dependence on T is optimal can be seen immediately from Chapter 2. An important drawback of using Hedge for this problem is the running time, which is exponential in d. We return to this issue in Section 7.4.

Online routing with semi-bandit feedback. In the remainder of this section we focus on the semi-bandit version. We use the bandit-to-experts reduction (Algorithm 7.1) with Hedge algorithm, as a concrete and simple application of this machinery to linear bandits. We assume that the costs are selected by a deterministic oblivious adversary, and we do not worry about the running time.

As a preliminary attempt, we can use Exp3 algorithm for this problem. However, expected regret would be proportional to square root of the number of actions, which in this case may be exponential in d.

Instead, we seek a regret bound of the form:

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

To this end, we use the reduction (Algorithm 7.1) with Hedge. The "extra information" in the reduction is the semi-bandit feedback. Recall that we also need to specify the "random exploration" and the "fake costs".

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. We assume that each edge e belongs to some u-v path $a^{(e)}$; this is without loss of generality, because otherwise we can just remove this edge from the graph.

We define fake costs for each edge e separately; the fake cost of a path is simply the sum of fake costs over its edges. 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 $\Pr[\Lambda_{t,e}] = \gamma/d$. The fake cost on edge e is

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{\gamma/d} & \text{if event } \Lambda_{t,e} \text{ happens} \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

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 7.1 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 7.2. *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 7.3. 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 7.1 with a different algorithm `ALG` for online learning with experts, it suffices to have an upper bound on regret against an oblivious adversary.

7.3 Combinatorial semi-bandits

The online routing problem with semi-bandit feedback is a special case of *combinatorial semi-bandits*, where edges are replaced with d "atoms", and u-v paths are replaced with feasible subsets of atoms. The family of feasible subsets can be arbitrary (but it is known to the algorithm).

Problem protocol: Combinatorial semi-bandits

Given: set S of atoms, and a family $\mathcal{F}$ of feasible actions (subsets of S).
For each round $t \in [T]$:

1. Adversary chooses costs $c_t(e) \in [0, 1]$ for all atoms e,
2. Algorithm chooses a feasible action $a_t \in \mathcal{F}$,
3. Algorithm incurs cost $c_t(a_t) = \sum_{e \in a_t} a_e \cdot c_t(e)$ and observes costs $c_t(e)$ for all atoms $e \in a_t$.

The algorithm and analysis from the previous section does not rely on any special properties of u-v paths. Thus, they carry over word-by-word to combinatorial semi-bandits, replacing edges with atoms, and u-v paths with feasible subsets. We obtain the following theorem:

Theorem 7.4. *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:

- *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 can encode various 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 depend on both the ad and on the user. Mathematically, the problem is very similar to the news articles: rounds correspond to users, atoms are the ads, and feasible subsets can encode constraints on which ads can or cannot be shown together. The difference is that the payments are no longer 0-1.
- *A slate of news articles:* Similar to the news articles problem, but the ordering of the articles on the webpage matters. This 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 each "atom" 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 several motivating examples, and allows for a unified solution. 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 7.5. Solving *combinatorial bandits* – the same problem with bandit feedback – requires more work. The main challenge is that we need to estimate fake costs for all atoms in the chosen action, whereas we only observe the total cost for the action. One solution is to construct a suitable *basis*: a subset of feasible actions (called *base actions*) such that each action can be represented as a linear combination of the base actions. Then we can use a version of Algorithm 7.1 where in the "random exploration" step we chose uniformly among the base actions. This gives us fake costs on the base actions. Then fake cost on each atom can be defined as the corresponding linear combination over the base actions. This approach works as long as the linear coefficients are small, and ensuring this property takes some work. This approach is worked out in Awerbuch and Kleinberg (2008), resulting in regret $\mathbb{E}[R(T)] \leq \tilde{O}(d^{10/3} \cdot T^{2/3})$.

Low regret *and* running time. Recall that `AlgSB` is slow: its running time per round is exponential in d, as it relies on `Hedge` with this many experts. We would like the running time per round be *polynomial* in d.

One should not hope to accomplish this in the full generality of combinatorial bandits. Indeed, even if the costs on all atoms were known, choosing the best feasible action (a feasible subset of minimal cost) is a well-known problem of *combinatorial optimization*, which is NP-hard. However, combinatorial optimization allows for polynomial-time solutions in many interesting special cases. For example, in the online routing problem discussed above the corresponding combinatorial optimization problem is a well-known shortest-path problem. Thus, a natural approach is to assume that we have access to an *optimization 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.

In Section 7.4, we use this oracle to construct a new algorithm for combinatorial bandits with full feedback, called *Follow the Perturbed Leader* (`FPL`). In each round, this algorithm inputs only the costs on the atoms, and makes only one oracle call. We derive a regret bound

$$\mathbb{E}[R(T)] \leq O(U\sqrt{dT}) \tag{7.3}$$

against an oblivious, U-bounded adversary such that the atom costs are at most $\frac{U}{d}$. (Recall that a regret bound against an oblivious adversary suffices for our purposes, as per Remark 7.3.)

We use algorithm `AlgSB` as before, but replace `Hedge` with `FPL`; call the new algorithm `AlgSBwithFPL`. The analysis from Section 7.2 carries over to `AlgSBwithFPL`. We take $U = d^2/\gamma$ as a known upper bound on the fake costs of actions, and note that the fake costs of atoms are at most $\frac{U}{d}$. Thus, we can apply regret bound (7.3) for `FPL` with fake costs, and obtain regret

$$\mathbb{E}[R(T)] \leq O(U\sqrt{dT}) + \gamma T.$$

Optimizing the choice of parameter γ, we immediately obtain the following theorem:

Theorem 7.6. *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 7.7. 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 (7.2) is known in advance.

Alternatively, we could have defined fake costs on atoms e as

$$\hat{c}_t(e) = \begin{cases} \frac{c_t(e)}{\Pr[e \in a_t \mid p_t]} & \text{if } e \in a_t \\ 0 & \text{otherwise.} \end{cases}$$

This definition leads to essentially the same regret bound (and, in fact, is somewhat better in practice). However, computing the probability $\Pr[e \in a_t \mid p_t]$ in a brute-force way requires iterating over all actions, which leads to running times exponential in d, similar to `Hedge`.

7.4 Follow the Perturbed Leader

Let us turn our attention to the full-feedback problem with linear costs. We do not restrict ourselves to combinatorial actions, and instead allow an arbitrary subset $\mathcal{A} \subset [0,1]^d$ of feasible actions. This subset is fixed over time and known to the algorithm. We posit an upper bound on the costs: we assume that the hidden vector v_t satisfies $v_t \in [0, U/d]^d$, for some known parameter U, so that $c_t(a) \leq U$ for each action a.

We design an algorithm, called Follow the Perturbed Leader (`FPL`), that is computationally efficient and satisfies regret bound (7.3). In particular, this suffices to complete the proof of Theorem 7.6.

We assume than the algorithm has access to 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. The implementation of the oracle is domain-specific, and is irrelevant to our analysis. We prove the following theorem:

Theorem 7.8. *Assume that $v_t \in [0, U/d]^d$ for some known parameter U. Algorithm* `FPL` *achieves regret $\mathbb{E}[R(T)] \leq 2U \cdot \sqrt{dT}$. The running time in each round is polynomial in d plus one call to the oracle.*

Remark 7.9. The set of feasible actions $\mathcal{A}$ can be infinite, as long as a suitable 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. Whereas Hedge is not even well-defined for 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.

Follow the leader. 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.

While this approach works fine for i.i.d. costs, it breaks for adversarial costs. 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 a way that harms the algorithm. In fact, this can be done to any deterministic online learning algorithm, as per Theorem 5.11. For concreteness, consider 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

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

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, $T/2$.

Fix: perturb the history! Let us use randomization to side-step the synchronization issue discussed above. We perturb the history before handing it to the oracle. 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 $v_{0:t-1}$:

$$a_{t+1} = M(v_{0:t}).$$

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 $[-\frac{1}{\epsilon}, \frac{1}{\epsilon}]$. The parameter ϵ can be tuned according to T, U, and d; in the end, we use $\epsilon = \frac{\sqrt{d}}{U\sqrt{T}}$.

7.4.1 Analysis of the algorithm

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 online learning with experts; this is 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 7.10. *For each value of parameter $\epsilon > 0$,*
(i) $\texttt{cost}(\texttt{BPL}) \le \texttt{OPT} + \frac{d}{\epsilon}$
(ii) $\mathbb{E}[\texttt{cost}(\texttt{FPL})] \le \mathbb{E}[\texttt{cost}(\texttt{BPL})] + \epsilon \cdot U^2 \cdot T$

Then choosing $\epsilon = \frac{\sqrt{d}}{U\sqrt{T}}$ gives Theorem 7.8. 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, it holds that

$$v \cdot M(v) \le v \cdot a \quad \text{for any cost vector } v \text{ and feasible action } a. \tag{7.4}$$

The main argument proceeds as follows:

$$\begin{aligned}
\texttt{cost}(\texttt{BPL}) + v_0 \cdot M(v_0) &= \sum_{t=0}^{T} v_t \cdot M(v_{0:T}) && \textit{(by definition of } \texttt{BPL}\textit{)} \\
&\le v_{0:T} \cdot M(v_{0:T}) && \textit{(see Claim 7.11 below)} \quad (7.5) \\
&\le v_{0:T} \cdot M(v_{1:T}) && \textit{(by (7.4) with } a = M(v_{1:T})\textit{)} \\
&= v_0 \cdot M(v_{1:T}) + \underbrace{v_{1:T} \cdot M(v_{1:T})}_{\texttt{OPT}}.
\end{aligned}$$

Subtracting $v_0 \cdot M(v_0)$ from both sides, we obtain Lemma 7.10(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 (7.5) follows from the following claim, with $i = 0$ and $j = T$.

Claim 7.11. *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:

$$\begin{aligned}
\sum_{t=i}^{j-1} v_t \cdot M(v_{i:t}) &\le v_{i:j-1} \cdot M(v_{i:j-1}) && \textit{(by the inductive hypothesis)} \\
&\le v_{i:j-1} \cdot M(v_{i:j}) && \textit{(by (7.4) with } a = M(v_{i:j})\textit{)}.
\end{aligned}$$

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

$$\mathbb{E}[\underbrace{v_t \cdot M(v_{0:t-1})}_{c_t(a_t)\text{ for FPL}}] \leq \mathbb{E}[\underbrace{v_t \cdot M(v_{0:t})}_{c_t(a_t)\text{ for BPL}}] + \epsilon U^2. \tag{7.6}$$

Summing up over all T rounds gives Lemma 7.10(ii).

It turns out that for proving (7.6) much of the structure in our problem is irrelevant. Specifically, we can denote $f(u) = v_t \cdot M(u)$ and $v = v_{1:t-1}$, and, essentially, prove (7.6) for arbitrary $f()$ and v.

Claim 7.12. *For any vectors $v \in \mathbb{R}^d$ and $v_t \in [0, U/d]^d$, and any function $f : \mathbb{R}^d \to [0, R]$,*

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

In words: changing the input of function f from $v_0 + v$ to $v_0 + v + v_t$ does not substantially change the output, in expectation over v_0. What we actually prove is the following:

Claim 7.13. *Fix $v_t \in [0, U/d]^d$. There exists a random variable $v_0' \in \mathbb{R}^d$ such that (i) v_0' and $v_0 + v_t$ have the same marginal distribution, and (ii) $\Pr[v_0' \neq v_0] \leq \epsilon U$.*

It is easy to see that Claim 7.12 follows from Claim 7.13:

$$\begin{aligned} |\,\mathbb{E}\,[f(v_0 + v) - f(v_0 + v + v_t)]\,| &= \left|\,\mathbb{E}\left[f(v_0 + v) - f(v_0' + v)\right]\,\right| \\ &\leq \Pr[v_0' \neq v_0] \cdot R = \epsilon U R. \end{aligned}$$

It remains to prove Claim 7.13. First, let us prove this claim in one dimension:

Claim 7.14. *let X be a random variable uniformly distributed on the interval $[-\frac{1}{\epsilon}, \frac{1}{\epsilon}]$. We claim that for any $a \in [0, U/d]$ there exists a deterministic function $g(X, a)$ of X and a such that $g(X, a)$ and $X + a$ have the same marginal distribution, and $\Pr[g(X, a) \neq X] \leq \epsilon U/d$.*

Proof. Let us define

$$g(X, a) = \begin{cases} X & \text{if } X \in [v - \frac{1}{\epsilon}, \frac{1}{\epsilon}], \\ a - X & \text{if } X \in [-\frac{1}{\epsilon}, v - \frac{1}{\epsilon}). \end{cases}$$

It is easy to see that $g(X, a)$ is distributed uniformly on $[v - \frac{1}{\epsilon}, v + \frac{1}{\epsilon}]$. This is because $a - X$ is distributed uniformly on $[\frac{1}{\epsilon}, v + \frac{1}{\epsilon}]$ conditional on $X \in [-\frac{1}{\epsilon}, v - \frac{1}{\epsilon})$. Moreover,

$$\Pr[g(X, a) \neq X] \leq \Pr[X \notin [v - \tfrac{1}{\epsilon}, \tfrac{1}{\epsilon}]] = \epsilon v/2 \leq \tfrac{\epsilon U}{2d}. \qquad \square$$

To complete the proof of Claim 7.13, write $v_t = (v_{t,1}, v_{t,2}\,,\ \ldots\ , v_{t,d})$, and define $v_0' \in \mathbb{R}^d$ by setting its j-th coordinate to $Y(v_{0,j}, v_{t,j})$, for each coordinate j. We are done!

Chapter 8

Contextual Bandits *(rev. Sep'18)*

In *contextual bandits*, rewards in each round depend on a *context*, which is observed by the algorithm prior to making a decision. We cover the basics of three prominent versions of contextual bandits: with a Lipschitz assumption, with a linearity assumption, and with a fixed policy class. We also touch upon offline learning from contextual bandit data. Finally, we discuss challenges that arise in large-scale applications, and a system design that address these challenges in practice.

We consider a generalization called *contextual bandits*, defined as follows:

Problem protocol: Contextual bandits

For each round $t \in [T]$:
1. algorithm observes a "context" x_t,
2. algorithm picks an arm a_t,
3. reward $r_t \in [0, 1]$ is realized.

The reward r_t in each round t depends both on the context x_t and the chosen action a_t. We make the IID assumption: reward r_t is drawn independently from some distribution parameterized by the (x_t, a_t) pair, but same for all rounds 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. We assume contexts $x_1, x_2, \ldots$ are chosen by an oblivious adversary.

Several variants of this setting have been studied in the literature. We discuss three prominent variants in this chapter: with a Lipschitz assumption, with a linearity assumption, and with a fixed policy class.

Motivation. The main motivation is that a user with a known "user profile" arrives in each round, and the context is the user profile. The algorithm can personalize the user's experience. Natural application scenarios include choosing which news articles to showcase, which ads to display, which products to recommend, or which webpage layouts to use. Rewards in these applications 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) user profiles. First, contexts can include known features of the environment, such as day of the week, time of the day, season (*e.g.,* Summer, pre-Christmas shopping season), or proximity to a major event (*e.g.,* Olympics, elections). Second, some actions may be unavailable in a given round and/or for a given user, and a context can include the set of feasible actions. Third, actions can come with features of their own, and it may be convenient to include this information into the context, esp. if these features can change over time.

Regret relative to best response. For ease of exposition, we assume a fixed and known time horizon T. The set of actions and the set of all contexts are $\mathcal{A}$ and $\mathcal{X}$, resp.; $K = |\mathcal{A}|$ is the number of actions.

The (total) reward of an algorithm ALG is $\mathtt{REW}(\mathtt{ALG}) = \sum_{t=1}^{T} r_t$, so that the expected reward is

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

A natural benchmark is the best-response policy, $\pi^*(x) = \max_{a\in\mathcal{A}} \mu(a|x)$. Then regret is defined as

$$R(T) = \mathtt{REW}(\pi^*) - \mathtt{REW}(\mathtt{ALG}). \tag{8.1}$$

8.1 Warm-up: small number of contexts

One straightforward approach for contextual bandits is to apply a known bandit algorithm ALG such as UCB1: namely, run a separate copy of this algorithm for each context.

```
Initialization: For each context $x$, create an instance $\mathtt{ALG}_x$ of algorithm ALG
for each round $t$ do
    invoke algorithm $\mathtt{ALG}_x$ with $x = x_t$
    "play" action $a_t$ chosen by $\mathtt{ALG}_x$, return reward $r_t$ to $\mathtt{ALG}_x$.
end
```

Algorithm 8.1: Contextual bandit algorithm for a small number of contexts

Let n_x be the number of rounds in which context x arrives. Regret accumulated in such rounds is $\mathbb{E}[R_x(T)] = O(\sqrt{K n_x \ln T})$. The total regret (from all contexts) is

$$\mathbb{E}[R(T)] = \textstyle\sum_{x\in\mathcal{X}} \mathbb{E}[R_x(T)] = \sum_{x\in\mathcal{X}} O(\sqrt{K n_x \ln T}) \leq O(\sqrt{KT\,|\mathcal{X}|\,\ln T}).$$

Theorem 8.1. *Algorithm 8.1 has regret* $\mathbb{E}[R(T)] = O(\sqrt{KT\,|\mathcal{X}|\,\ln T})$, *provided that the bandit algorithm* ALG *has regret* $\mathbb{E}[R_{\mathtt{ALG}}(T)] = O(\sqrt{KT \log T})$.

Remark 8.2. 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 contextual bandits with a large $|\mathcal{X}|$, we either assume some structure (as in Sections 8.2 and 8.3, or change the objective (as in Section 8.4).

8.2 Lipshitz contextual bandits

Let us consider contextual bandits with Lipschtz-continuity, as a simple end-to-end example of how structure allows to handle contextual bandits with a large number of contexts. We assume that contexts map into the $[0, 1]$ interval (*i.e.,* $\mathcal{X} \subset [0, 1]$) so that the expected rewards are Lipshitz with respect to the contexts:

$$|\mu(a|x) - \mu(a|x')| \leq L \cdot |x - x'| \quad \text{for any arms } a, a' \in \mathcal{A} \text{ and contexts } x, x' \in \mathcal{X}, \tag{8.2}$$

where L is the Lipschitz constant which is known to the algorithm.

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 $\epsilon = 1/(d-1)$, where the integer d is the number of points in S, to be adjusted later in the analysis.

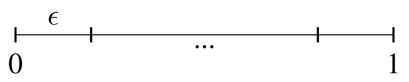


Figure 8.1: Discretization of the context space

We will use the contextual bandit algorithm from Section 8.1, 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:

$$\text{In each round } t\text{, "pre-process" the context } x_t \text{ by replacing it with } f_S(x_t)\text{, and call } \texttt{ALG}_S. \tag{8.3}$$

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.,

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

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 8.1, so it remains to upper-bound the discretization error and adjust the discretization step ϵ.

Claim 8.3. $\mathbb{E}[\texttt{DE}(S)] \leq \epsilon L T$.

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

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

Summing this up over all rounds t, we obtain

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

Thus, regret is

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

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

Theorem 8.4. *Consider the Lispchitz contextual bandits problem with contexts in* $[0,1]$. *The uniform discretization algorithm (8.3) yields regret* $\mathbb{E}[R(T)] = O(T^{2/3}(LK \ln T)^{1/3})$.

An astute reader would notice a similarity with the uniform discretization result in Theorem 4.1. In fact, these two results admit a common generalization in which the Lipschitz condition applies to both contexts and arms, and arbitrary metrics are allowed. Specifically, the Lipschitz condition is now

$$|\mu(a|x) - \mu(a'|x')| \leq D_{\mathcal{X}}(x,x') + D_{\mathcal{A}}(a,a') \quad \text{for any arms } a,a' \text{ and contexts } x,x', \tag{8.4}$$

where $D_{\mathcal{X}}, D_{\mathcal{A}}$ are arbitrary metrics on contexts and arms, respectively, that are known to the algorithm. This generalization is fleshed out in Exercise 8.1.

8.3 Linear contextual bandits: LinUCB algorithm (no 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

$$x = (x_a \in \{0,1\}^d : a \in \mathcal{A}) \tag{8.5}$$

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 (8.5), and the expected rewards are linear:

$$\mu(a|x) = x_a \cdot \theta_a \quad \text{for all arms } a \text{ and contexts } x, \tag{8.6}$$

for some fixed but unknown vector $\theta = (\theta_a \in \mathcal{R}^d : a \in \mathcal{A})$.[1]

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 8.2.

```
for each round $t = 1, 2, \ldots$ 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 8.2: 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.

[1]We allow the θ_a in (8.6) to depend on the arm a. The special case when all θ_a's are the same is also interesting.

8.4 Contextual bandits with a 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 Equation (8.1)). 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.

We assume that contexts arrive as independent samples from some fixed distribution $\mathcal{D}$ over contexts. The expected reward of a given policy π is then well-defined:

$$\mu(\pi) = \mathop{\mathbb{E}}_{x \in \mathcal{D}} \left[\mu(\pi(x)) \mid x \right]. \tag{8.7}$$

This quantity, also known as *policy value*, gives a concrete way to compare policies to one another. The appropriate notion of regret, for an algorithm $\texttt{ALG}$, is relative to the best policy in a given policy class Π:

$$R_\Pi(T) = T \max_{\pi \in \Pi} \mu(\pi) - \texttt{REW}(\texttt{ALG}). \tag{8.8}$$

Note that the definition (8.1) can be seen a special case when Π is the class of all policies. For ease of presentation, we assume $K < \infty$ actions throughout; the set of actions is denoted $[K]$.

Remark 8.5 (Some examples). A policy can be based on a *score predictor*: given a context x, it assigns a numerical score $\nu(a|x)$ to each action a. Such score can represent an estimated expected reward of this action, or some other quality measure. The policy simply picks an action with a highest predicted reward. For a concrete example, if contexts and actions are represented as known feature vectors in $\mathbb{R}^d$, a *linear* score predictor is $\nu(a|x) = aMx$, for some fixed $d \times d$ weight matrix M.

A policy can also be based on a *decision tree*, a flowchart in which each internal node corresponds to a "test" on some attribute(s) of the context (*e.g.,* is the user male of female), branches correspond to the possible outcomes of that test, and each terminal node is associated with a particular action. Execution starts from the "root node" of the flowchart and follows the branches until a terminal node is reached.

Preliminary result. This problem can be solved with algorithm $\texttt{Exp4}$ from Chapter 6, with policies $\pi \in \Pi$ as "experts". Specializing Theorem 6.12 to the setting of contextual bandits, we obtain:

Theorem 8.6. *Consider contextual bandits with policy class* Π. *Algorithm* $\texttt{Exp4}$ *with expert set* Π *yields regret* $\mathbb{E}[R_\Pi(T)] = O(\sqrt{KT \log |\Pi|})$. *However, the running time per round is linear in* $|\Pi|$.

This is a very powerful regret bound: it works for an arbitrary policy class Π, and the logarithmic dependence on $|\Pi|$ makes it tractable 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 $|\mathcal{X}|$. This regret bound is essentially the best possible. Specifically, there is a nearly matching lower bound (similar to (6.1)), which holds for any given triple of parameters $K, T, |\Pi|$:

$$\mathbb{E}[R(T)] \geq \min\left(T,\ \Omega\left(\sqrt{KT \log(|\Pi|)/\log(K)}\right)\right). \tag{8.9}$$

However, the running time of $\texttt{Exp4}$ scales as $|\Pi|$ rather than $\log |\Pi|$, which makes the algorithm prohibitively slow in practice. In what follows, we achieve similar regret rates, but with faster algorithms.

Connection to a classification problem. We make a connection to a well-studied classification problem in "traditional" machine learning. This connection leads to faster algorithms, and is probably the main motivation for the setting of contextual bandits with policy sets.

To build up the intuition, let us consider the full-feedback version of contextual bandits:

Problem protocol: Contextual bandits with full feedback

For each round $t = 1, 2, \ldots$:
1. algorithm observes a "context" x_t,
2. algorithm picks an arm a_t,
3. rewards $\tilde{r}_t(a) \geq 0$ are observed for all arms $a \in \mathcal{A}$.

In fact, let us make the problem even easier. Suppose we already have N data points of the form $(x_t; \tilde{r}_t(a) : a \in \mathcal{A})$. What is the "best-in-hindsight" policy for this dataset? More precisely, what is a policy $\pi \in \Pi$ with a largest *realized policy value*

$$\tilde{r}(\pi) = \frac{1}{N} \sum_{t=1}^{N} \tilde{r}_t(\pi(x_i)). \tag{8.10}$$

This happens to be a well-studied problem called "cost-sensitive multi-class classification":

Problem: Cost-sensitive multi-class classification for policy class Π

Given: data points $(x_t; \tilde{r}_t(a) : a \in \mathcal{A})$, $t \in [N]$.
Find: policy $\pi \in \Pi$ with a largest realized policy value (8.10).

In the terminology of classification problems, each context x_t is an "example", and the arms correspond to different possible "labels" for this example. Each label has an associated reward/cost. We obtain a standard binary classification problem if for each data point t, there is one "correct label" with reward 1, and rewards for all other labels are 0. The practical motivation for finding the "best-in-hindsight" policy is that it is likely to be a good policy for future context arrivals. In particular, such policy is near-optimal under the IID assumption, see Exercise 8.2 for a precise formulation.

An algorithm for this problem will henceforth be called a *classification oracle* for policy class Π. While the exact optimization problem is NP-hard for many natural policy classes, practically efficient algorithms exist for several important policy classes such as linear classifiers, decision trees and neural nets.

A very productive approach for designing contextual bandit algorithms uses a classification oracle as a subroutine. 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, algorithms can use any available classification oracle; then the relevant policy class Π is simply the policy class that the oracle optimizes over.

A simple oracle-based algorithm. Consider a simple explore-then-exploit algorithm that builds on a classification oracle. First, we explore uniformly for the first N rounds, where N is a parameter. Each round t of exploration gives a data point $(x_t, \tilde{r}_t(a) \in \mathcal{A})$ for the classification oracle, where the "fake rewards" $\tilde{r}_t(\cdot)$

are given by inverse propensity scoring:

$$\tilde{r}_t(a) = \begin{cases} r_t K & \text{if } a = a_t \\ 0, & \text{otherwise.} \end{cases} \tag{8.11}$$

We call the classification oracle and use the returned policy in the remaining rounds; see Algorithm 8.3.

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 $(x_t, \tilde{r}_t(a) \in \mathcal{A})$, $t \in [N]$ as per Equation (8.11).
3. Exploitation: in each subsequent round, use the policy π_0 returned by the oracle.

Algorithm 8.3: Explore-then-exploit with a classification oracle

Remark 8.7. Algorithm 8.3 is modular in two ways: it can take an arbitrary classification oracle, and it can use any other unbiased estimator instead of Equation (8.11). In particular, the proof below only uses the fact that Equation (8.11) is an unbiased estimator with $\tilde{r}_t(a) \leq K$.

For a simple analysis, assume that the rewards are in $[0,1]$ and that the oracle is *exact*, in the sense that it returns a policy $\pi \in \Pi$ that exactly maximizes $\mu(\pi)$.

Theorem 8.8. *Let $\mathcal{O}$ be an exact classification oracle for some policy class Π. Algorithm 8.3 parameterized with oracle $\mathcal{O}$ and $N = T^{2/3}(K \log(|\Pi| T))^{\frac{1}{3}}$ has regret*

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

This regret bound has two key features: logarithmic dependence on $|\Pi|$ and $\tilde{O}(T^{2/3})$ dependence on T.

Proof. Let us consider an arbitrary N for now. For a given policy π, we estimate its expected reward $\mu(\pi)$ using the realized policy value from (8.10), where the action costs $\tilde{r}_t(\cdot)$ are from (8.11). Let us prove that $\tilde{r}(\pi)$ is an unbiased estimate for $\mu(\pi)$:

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

which implies $\mathbb{E}[\tilde{r}(\pi)] = \mu(\pi)$, as claimed. Now, let us use this estimate to set up a "clean event":

$$\{ \mid \tilde{r}(\pi) - \mu(\pi) \mid \leq \texttt{conf}(N) \text{ for all policies } \pi \in \Pi \},$$

where the confidence term is $\texttt{conf}(N) = O(\sqrt{\frac{K \log(T|\Pi|)}{N}})$. 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 the "fake rewards" $\tilde{r}_t(\cdot)$ could be as large as K. 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, $\tilde{r}(\pi_0)$ is maximal among all policies $\pi \in \Pi$. In particular, $\tilde{r}(\pi_0) \geq \tilde{r}(\pi^*)$. 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 $\texttt{conf}$ 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 $\mathbb{E}[R_\Pi(T)] = O(N)$. $\square$

Remark 8.9. If the oracle 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.

Oracle-efficient algorithm with optimal regret. A near-optimal regret bound can in fact be achieved with an algorithm that makes only a small number of oracle calls. Specifically, one can achieve regret $O(\sqrt{KT\log(T|\Pi|)})$ with only $\tilde{O}(\sqrt{KT/\log|\Pi|})$ oracle calls across all T rounds. This sophisticated result can be found in (Agarwal et al., 2014). Its exposition is beyond the scope of this book.

8.5 Policy evaluation and training

Data collected by a contextual bandit algorithm can be analyzed "offline", separately from running the algorithm. Typical tasks are estimating the value of a given policy (*policy evaluation*), and learning a policy that performs best on the dataset (*policy training*). While these tasks are formulated in the terminology from Section 8.4, they are meaningful for all settings discussed in this chapter.

Let us make things more precise. Assume that a contextual bandit algorithm has been running for T rounds according to the following extended protocol:

Problem protocol: Contextual bandit data collection

For each round $t \in [N]$:
1. algorithm observes a "context" x_t,
2. algorithm picks a sampling distribution p_t over arms,
3. arm a_t is drawn independently from distribution p_t,
4. rewards $\tilde{r}_t(a)$ are realized for all arms $a \in \mathcal{A}$,
4. reward $r_t = \tilde{r}_t(a_t) \in [0, 1]$ is recorded.

Thus, we have N data points of the form (x_t, p_t, a_t, r_t). The sampling probabilities p_t are essential to form the IPS estimates, as explained below. It is particularly important to record the sampling probability for the chosen action, $p_t(a_t)$. Policy evaluation and training are defined as follows:

Problem: Policy evaluation and training

Input: data points (x_t, p_t, a_t, r_t), $t \in [N]$.
Policy evaluation: estimate policy value $\mu(\pi)$ for a given policy π.
Policy training: find policy $\pi \in \Pi$ that maximizes $\mu(\pi)$ over a given policy class Π.

Inverse propensity scoring. Policy evaluation can be addressed via inverse propensity scoring (IPS), like in Algorithm 8.3. This approach is simple and does not rely on a particular model of the rewards such as linearity or Lipschitzness. We estimate the value of each policy π as follows:

$$\texttt{IPS}(\pi) = \sum_{t \in [N]:\; \pi(x_t) = a_t} \frac{r_t}{p_t(a_t)}. \tag{8.12}$$

Just as in the proof of Theorem 8.8, one can show that $\texttt{IPS}(\pi)$ is equal to the policy value $\mu(\pi)$ in expectation, and close to it with high probability, as long as the probabilities are large enough.

Lemma 8.10. $\mathbb{E}[\texttt{IPS}(\pi)] = \mu(\pi)$ *for each policy* π. *Moreover, for each* $\delta > 0$, *with probability at least* $1 - \delta$

$$|\texttt{IPS}(\pi) - \mu(\pi)| \leq O\left(\sqrt{\tfrac{1}{p_0}\;\log(\tfrac{1}{\delta})/N}\right), \quad \textit{where } p_0 = \min_{t,a} p_t(a). \tag{8.13}$$

Remark 8.11. How many data points do we need to evaluate M policies simultaneously? To make this question more precise, suppose we have some fixed parameters $\epsilon, \delta > 0$ in mind, and want to ensure that

$$\Pr\left[\; |\texttt{IPS}(\pi) - \mu(\pi)| \leq \epsilon \quad \text{for each policy } \pi \;\right] > 1 - \delta.$$

How large should N be, as a function of M and the parameters? Taking a union bound over (8.13), we see that it suffices to take

$$N \sim \frac{\sqrt{\log(M/\delta)}}{p_0 \cdot \epsilon^2}.$$

The logarithmic dependence on M is due to the fact that each data point t can be reused to evaluate many policies, namely all policies π with $\pi(x_t) = a_t$.

We can similarly compare $\texttt{IPS}(\pi)$ with the *realized* policy value $\tilde{r}(\pi)$, as defined in (8.10). This comparison does not rely on IID rewards: it applies whenever rewards are chosen by an oblivious adversary.

Lemma 8.12. *Assume rewards* $\tilde{r}_t(\cdot)$ *are chosen by a deterministic oblivious adversary. Then we have* $\mathbb{E}[\texttt{IPS}(\pi)] = \tilde{r}(\pi)$ *for each policy* π. *Moreover, for each* $\delta > 0$, *with probability at least* $1 - \delta$ *we have:*

$$|\texttt{IPS}(\pi) - \tilde{r}(\pi)| \leq O\left(\sqrt{\tfrac{1}{p_0}\;\log(\tfrac{1}{\delta})/N}\right), \quad \textit{where } p_0 = \min_{t,a} p_t(a). \tag{8.14}$$

This lemma implies Lemma 8.10. It easily follows from a concentration inequality, see Exercise 8.3.

Policy training can be implemented similarly: by maximizing $\texttt{IPS}(\pi)$ over a given policy class Π. A maximizer π_0 can be found by a single call to the classification oracle for Π: indeed, call the oracle with "fake rewards" defined by estimates: $\rho_t(a) = \mathbf{1}_{\{a = a_t\}} \frac{r_t}{p_t(a_t)}$ for each arm a and each round t. The performance guarantee follows from Lemma 8.12:

Corollary 8.13. *Consider the setting in Lemma 8.12. Fix policy class* Π *and let* $\pi_0 \in \operatorname{argmax}_{\pi\in\Pi} \texttt{IPS}(\pi)$. *Then for each* $\delta > 0$*, with probability at least* $\delta > 0$ *we have*

$$\max_{\pi\in\Pi} \tilde{r}(\pi) - \tilde{r}(\pi_0) \leq O\left(\sqrt{\tfrac{1}{p_0}\ \log(\tfrac{|\Pi|}{\delta})/N}\right). \tag{8.15}$$

Model-dependent approaches. One could estimate the mean rewards $\mu(a|x)$ directly, *e.g.,* with linear regression, and then use the resulting estimates $\hat{\mu}(a|x)$ to estimate policy values:

$$\hat{\mu}(\pi) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\hat{\mu}(\pi(x) \mid x)].$$

Such approaches are typically motivated by some model of rewards, *e.g.,* linear regression is motivated by the linear model. Their validity depends on how well the data satisfies the assumptions in the model.

For a concrete example, linear regression based on the model in Section 8.3 constructs estimates $\hat{\theta}_a$ for latent vector θ_a, for each arm a. Then rewards are estimated as $\hat{\mu}(a|x) = x \cdot \theta_a$.

Model-dependent reward estimates can naturally be used for policy optimization:

$$\pi(x) = \operatorname*{argmax}_{a\in\mathcal{A}} \hat{\mu}(a|x).$$

Such policy can be good even if the underlying model is not. No matter how a policy is derived, it can be evaluated in a model-independent way using the IPS methodology described above.

8.6 Contextual bandits in practice: challenges and a system design

Implementing contextual bandit algorithms for large-scale applications runs into a number of engineering challenges and necessitates a design of a *system* along with the algorithms. We present a system for contextual bandits, called the Decision Service, building on the machinery from the previous two sections.

The key insight is the distinction between the *front-end* of the system, which directly interacts with users and needs to be extremely fast, and the *back-end*, which can do more powerful data processing at slower time scales. *Policy execution*, computing the action given the context, must happen in the front-end, along with data collection (*logging*). Policy evaluation and training usually happen in the back-end. When a better policy is trained, it can be deployed into the front-end. This insight leads to a particular methodology, organized as a loop in Figure 8.2. Let us discuss the four components of this loop in more detail.

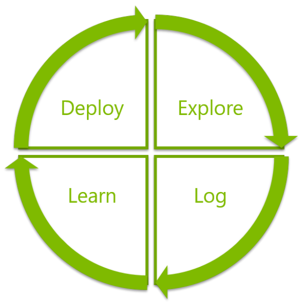


Figure 8.2: The learning loop.

Exploration Actions are chosen by the *exploration policy*: a fixed policy that runs in the front-end and combines exploration and exploitation. Conceptually, it takes one or more *default policies* as subroutines, and adds some exploration on top. A default policy is usually known to be fairly good; it could be a policy already deployed in the system and/or trained by a machine learning algorithm. One basic exploration policy is *ϵ-greedy*: choose an action uniformly at random with probability ϵ ("exploration branch"), and execute the default policy with the remaining probability ("exploitation branch"). If several default policies are given, the exploitation branch chooses uniformly at random among them; this provides an additional layer of exploration, as the default policies may disagree with one another.

If a default policy is based on a score predictor $\nu(a|x)$, as in Example 8.5, an exploration policy can give preference to actions with higher scores. One such exploration policy, known as *SoftMax*, assigns to each action a the probability proportional to $e^{\tau \cdot \nu(a|x)}$, where τ is the exploration parameter. Note that $\tau = 0$ corresponds to the uniform action selection, and increasing the τ favors actions with higher scores. Generically, exploration policy is characterized by its type (*e.g.,* ϵ-greedy or SoftMax), exploration parameters (such as the ϵ or the τ), and the default policy/policies.

Logging Logging runs in the front-end, and records the "data points" defined in Section 8.5. Each logged data point includes (x_t, a_t, r_t) – the context, the chosen arm, and the reward – and also $p_t(a_t)$, the probability of the chosen action. Additional information may be included to help with debugging and/or machine learning, *e.g.,* time stamp, current exploration policy, or full sampling distribution p_t.

In a typical high-throughput application such as a website, the reward (*e.g.,* a click) is observed long after the action is chosen – much longer than a front-end server can afford to "remember" a given user. Accordingly, the reward is logged separately. The *decision tuple*, comprising the context, the chosen action, and sampling probabilities, is recorded by the front-end when the action is chosen. The reward is logged after it is observed, probably via a very different mechanism, and joined with the decision tuple in the back-end. To enable this join, the front-end server generates a unique "tuple ID" which is included in the decision tuple and passed along to be logged with the reward as the *outcome tuple*.

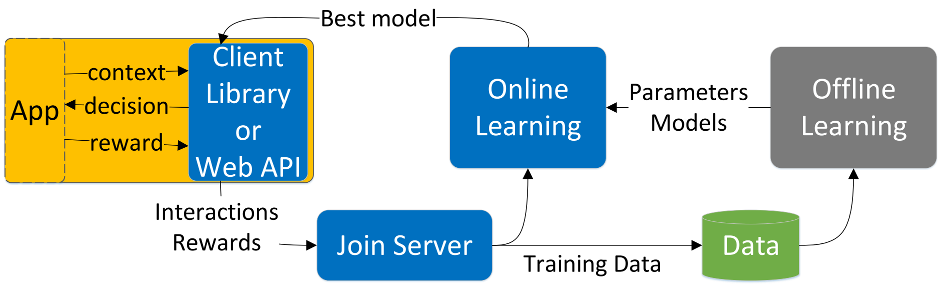
Logging often goes wrong in practice. The sampling probabilities p_t may be recorded incorrectly, or accidentally included as features. Features may be stored as references to a database which is updated over time (so the feature values are no longer the ones observed by the exploration policy). When optimizing an intermediate step in a complex system, the action chosen initially might be overridden by business logic, and the recorded action might incorrectly be this final action rather than the initially chosen action. Finally, rewards may be lost or incorrectly joined to the decision tuples.

Learning Policy training, discussed in Section 8.5, happens in the back-end, on the logged data. The main goal is to learn a better "default policy", and perhaps also better exploration parameters. The policy training algorithm should be *online*, allowing fast updates when new data points are received, so as to enable rapid iterations of the learning loop in Figure 8.2.

Offline learning uses logged data to simulate experiments on live users. Policy evaluation, for example, simulates deploying a given policy. One can also experiment with alternative exploration policies or parameterizations thereof. Further, one can try out other algorithms for policy training, such as those that need more computation, use different hyper-parameters, are based on estimators other than IPS, or lead to different policy classes. The algorithms being tested do not need to be approved for a live deployment (which may be a big hurdle), or implemented at a sufficient performance and reliability level for the said deployment (which tends to be very expensive).

Policy deployment New default policies and/or exploration parameters are deployed into the exploration policy, thereby completing the learning loop in Figure 8.2. As a safeguard, one can use human oversight and/or policy evaluation to compare the new policy with others before deploying it.[2] The deployment process should be automatic and frequent. The frequency of deployments depends on the delays in data collection and policy training, and on the need of human oversight. Also, some applications require a new default policy to improve over the old one by a statistically significant margin, which may require waiting for more data points.

The methodology described above leads to the following system design:



The front-end interacts with the application via a provided software library ("Client Library") or via an Internet-based protocol ("Web API"). The Client Library implements exploration and logging, and the Web API is an alternative interface to the said library. Decision tuples and outcome tuples are joined by the Join Server, and then fed into a policy training algorithm (the "Online Learning" box). Offline Learning is implemented as a separate loop which can operate at much slower frequencies.

Modularity of the design is essential, so as to integrate easily with an application's existing infrastructure. In particular, the components have well-defined interfaces, and admit multiple consistent implementations. This avoids costly re-implementation of existing functionality, and allows the system to improve seamlessly as better implementations become available.

Essential issues and extensions

Let discuss several additional issues and extensions that tend to be essential in applications. As a running example, consider optimization of a simple news website called `SimpleNews`. While this example is based on a successful product deployment, we ignore some real-life complications for the sake of clarity. Thus, `SimpleNews` displays exactly one headline to each user. When a user arrives, some information is available pertaining to the interaction with this particular user, *e.g.,* age, gender, geolocation, time of day, and possibly much more; such information is summarily called the *context*. The website chooses a news topic (*e.g.,* politics, sports, tech, etc.), possibly depending on the context, and the top headline for the chosen topic is displayed. The website's goal is to maximize the number of clicks on the headlines that it displays. For this example, we are only concerned with the choice of news topic: that is, we want to pick a news topic whose top headline is most likely to be clicked on for a given context.

Fragmented outcomes. A good practice is to log the entire observable outcome, not just the reward. For example, the outcome of choosing a news topic in `SimpleNews` includes the *dwell time*: the amount of time

[2]A standard consideration in machine learning applies: a policy should not be evaluated on the training data. Instead, a separate dataset should be set aside for policy evaluation.

the user spent reading a suggested article (if any). Multiple outcome fragments may arrive at different times. For example, the dwell time `SimpleNews` is recorded long after the initial click on the article. Thus, multiple outcome tuples may be logged by the front-end. If the reward depends on multiple outcome fragments, it may be computed by the Join Server after the outcome tuples are joined, rather than in the front-end.

Reward metrics. There may be several reasonable ways to define rewards, especially with fragmented outcomes. For example, in `SimpleNews` the reward can include a bonus that depends on the dwell time, and there may be several reasonable choices for defining such a bonus. Further, the reward may depend on the context, *e.g.,* in `SimpleNews` some clicks may be more important than others, depending on the user's demographics. Thus, a reward metric used by the application is but a convenient proxy for the long-term objective such as cumulative revenue or long-term customer satisfaction. The reward metric may change over time when priorities change or when a better proxy for the long-term objective is found. Further, it may be desirable to consider several reward metrics at once, *e.g.,* for safeguards. Offline learning can be used to investigate the effects of switching to another reward metric in policy training.

Non-stationarity. While we've been positing a stationarity environment so far, applications exhibit only periods of near-stationarity in practice. To cope with a changing environment, we use a continuous loop in which the policies are re-trained and re-deployed quickly as new data becomes available. Therefore, the infrastructure and the policy training algorithm should process new data points sufficiently fast. Policy training should de-emphasize older data points over time. Enough data is needed within a period of near-stationarity (so more data is needed to adapt to faster rate of change).

Non-stationarity is partially mitigated if some of it is captured by the context. For example, users of `SimpleNews` may become more interested in sports during major sports events such as Olympics. If the presence of such events is included as features in the context, the response to a particular news topic given a particular context becomes more consistent across time.

In a non-stationary environment, the goal of policy evaluation is no longer to estimate the expected reward (simply because this quantity changes over time) or predict rewards in the future. Instead, the goal is *counterfactual*: estimate the policy's performance if it were used when the exploration data was collected. This is a mathematically precise goal that is achievable (say) by the IPS estimator regardless of how the environment changes in the meantime. When algorithms for exploration and/or policy training are evaluated on the exploration data, the goal is counterfactual in a similar sense. This is very useful for comparing these algorithms with alternative approaches. One hopes that such comparison on the exploration data would be predictive of a similar comparison performed via a live A/B test.

Feature selection. Selecting which features to include in the context is a fundamental issue in machine learning. Adding more features tends to lead to better rewards in the long run, but may slow down the learning. The latter could be particular damaging if the environment changes over time and fast learning is essential. The same features can be represented in different ways, *e.g.,* a feature with multiple possible values can be represented either as one value or as a bit vector (*1-hot encoding*) with one bit for each possible value. Feature representation is essential, as general-purpose policy training algorithms tend to be oblivious to what features *mean*. A good feature representation may depend on a particular policy training algorithm. Offline learning can help investigate the effects of adding/removing features or changing their representation. For this purpose, additional observable features (not included in the context) can be logged in the decision tuple and passed along to offline learning.

Ease of adoption. Developers might give up on a new system, unless it is easy to adopt. Implementation should avoid or mitigate dependencies on particular programming languages or external libraries. The trial experience should be seamless, *e.g.,* sensible defaults should be provided for all components.

Road to deployment

Consider a particular application such as `SimpleNews`, call it `APP`. Deploying contextual bandits for this application should follow a process, even when the contextual bandit system such as the Decision Service is available. One should do some prep-work: frame the `APP` as a contextual bandit problem, verify that enough data would be available, and have a realistic game plan for integrating the system with the existing infrastructure. Next step is a pilot deployment on a small fraction of traffic. The goal here is to validate the system for `APP` and debug problems; deep integration and various optimizations can come later. Finally, the system is integrated into `APP`, and deployed on a larger fraction of traffic.

Framing the problem. Interactions between `APP` and its users should be interpreted as a sequence of small interactions with individual users, possibly overlapping in time. Each small interaction should follow a simple template: observe a *context*, make a *decision*, choosing from the available alternatives, and observe the *outcome* of this decision. The meaning of contexts, decisions and outcomes should be consistent throughout. The *context*, typically represented as a vector of features, should encompass the properties of the current user and/or task to be accomplished, and must be known to `APP`. The *decision* must be controlled by `APP`. The set of feasible actions should be known: either fixed in advance or specified by the context. It is often useful to describe actions with features of their own, a.k.a. *action-specific features*. The *outcome* consists of one or several events, all of which must be observable by `APP` not too long after the action is chosen. The outcome (perhaps jointly with the context) should define a *reward*: the short-term objective to be optimized. Thus, one should be able to fill in the table below:

	`SimpleNews`	Your `APP`
Context	(gender, location, time-of-day)	
Decision	a news topic to display	
Feasible actions	{politics, sports, tech, arts}	
Action-specific features	none	
Outcome	click or no click within 20 seconds	
Reward	1 if clicked, 0 otherwise	

Table 8.1: Framing the problem (and using `SimpleNews` as an example)

As discussed above, feature selection and reward definition can be challenging. Defining *decisions* and *outcomes* may be non-trivial, too. For example, `SimpleNews` can be modified so that actions correspond to news articles, rather than news topics, and each *decision* consists of choosing a slate of news articles.

Scale and feasibility. To estimate the scale and feasibility of the learning problem in `APP`, one needs to estimate a few parameters: the number of features (`#features`), the number of feasible actions (`#actions`), a typical delay between making a decision and observing the corresponding outcome, and *the data rate*: the number of experimental units per a time unit. If the outcome includes a rare event whose frequency is crucial to estimate — *e.g.,* clicks are typically rare compared to non-clicks, — then we also need a rough estimate of this frequency. Finally, we need the *stationarity timescale*: the time interval during which the environment does not change too much, namely the distribution of arriving contexts (where a context includes the set of feasible actions and, if applicable, their features), and the expected rewards for each context-action pair. In the `SimpleNews` example, this corresponds to the distribution of arriving user profiles and the click probability for the top headline (for a given news topic, when presented to a typical user with a given user profile). To summarize, one should be able to fill in the following table:

	`SimpleNews`	Your APP
`#features`	3	
`#actions`	4 news topics	
Typical delay	5 sec	
Data rate	100 users/sec	
Rare event frequency	typical click prob. 2-5%	
Stationarity timescale	one week	

Table 8.2: The scalability parameters (using `SimpleNews` as an example)

For policy training algorithms based on linear classifiers, a good rule-of-thumb is that the stationarity timescale should be much larger than the typical delay, and moreover we should have

$$\texttt{StatInterval} \times \texttt{DataRate} \times \texttt{RareEventFreq} \gg \texttt{\#actions} \times \texttt{\#features} \tag{8.16}$$

The left-hand side is the number of rare events in the timescale, and the right-hand side characterizes the complexity of the learning problem.

8.7 Bibliographic remarks and further directions

[TODO: Add deeper discussion of linear CB and CB with policy sets. Add some discussion of offline policy learning and other versions of CB.]

Lipschitz contextual bandits. Contextual bandits with a Lipschitz condition on contexts have been introduced in Hazan and Megiddo (2007), along with a solution via uniform discretization of contexts. The extension to the more general Lipschitz condition (8.4) has been observed in Lu et al. (2010). The regret bound in Exercise 8.1 is optimal in the worst case (Lu et al., 2010; Slivkins, 2014).

Adaptive discretization improves over uniform discretization, much like it does in Chapter 4. The key is to discretize the context-arms pairs, rather than contexts and arms separately. This approach is implemented in Slivkins (2014), achieving regret bounds that are optimal in the worst case, and improve for "nice" problem instances. For precisely, there is a contextual version of the "raw" regret bound (4.11) in terms of the covering numbers, and an analog of Theorem 4.18 in terms of a suitable version of the zooming dimension. Both regret bounds are essentially the best possible. This approach extends to an even more general Lipschitz condition when the right-hand side of (8.2) is an arbitrary metric on context-arm pairs.

This machinery can handle some adversarial bandit problems. Consider the special case of Lipschitz contextual bandits when the context x_t is simply the time t. The Lipschitz condition can be written as

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

where $\mathcal{D}_a$ is a metric on $[T]$ that is known to the algorithm, and possibly parameterized by the arm a. This condition describes a bandit problem with randomized adversarial rewards such that the expected rewards can only change slowly. The paradigmatic special cases are $\mathcal{D}_a(t, t') = \sigma_a \cdot |t - t'|$, bounded change in each round, and $\mathcal{D}_a(t, t') = \sigma_a \cdot \sqrt{|t - t'|}$. The latter case subsumes a scenario when mean rewards follow a random walk. More precisely, the mean reward of each arm a evolves as a random walk with step $\pm\sigma_a$ on the $[0, 1]$ interval with reflecting boundaries. In Slivkins (2014), these problems are solved using a general algorithm for Lipschitz contextual bandits, achieving near-optimal "dynamic regret" (see Section 6.6).

Rakhlin et al. (2015); Cesa-Bianchi et al. (2017) tackle a version of Lipschitz contextual bandits in which the comparison benchmark is the best *Lipschitz policy*: a mapping π from contexts to actions which satisfies $D_{\mathcal{A}}(\pi(x), \pi(x')) \leq D_{\mathcal{X}}(x, x')$ for any two contexts x, x', where $D_{\mathcal{A}}$ and $D_{\mathcal{X}}$ are the metrics from (8.4). Several feedback models are considered, including bandit feedback and full feedback.

Linear contextual bandits. Algorithm `LinUCB` has been introduced in Li et al. (2010), and analyzed in (Chu et al., 2011; Abbasi-Yadkori et al., 2011).

Contextual bandits with policy sets. Theorem 8.6 is from Auer et al. (2002b), and the complimentary lower bound (8.9) is due to Agarwal et al. (2012). The oracle-based approach to contextual bandits was pioneered in Langford and Zhang (2007), with an "epsilon-greedy"-style algorithm and regret bound similar to those in Theorem 8.8. Dudík et al. (2011) obtained a near-optimal regret bound, $O(\sqrt{KT\log(T|\Pi|)})$, via an algorithm that is computationally efficient "in theory". This algorithm makes $\text{poly}(T, K, \log|\Pi|)$ oracle calls and relies on the ellipsoid algorithm. Finally, a break-through result of (Agarwal et al., 2014) achieved the same regret bound via a "truly" computationally efficient algorithm which makes only $\tilde{O}(\sqrt{KT/\log|\Pi|})$ oracle calls across all T rounds. Another recent break-through (Syrgkanis et al., 2016a; Rakhlin and Sridharan, 2016; Syrgkanis et al., 2016b) extends contextual bandits 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.

It is worth noting that the results on oracle-based contextual bandit algorithms (except for the ϵ-greedy-based approach) do not immediately extend to approximate classification oracles. Moreover, the oracle's performance in practice does not necessarily carry over to its performance inside a contextual bandit algorithm, as the latter calls the oracle on carefully constructed artificial problem instances.

Practical aspects. The material in Section 8.6 is adapted from (Agarwal et al., 2016, 2017b). The Decision Service, a system for large-scale applications of contextual bandits, is available as a Cognitive Service on Microsoft Azure, at `http://aka.ms/mwt`.

8.8 Exercises and Hints

Exercise 8.1 (Lipschitz contextual bandits). Consider the Lipschitz condition in (8.4). Design an algorithm with regret bound $\tilde{O}(T^{(d+1)/(d+2)})$, where d is the covering dimension of $\mathcal{X} \times \mathcal{A}$.

Hint: Extend the uniform discretization approach, using the notion of ϵ-mesh from Definition 4.5. Fix Choose an ϵ-mesh $S_{\mathcal{X}}$ for $(\mathcal{X}, \mathcal{D}_{\mathcal{X}})$ and an ϵ-mesh $S_{\mathcal{A}}$ for $(\mathcal{A}, \mathcal{D}_{\mathcal{A}})$, for some $\epsilon > 0$ to be chosen in the analysis. Fix an optimal bandit algorithm `ALG` such as `UCB1`, with $S_{\mathcal{A}}$ as a set of arms. Run a separate copy $\texttt{ALG}_x$ of this algorithm for each context $x \in S_{\mathcal{X}}$.

Exercise 8.2 (Empirical policy value). Prove that realized policy value $\tilde{r}(\pi)$, as defined in (8.10), is close to the policy value $\mu(\pi)$. Specifically, observe that $\mathbb{E}[\tilde{r}(\pi)] = \mu(\pi)$. Next, fix $\delta > 0$ and prove that

$$|\mu(\pi) - \tilde{r}(\pi)| \leq \texttt{conf}(N) \quad \text{with probability at least } 1 - \delta/|\Pi|,$$

where the confidence term is $\texttt{conf}(N) = O\left(\sqrt{\frac{\log(|\Pi|/\delta)}{N}}\right)$. Letting π_0 be the polict with a largest realized policy value, it follows that

$$\max_{\pi \in \Pi} \mu(\pi) - \mu(\pi_0) \leq \texttt{conf}(N) \quad \text{with probability at least } 1 - \delta$$

Exercise 8.3 (Policy evaluation). Use Bernstein Inequality to prove Lemma 8.12. In fact, prove a stronger statement: for each policy π and each $\delta > 0$, with probability at least $1 - \delta$ we have:

$$|\texttt{IPS}(\pi) - \tilde{r}(\pi)| \leq O\left(\sqrt{V\ \log(\tfrac{1}{\delta})/N}\right), \quad \text{where } V = \max_t \sum_{a \in \mathcal{A}} \tfrac{1}{p_t(a)}. \tag{8.17}$$

Chapter 9

Bandits and Zero-Sum Games *(rev. Jan'18)*

This chapter explores connections between bandit algorithms and game theory. We consider a bandit algorithm playing a repeated zero-sum game against an adversary (which could, for example, be another bandit algorithm). We only assume that the algorithm satisfies a sublinear regret bound against an adaptive adversary. We are interested in convergence to an equilibrium: whether, in which sense, and how fast this happens. We present a sequence of results in this direction, focusing on best-response and regret-minimizing adversaries. Along the way, we give a self-contained proof of Von-Noemann's "Minimax Theorem" via bandits. We also present a simple result for general (not zero-sum) games.

Throughout this chapter, we consider the following setup. A bandit algorithm ALG plays a repeated game against another algorithm, called an *adversary* and denoted ADV. The game is characterized by a matrix M, and proceeds over T rounds. In each round t of the game, ALG chooses row i_t of M, and ADV chooses column j_t of M. They make their choices simultaneously, *i.e.,* without observing each other. The corresponding entry $M(i_t, j_t)$ specifies the cost for ALG. After the round, ALG observes the cost $M(i_t, j_t)$ and possibly some auxiliary feedback. Thus, the problem protocol is as follows:

Problem protocol: Repeated game between an algorithm and an adversary

In each round $t \in 1, 2, 3, \ldots, T$:

1. Simultaneously, ALG chooses a row i_t of M, and ADV chooses a column j_t of M.
2. ALG incurs cost $M(i_t, j_t)$, and observes feedback $\mathcal{F}_t = \mathcal{F}(t, i_t, j_t, M)$.

We are mainly interested in two feedback models: $\mathcal{F}_t = M(i_t, j_t)$, *i.e.,* bandit feedback, and $\mathcal{F} = (M(i, j_t) : \text{all rows } i)$, *i.e.,* full feedback (from ALG's perspective). However, all results in this chapter hold for an arbitrary feedback model $\mathcal{F}_t = \mathcal{F}(t, i_t, j_t, M)$, where $\mathcal{F}$ is a fixed *feedback function*.

ALG's objective is to minimize its total cost, $\texttt{cost}(\texttt{ALG}) = \sum_{t=1}^{T} M(i_t, j_t)$.

Unless noted otherwise, we consider zero-sum games, as the corresponding theory is well-developed and has rich applications. Formally, we posit that ALG's cost in each round t is also ADV's reward. In the standard game-theoretic terminology, each round is a *zero-sum game* between ALG and ADV, with *game matrix* M.[1]

[1]Equivalently, ALG and ADV incur costs $M(i_t, j_t)$ and $-M(i_t, j_t)$, respectively. Hence the term 'zero-sum game'.

Regret. The only property of ALG that we will rely on is its regret. Specifically, we consider a bandit problem with the same feedback model $\mathcal{F}$, and for this bandit problem we consider "hindsight regret" $R(T)$ against an adaptive adversary, relative to the "best observed arm", as defined in Chapter 5.1. All results in this chapter are meaningful even if we only control *expected* regret $\mathbb{E}[R(T)]$, more specifically if $\mathbb{E}[R(t)] = o(t)$. Recall that this property is satisfied by algorithms Hedge and Exp3 for full feedback and bandit feedback, respectively (see Chapter 5.3 and Chapter 6).

Let us recap the definition of $R(T)$. Using the terminology from Chapter 5.1, the algorithm's cost for choosing row i in round t is $c_t(i) = M(i, j_t)$. Let $\texttt{cost}(i) = \sum_{t=1}^{T} c_t(i)$ be the total cost for always choosing row i, under the observed costs $c_t(\cdot)$. Then

$$\texttt{cost}^* := \min_{\text{rows } i} \texttt{cost}(i)$$

is the cost of the "best observed arm". Thus, as per Equation (5.1),

$$R(T) = \texttt{cost}(\texttt{ALG}) - \texttt{cost}^*. \tag{9.1}$$

This is what we will mean by *regret* throughout this chapter.

Motivation. At face value, the setting in this chapter is about a repeated game between agents that use regret-minimizing algorithms. Consider algorithms for "actual" games such as chess, go, or poker. Such algorithm can have a "configuration" that can be tuned over time by a regret-minimizing "meta-algorithm" (which interprets configurations as actions, and wins/losses as payoffs). Further, agents in online commerce, such as advertisers in an ad auction and sellers in a marketplace, engage in repeated interactions such that each interaction proceeds as a game between agents according to some fixed rules, depending on bids, prices, or other signals submitted by the agents. An agent may adjust its behavior in this environment using a regret-minimizing algorithm. (However, the resulting repeated game is not zero-sum.)

Our setting, and particularly Theorem 9.4 in which both ALG and ADV minimize regret, is significant in several other ways. First, it leads to an important prediction about human behavior: namely, humans would approximately arrive at an equilibrium if they behave so as to minimize regret (which is a plausible behavioural model). Second, it provides a way to compute an approximate Nash equilibrium for a given game matrix. Third, and perhaps most importantly, the repeated game serves as a subroutine for a variety of algorithmic problems, see Section 9.5 for specific pointers. In particular, such subroutine is crucial for a bandit problem discussed in Chapter 10.

9.1 Basics: guaranteed minimax value

Game theory basics. Imagine that the game consists of a single round, *i.e.,* $T = 1$. Let $\Delta_{\texttt{rows}}$ and $\Delta_{\texttt{cols}}$ denote the set of all distributions over rows and colums of M, respectively. Let us extend the notation $M(i, j)$ to distributions over rows/columns: for any $p \in \Delta_{\texttt{rows}}, q \in \Delta_{\texttt{cols}}$, we define

$$M(p, q) := \mathop{\mathbb{E}}_{i \sim p,\, j \in q} [\, M(i, j)\,] = p^\top M q,$$

where distributions are interpreted as column vectors.

Suppose ALG chooses a row from some distribution $p \in \Delta_{\texttt{rows}}$ known to the adversary. Then the adversary can choose a distribution $q \in \Delta_{\texttt{cols}}$ so as to maximize its expected reward $M(p, q)$. (Any column

j in the support of q would maximize $M(p,\cdot)$, too.) Accordingly, the algorithm should choose p so as to minimize its maximal cost,

$$f(p) := \sup_{q\in\Delta_{\texttt{cols}}} M(p,q) = \max_{\text{columns } j} M(p,j). \tag{9.2}$$

A distribution $p=p^*$ that minimizes $f(p)$ exactly is called a *minimax strategy*. At least one such p^* exists, as an argmin of a continuous function on a closed and bounded set. A minimax strategy achieves cost

$$v^* = \min_{p\in\Delta_{\texttt{rows}}}\ \max_{q\in\Delta_{\texttt{cols}}} M(p,q),$$

called the *minimax value* of the game M. Note that p^* guarantees cost at most v^* against any adversary:

$$M(p^*,j) \le v^* \quad \forall \text{ column } j. \tag{9.3}$$

Arbitrary adversary. We apply the regret property of the algorithm and deduce that algorithm's expected average costs are approximately at least as good as the minimax value v^*. Indeed,

$$\texttt{cost}^* = \min_{\text{rows } i} \texttt{cost}(i) \le \mathop{\mathbb{E}}_{i\sim p^*}[\,\texttt{cost}(i)\,] = \textstyle\sum_{t=1}^T M(p^*,j_t) \le T\,v^*.$$

Recall that p^* is the minimax strategy, and the last inequality follows by Equation (9.3). By definition of regret, we have:

Theorem 9.1. *For an arbitrary adversary* ADV *it holds that*

$$\tfrac{1}{T}\,\texttt{cost}(\texttt{ALG}) \le v^* + R(T)/T.$$

In particular, if we (only) posit sublinear expected regret, $\mathbb{E}[R(t)] = o(t)$, then the expected average cost $\frac{1}{T}\,\mathbb{E}[\texttt{cost}(\texttt{ALG})]$ is asymptotically upper-bounded by v^*.

Best-response adversary. Assume a specific (and very powerful) adversary called the *best-response adversary*. Such adversary operates as follows: in each round t, it chooses a column j_t which maximizes its expected reward given the *history*

$$\mathcal{H}_t := (\,(i_1,j_1)\,,\ \ldots\ ,(i_{t-1},j_{t-1})\,)\,, \tag{9.4}$$

which encompasses what happened in the previous rounds. In a formula,

$$j_t = \min\left(\operatorname*{argmax}_{\text{columns } j} \mathbb{E}\left[M(i_t,j)\mid \mathcal{H}_t\,\right]\right), \tag{9.5}$$

where the expectation is over the algorithm's choice of row i_t.[2]

We consider the algorithm's average play $\bar{\imath} := (i_1+\ldots+i_T)/T \in \Delta_{\texttt{rows}}$, and argue that its expectation, $\mathbb{E}[\bar{\imath}]$, performs well against an arbitrary column j. This is easy to prove:

$$\begin{aligned}
\mathbb{E}[M(i_t,\,j)] &= \mathbb{E}[\ \mathbb{E}[M(i_t,\,j)\mid \mathcal{H}_t]\ ] \\
&\le \mathbb{E}[\ \mathbb{E}[M(i_t,\,j_t)\mid \mathcal{H}_t]\ ] && \textit{(by best response)} \\
&= \mathbb{E}[M(i_t,\,j_t)] \\
M(\,\mathbb{E}[\bar{\imath}],j\,) &= \tfrac{1}{T}\textstyle\sum_{t=1}^T \mathbb{E}[M(i_t,\,j)] && \textit{(by linearity of } M(\cdot,\cdot)\textit{)} \\
&\le \tfrac{1}{T}\textstyle\sum_{t=1}^T \mathbb{E}[\ M(i_t,j_t)\ ] \\
&= \tfrac{1}{T}\,\mathbb{E}[\texttt{cost}(\texttt{ALG})]. && (9.6)
\end{aligned}$$

[2]Note that the column j in the argmax need not be unique, hence the $\min$ in front of the argmax. Any other tie-breaking rule would work, too. Such column j also maximizes the expected reward $\mathbb{E}\left[M(i_t,\cdot)\mid \mathcal{H}_t\,\right]$ over all distributions $q\in\Delta_{\texttt{cols}}$.

Plugging this into Theorem 9.1, we prove the following:

Theorem 9.2. *If* ADV *is the best-response adversary, as in (9.5), then*

$$M(\,\mathbb{E}[\bar{\imath}], q\,) \leq \tfrac{1}{T}\,\mathbb{E}[\texttt{cost(ALG)}] \leq v^* + \mathbb{E}[R(T)]/T \qquad \forall q \in \Delta_{\texttt{cols}}.$$

Thus, if $\mathbb{E}[R(t)] = o(t)$ then ALG's expected average play $\mathbb{E}[\bar{\imath}]$ asymptotically achieves the minimax property of p^*, as expressed by Equation (9.3).

9.2 The minimax theorem

A fundamental fact about the minimax value is that it equals the maximin value:

$$\min_{p\in\Delta_{\texttt{rows}}} \; \max_{q\in\Delta_{\texttt{cols}}} \; M(p,q) = \max_{q\in\Delta_{\texttt{cols}}} \; \min_{p\in\Delta_{\texttt{rows}}} \; M(p,q). \tag{9.7}$$

In other words, the $\max$ and the $\min$ can be switched. The maximin value is well-defined, in the sense that the $\max$ and the $\min$ exist, for the same reason as they do for the minimax value.

The maximin value emerges naturally in the single-round game if one switches the roles of ALG and ADV so that the former controls the columns and the latter controls the rows (and M represents algorithm's rewards rather than costs). Then a *maximin strategy* – a distribution $q = q^* \in \Delta_{\texttt{cols}}$ that maximizes the right-hand size of (9.20) – arises as the algorithm's best response to a best-responding adversary. Moreover, we have an analog of Equation (9.3):

$$M(p,q^*) \geq v^\sharp \quad \forall p \in \Delta_{\texttt{rows}}, \tag{9.8}$$

where $v^\sharp$ be the right-hand side of (9.20). In words, maximin strategy q^* guarantees reward at least $v^\sharp$ against any adversary. Now, since $v^* = v^\sharp$ by Equation (9.20), we can conclude the following:

Corollary 9.3. $M(p^*,q^*) = v^*$*, and the pair* (p^*,q^*) *forms a* Nash equilibrium*, in the sense that*

$$p^* \in \operatorname*{argmin}_{p\in\Delta_{\texttt{rows}}} M(p,q^*) \textit{ and } q^* \in \operatorname*{argmax}_{q\in\Delta_{\texttt{cols}}} M(p^*,q). \tag{9.9}$$

With this corollary, Equation (9.20) is a celebrated early result in mathematical game theory, known as the *minimax theorem*. Surprisingly, it admits a simple alternative proof based on the existence of sublinear-regret algorithms and the machinery developed earlier in this chapter.

Proof of Equation (9.20). The $\geq$ direction is easy:

$$\begin{aligned} M(p,q) &\geq \min_{p'\in\Delta_{\texttt{rows}}} \; M(p',q) \qquad \forall q \in \Delta_{\texttt{cols}} \\ \max_{q\in\Delta_{\texttt{cols}}} M(p,q) &\geq \max_{q\in\Delta_{\texttt{cols}}} \; \min_{p'\in\Delta_{\texttt{rows}}} \; M(p',q). \end{aligned}$$

The $\leq$ direction is the difficult part. Let us consider a full-feedback version of the repeated game studied earlier. Let ALG be any algorithm with subliner expected regret, *e.g.,* Hedge algorithm from Chapter 5.3, and let ADV be the best-response adversary. Define

$$h(q) := \inf_{p\in\Delta_{\texttt{rows}}} \; M(p,q) = \min_{\text{rows } i} \; M(i,q) \tag{9.10}$$

for each distribution $q \in \Delta_{\texttt{cols}}$, similarly to how $f(p)$ is defined in (9.2). We need to prove that

$$\min_{p\in\Delta_{\texttt{rows}}} f(p) \leq \max_{q\in\Delta_{\texttt{cols}}} h(q). \tag{9.11}$$

Let us take care of some preliminaries. Let $\bar{\jmath} := (j_1 + \ldots + j_T)/T \in \Delta_{\texttt{cols}}$ be average play of ADV. Then:

$$\begin{aligned}
\mathbb{E}[\texttt{cost}(i)] &= \textstyle\sum_{t=1}^{T} \mathbb{E}[\,M(i, j_t)\,] = T\;\mathbb{E}[\,M(i,\bar{\jmath})\,] = T\,M(i,\,\mathbb{E}[\bar{\jmath}]) \qquad \text{for each row } i\\
\tfrac{1}{T}\;\mathbb{E}[\texttt{cost}^*] &\leq \tfrac{1}{T}\min_{\text{rows } i} \mathbb{E}[\texttt{cost}(i)] = \min_{\text{rows } i} M(i,\,\mathbb{E}[\bar{\jmath}]) = \min_{p\in\Delta_{\texttt{rows}}} M(p,\,\mathbb{E}[\bar{\jmath}]) = h(\mathbb{E}[\bar{\jmath}]).
\end{aligned} \tag{9.12}$$

The crux of the argument is as follows:

$$\begin{aligned}
\min_{p\in\Delta_{\texttt{rows}}} f(p) &\leq f(\,\mathbb{E}[\bar{\imath}]\,)\\
&\leq \tfrac{1}{T}\;\mathbb{E}[\texttt{cost}(\texttt{ALG})] && \textit{(by best response, see (9.6))}\\
&= \tfrac{1}{T}\;\mathbb{E}[\texttt{cost}^*] + \tfrac{\mathbb{E}[R(T)]}{T} && \textit{(by definition of regret)}\\
&\leq h(\mathbb{E}[\bar{\jmath}]) + \tfrac{\mathbb{E}[R(T)]}{T} && \textit{(using } h(\cdot)\textit{, see (9.12))}\\
&\leq \max_{q\in\Delta_{\texttt{cols}}} h(q) + \tfrac{\mathbb{E}[R(T)]}{T}.
\end{aligned}$$

Now, taking $T \to \infty$ implies Equation (9.11) because $\mathbb{E}[R(T)]/T \to 0$, completing the proof. □

9.3 Regret-minimizing adversary

The most interesting version of our game is when the adversary itself is a regret-minimizing algorithm (possibly in a different feedback model). More formally, we posit that after each round t the adversary observes its reward $M(i_t, j_t)$ and auxiliary feedback $\mathcal{F}'_t = \mathcal{F}'(t, i_t, j_t, M)$, where $\mathcal{F}'$ is some fixed feedback model. We consider the regret of ADV in this feedback model, denote it $R'(T)$. We use the minimax theorem to prove that $(\bar{\imath}, \bar{\jmath})$, the average play of ALG and ADV, approximates the Nash equilibrium (p^*, q^*).

First, let us express $\texttt{cost}^*$ in terms of the function $h(\cdot)$ from Equation (9.10):

$$\begin{aligned}
\texttt{cost}(i) &= \textstyle\sum_{t=1}^{T} M(i, j_t) = T\,M(i,\bar{\jmath}) \qquad \text{for each row } i\\
\tfrac{1}{T}\,\texttt{cost}^* &= \min_{\text{rows } i} M(i,\,\bar{\jmath}) = h(\bar{\jmath}).
\end{aligned} \tag{9.13}$$

Let us analyze ALG's costs:

$$\begin{aligned}
\tfrac{1}{T}\,\texttt{cost}(\texttt{ALG}) - \tfrac{R(T)}{T} &= \tfrac{1}{T}\,\texttt{cost}^* && \textit{(regret for } \texttt{ALG}\textit{)}\\
&= h(\bar{\jmath}) && \textit{(using } h(\cdot)\textit{, see (9.13))}\\
&\leq h(q^*) = v^\sharp && \textit{(by definition of maximin strategy } q^*\textit{)}\\
&= v^* && \textit{(by Equation (9.20))}
\end{aligned} \tag{9.14}$$

Similarly, analyze the rewards of ADV. Let $\texttt{rew}(j) = \sum_{t=1}^{T} M(i_t, j)$ be the total reward collected by the adversary for always choosing column j, and let

$$\texttt{rew}^* := \max_{\text{columns } j} \texttt{rew}(j)$$

be the reward of the "best observed column". Let's take care of some formalities:

$$\begin{aligned}
&\texttt{rew}(j) = \textstyle\sum_{t=1}^{T} M(i_t, j) = T\, M(\bar{\imath},\, j) \qquad \text{for each column } j \\
&\tfrac{1}{T}\,\texttt{rew}^* = \max_{\text{columns } j} M(\bar{\imath},\, j) = \max_{q\in\Delta_{\texttt{cols}}} M(\bar{\imath},\, q) = f(\bar{\imath}).
\end{aligned} \tag{9.15}$$

Now, let us use the regret of ADV:

$$\begin{aligned}
\tfrac{1}{T}\,\texttt{cost}(\texttt{ALG}) + \tfrac{R'(T)}{T} &= \tfrac{1}{T}\,\texttt{rew}(\texttt{ADV}) + \tfrac{R'(T)}{T} && \textit{(zero-sum game)} \\
&= \tfrac{1}{T}\,\texttt{rew}^* && \textit{(regret for } \texttt{ADV}\textit{)} \\
&= f(\bar{\imath}) && \textit{(using } f(\cdot)\textit{, see (9.15))} \\
&\geq f(p^*) = v^* && \textit{(by definition of minimax strategy } p^*\textit{)}
\end{aligned} \tag{9.16}$$

Putting together (9.14) and (9.16), we obtain

$$v^* - \tfrac{R'(T)}{T} \leq f(\bar{\imath}) - \tfrac{R'(T)}{T} \leq \tfrac{1}{T}\,\texttt{cost}(\texttt{ALG}) \leq h(\bar{\jmath}) + \tfrac{R(T)}{T} \leq v^* + \tfrac{R(T)}{T}.$$

Thus, we have the following theorem:

Theorem 9.4. *Let $R'(T)$ be the regret of* ADV*. Then the average costs/rewards converge, in the sense that*

$$\tfrac{1}{T}\,\texttt{cost}(\texttt{ALG}) = \tfrac{1}{T}\,\texttt{rew}(\texttt{ADV}) \in \left[v^* - \tfrac{R'(T)}{T},\; v^* + \tfrac{R(T)}{T}\right].$$

*Moreover, the average play $(\bar{\imath}, \bar{\jmath})$ forms an ϵ_T-*approximate Nash equilibrium*, with $\epsilon_T := \frac{R(T)+R'(T)}{T}$:*

$$\begin{aligned}
M(\bar{\imath},\, q) &\leq v^* + \epsilon_T \qquad \forall q \in \Delta_{\texttt{cols}}, \\
M(p,\, \bar{\jmath}) &\geq v^* - \epsilon_T \qquad \forall p \in \Delta_{\texttt{rows}}.
\end{aligned}$$

The guarantees in Theorem 9.4 are strongest if ALG and ADV admit high-probability regret bounds, *i.e.,* if

$$R(T) \leq \tilde{R}(T) \text{ and } R'(T) \leq \tilde{R}'(T) \tag{9.17}$$

with probability at least $1-\gamma$, for some functions $\tilde{R}(t)$ and $\tilde{R}'(t)$. Then the guarantees in Theorem 9.4 hold, with $R(T)$ and $R'(T)$ replaced with, resp., $\tilde{R}(T)$ and $\tilde{R}'(T)$, whenever (9.17) holds. In particular, algorithm Hedge and a version of algorithm Exp3 achieve high-probability regret bounds, resp., for full feedback with $R(T) = O(\sqrt{T \log K})$, for bandit feedback with $R(T) = O(\sqrt{TK \log K})$, where K is the number of actions (Freund and Schapire, 1997; Auer et al., 2002b).

Further, we recover similar but weaker guarantees even if we only have a bound on *expected* regret:

Corollary 9.5. $\frac{1}{T}\,\mathbb{E}[\texttt{cost}(\texttt{ALG})] = \frac{1}{T}\,\mathbb{E}[\texttt{rew}(\texttt{ADV})] \in \left[v^* - \frac{\mathbb{E}[R'(T)]}{T},\; v^* + \frac{\mathbb{E}[R(T)]}{T}\right].$

Moreover, the expected average play $(\mathbb{E}[\bar{\imath}],\, \mathbb{E}[\bar{\jmath}])$ forms an $\mathbb{E}[\epsilon_T]$-approximate Nash equilibrium.

This is because $\mathbb{E}[\, M(\bar{\imath},\, q)\,] = M(\mathbb{E}[\bar{\imath}], q)$ by linearity, and similarly for $\bar{\jmath}$.

9.4 Beyond zero-sum games: coarse correlated equilibrium

What can we prove if the game is not zero-sum? While we would like to prove convergence to a Nash Equilibrium, like in Theorem 9.4, this does not hold in general. However, one can prove a weaker notion of convergence, as we explain below. Consider distributions over row-column pairs, let $\Delta_{\texttt{pairs}}$ be the set of all such distributions. We are interested in the average distribution defined as follows:

$$\bar{\sigma} := (\sigma_1 + \ldots + \sigma_T)/T \in \Delta_{\texttt{pairs}}, \qquad \text{where } \sigma_t := p_t \times q_t \in \Delta_{\texttt{pairs}} \tag{9.18}$$

We argue that $\bar{\sigma}$ is, in some sense, an approximate equilibrium.

Imagine there is a "coordinator" who takes some distribution $\sigma \in \Delta_{\texttt{pairs}}$, draws a pair (i,j) from this distribution, and recommends row i to ALG and column j to ADV. Suppose each player has only two choices: either "commit" to following the recommendation before it is revealed, or "deviate" and not look at the recommendation. The equilibrium notion that we are interested here is that each player wants to "commit" given that the other does.

Formally, assume ADV "commits". The expected costs for ALG are

$$\begin{aligned} U_\sigma &:= \mathop{\mathbb{E}}_{(i,j)\sim\sigma} M(i,j) && \text{if ALG "commits"}, \\ U_\sigma(i_0) &:= \mathop{\mathbb{E}}_{(i,j)\sim\sigma} M(i_0,j) && \text{if ALG "deviates" and chooses row } i_0 \text{ instead.} \end{aligned}$$

Distribution $\sigma \in \Delta_{\texttt{pairs}}$ is a *coarse correlated equilibrium* (CCE) if $U_\sigma \geq U_\sigma(i_0)$ for each row i_0, and a similar property holds for ADV.

We are interested in the approximate version of this property: $\sigma \in \Delta_{\texttt{pairs}}$ is an ϵ-approximate CCE if

$$U_\sigma \geq U_\sigma(i_0) - \epsilon \qquad \text{for each row } i_0 \tag{9.19}$$

and similarly for ADV. It is easy to see that distribution $\bar{\sigma}$ achieves this with $\epsilon = \frac{\mathbb{E}[R(T)]}{T}$. Indeed,

$$\begin{aligned} U_{\bar{\sigma}} &:= \mathop{\mathbb{E}}_{(i,j)\sim\bar{\sigma}} [\, M(i,j) \,] = \tfrac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{(i,j)\sim\sigma_t} [\, M(i,j) \,] = \tfrac{1}{T}\, \mathbb{E}[\texttt{cost}(\texttt{ALG})] \\ U_{\bar{\sigma}}(i_0) &:= \mathop{\mathbb{E}}_{(i,j)\sim\bar{\sigma}} [\, M(i_0,j) \,] = \tfrac{1}{T} \sum_{t=1}^{T} \mathop{\mathbb{E}}_{j\sim q_t} [\, M(i_0,j) \,] = \tfrac{1}{T}\, \mathbb{E}[\texttt{cost}(i_0)]. \end{aligned}$$

Hence, $\bar{\sigma}$ satisfies Equation (9.19) with $\epsilon = \frac{\mathbb{E}[R(T)]}{T}$ by definition of regret. Thus:

Theorem 9.6. *Distribution $\bar{\sigma}$ defined in (9.18) forms an ϵ_T-approximate CCE, where $\epsilon_T = \frac{\mathbb{E}[R(T)]}{T}$.*

9.5 Bibliographic remarks and further directions

Our analysis can be refined for algorithm Hedge and/or the best-response adversary (as defined in Equation (9.5)). First, the best-response adversary can be seen as a regret-minimizing algorithm which satisfies Theorem 9.4 with $\mathbb{E}[R'(T)] \leq 0$; see Exercise 9.2. Second, the Hedge vs. Hedge game satisfies the approximate Nash property from Theorem 9.4 with high probability, even though the algorithm itself does not satisfy a high-probability regret bound; see Exercise 9.3(a). Third, the same game admits the same property *with probability* 1, albeit in a modified feedback model; see Exercise 9.3(b). Finally, the Hedge vs. Best

Response game also satisfies the approximate Nash property with probability 1; see Exercise 9.3(c). The last two results can also be used for computing an approximate Nash equilibrium for a known matrix M.

The results in this chapter admit multiple extensions, some of which are discussed below.

- Theorem 9.4 can be extended to *stochastic games*, where in each round t, the game matrix M_t is drawn independently from some fixed distribution over game matrices (see Exercise 9.4).
- One can use algorithms with better regret bounds if the game matrix M allows it. For example, ALG can be an algorithm for Lipschitz bandits if all functions $M(\cdot, j)$ satisfy a Lipschitz condition.[3]
- Theorem 9.4 and the minimax theorem can be extended to infinite action sets, as long as ALG and ADV admit $o(T)$ expected regret for the repeated game with a given game matrix; see Exercise 9.5 for a precise formulation.
- $\texttt{cost}(\texttt{ALG})$ can sometimes converge faster, as a function of T, than suggested by the analysis in this chapter. Convergence of $\texttt{cost}(\texttt{ALG})$ to the corresponding equilibrium cost v^* is characterized by the *convergence rate* $\frac{|\texttt{cost}(\texttt{ALG})-v^*|}{T}$. Plugging in generic $\tilde{O}(\sqrt{T})$ regret bounds into Theorem 9.4 yields convergence rate $\tilde{O}\left(\frac{1}{\sqrt{T}}\right)$. However, recent literature obtains $\tilde{O}(\frac{1}{t})$ convergence rates, under various assumptions about the game matrix, the feedback model, or the algorithms themselves (Rakhlin and Sridharan, 2013; Daskalakis et al., 2015; Syrgkanis et al., 2015; Foster et al., 2016; Wei and Luo, 2018). This literature does not rely on "generic" regret bounds, and instead derives improved regret bounds or algorithms that engage in a repeated game.

9.6 Exercises and Hints

Exercise 9.1 (game-theory basics).

(a) Prove that a distribution $p \in \Delta_{\texttt{rows}}$ satisfies Equation (9.3) if and only if it is a minimax strategy.

Hint: $M(p^*, q) \leq f(p^*)$ if p^* is a minimax strategy; $f(p) \leq v^*$ if p satisfies (9.3).

(b) Prove that $p \in \Delta_{\texttt{rows}}$ and $q \in \Delta_{\texttt{cols}}$ form a Nash equilibrium if and only if p is a minimax strategy and q is a maximin strategy.

Exercise 9.2 (best-response adversary). The best-response adversary, as defined in Equation (9.5), can be seen as an algorithm in the setting of Chapter 9.3. (More formally, this algorithm knows ALG and the game matrix M, and receives auxiliary feedback $\mathcal{F}_t = j_t$.) Prove that its expected regret satisfies $\mathbb{E}[R'(T)] \leq 0$.

Take-away: Thus, Theorem 9.4 applies to the best-response adversary, with $\mathbb{E}[R'(T)] \leq 0$.

Exercise 9.3 (Hedge vs. Hedge). Suppose both ALG and ADV are implemented by algorithm Hedge. Assume that M is a $K \times K$ matrix with entries in $[0, 1]$, and the parameter in Hedge is $\epsilon = \sqrt{(\ln K)/(2T)}$.

(a) Consider the full-feedback model. Prove that the average play $(\bar{\imath}, \bar{\jmath})$ forms a δ_T-approximate Nash equilibrium with high probability (*e.g.,* with probability at least $1-T^{-2}$), where $\delta_T = O\left(\sqrt{\frac{\ln(KT)}{T}}\right)$.

[3]That is, if $|M(i,j) - M(i',j)| \leq D(i,i')$ for all rows i, i' and all columns j, for some known metric D on rows.

(b) Consider a modified feedback model: in each round t, `ALG` is given costs $M(\cdot, q_t)$, and `ADV` is given rewards $M(p_t, \cdot)$, where $p_t \in \Delta_{\texttt{rows}}$ and $q_t \in \Delta_{\texttt{cols}}$ are the distributions chosen by `ALG` and `ADV`, respectively. Prove that the average play $(\bar{\imath}, \bar{\jmath})$ forms a δ_T-approximate Nash equilibrium.

(c) Suppose `ADV` is the best-response adversary, as per Equation (9.5), and `ALG` is `Hedge` with full feedback. Prove that $(\bar{\imath}, \bar{\jmath})$ forms a δ_T-approximate Nash equilibrium.

Hint: Follow the steps in the proof of Theorem 9.4, but use the probability-1 performance guarantee for `Hedge` (Equation (5.10) on page 63) instead of the standard definition of regret (9.1).

For part (a), define $\widetilde{\texttt{cost}}(\texttt{ALG}) := \sum_{t\in[T]} M(p_t, j_t)$ and $\widetilde{\texttt{rew}}(\texttt{ADV}) := \sum_{t\in[T]} M(i_t, q_t)$, and use Equation (5.10) to conclude that they are close to $\texttt{cost}^*$ and $\texttt{rew}^*$, respectively. Apply Azuma-Hoeffding inequality to prove that both $\widetilde{\texttt{cost}}(\texttt{ALG})$ and $\widetilde{\texttt{rew}}(\texttt{ADV})$ are close to $\sum_{t\in[T]} M(p_t, q_t)$.

For part (b), define $\widetilde{\texttt{cost}}(\texttt{ALG}) := \widetilde{\texttt{rew}}(\texttt{ADV}) := \sum_{t\in[T]} M(p_t, q_t)$, and use Equation (5.10) to conclude that it is close to both $cost^*$ and $\texttt{rew}^*$.

For part (c), define $\widetilde{\texttt{cost}}(\texttt{ALG}) = \widetilde{\texttt{rew}}(\texttt{ADV})$ and handle `Hedge` as in part (b). Modify (9.16), starting from $\widetilde{\texttt{rew}}(\texttt{ADV})$, to handle the best-response adversary.

Exercise 9.4 (stochastic games). Consider an extension to stochastic games: in each round t, the game matrix M_t is drawn independently from some fixed distribution over game matrices. Assume all matrices have the same dimensions (number of rows and columns). Suppose both `ALG` and `ADV` are regret-minimizing algorithms, as in Section 9.3, and let $R(T)$ and $R'(T)$ be their respective regret. Prove that the average play $(\bar{\imath}, \bar{\jmath})$ forms a δ_T-approximate Nash equilibrium for the expected game matrix $M = \mathbb{E}[M_t]$, with

$$\delta_T = \tfrac{R(T)+R'(T)}{T} + \texttt{err}, \text{ where } \texttt{err} = \left| \textstyle\sum_{t\in[T]} M_t(i_t, j_t) - M(i_t, j_i) \right|.$$

Hint: The total cost/reward is now $\sum_{t\in[T]} M_t(i_t, j_t)$. Transition from this to $\sum_{t\in[T]} M(i_t, j_t)$ using the error term `err`, and follow the steps in the proof of Theorem 9.4.

Note: If all matrix entries are in the $[a, b]$ interval, then $\texttt{err} \in [0, b-a]$, and by Azuma-Hoeffing inequality $\texttt{err} < O(b-a)\sqrt{\log(T)/T}$ with probability at most $1 - T^{-2}$.

Exercise 9.5 (infinite action sets). Consider an extension of the repeated game to infinite action sets. Formally, `ALG` and `ADV` have action sets I, J, resp., and the game matrix M is a function $I \times J \to [0, 1]$. I and J can be arbitrary sets with well-defined probability measures such that each singleton set is measurable. Assume there exists a function $\tilde{R}(T) = o(T)$ such that expected regret of `ALG` is at most $\tilde{R}(T)$ for any `ADV`, and likewise expected regret of `ADV` is at most $\tilde{R}(T)$ for any `ALG`.

(a) Prove an appropriate versions of the minimax theorem:

$$\inf_{p\in\Delta_{\texttt{rows}}} \sup_{j\in J} \int M(p, j)\,\mathrm{d}p = \sup_{q\in\Delta_{\texttt{cols}}} \inf_{i\in I} \int M(i, q)\,\mathrm{d}q, \tag{9.20}$$

where $\Delta_{\texttt{rows}}$ and $\Delta_{\texttt{cols}}$ are now the sets of all probability measures on I and J.

(b) Formulate and prove an appropriate version of Theorem 9.4.

Hint: Follow the steps in Sections 9.2 and 9.3 with minor modifications. Distributions over rows and columns are replaced with probability measures over I and J. Maxima and minima over I and J are replaced with $\sup$ and $\inf$. Best response returns a distribution over Y (rather than a particular column).

Chapter 10

Bandits with Knapsacks *(rev. March'19)*

We study "Bandits with Knapsacks" (`BwK`), a general framework for bandit problems with "global constraints" such as supply constraints in dynamic pricing. We introduce the framework, support it with several motivating examples, and explain how it is more difficult compared to the "usual" bandits. We describe three different algorithms for `BwK` that achieve optimal regret bounds.

10.1 Definitions, examples, and discussion

Motivating example. We start with a motivating example: *dynamic pricing with limited supply*. The basic version of this problem is as follows. The algorithm is a seller, with a limited inventory of B identical items for sale. There are T rounds. In each round t, the algorithm chooses a price $p_t \in [0, 1]$ and offers one item for sale at this price. A new customer arrives, having in mind some value v_t for this item (known as *private value*). We posit that v_t is drawn independently from some fixed but unknown distribution. The customer buys the item if and only if $v_t \geq p_t$, and leaves. The algorithm stops after T rounds or after there are no more items to sell, whichever comes first. The algorithm's goal is to maximize revenue from the sales; there is no premium or rebate for the left-over items. Recall that the special case $B = T$ (*i.e.,* unlimited supply of items) falls under "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.

More generally, the algorithm may have $d > 1$ products in the inventory, with a limited supply of each. In each round t, the algorithm chooses a price $p_{t,i}$ for each product i, and offers one copy of each product for sale. A new customer arrives, with a vector of private values $v_t = (v_{t,1}, \ldots, v_{t,d})$, and buys each product i such that $v_{t,i} \geq p_{t,i}$. The vector v_t is drawn independently from some distribution.[1] We interpret this scenario as a bandit problem with IID rewards, where actions correspond to price vectors $p_t = (p_{t,1}, \ldots, p_{t,d})$, and we have a separate "global constraint" on each product.

General framework. We introduce a general framework for bandit problems with global constraints, called "bandits with knapsacks". In this framework, there are several constrained *resources* being consumed by the algorithm, such as the inventory of products in the dynamic pricing problem. One of these resources is time: each arm consumes one unit of the "time resource" in each round, and its budget is the time horizon T. The algorithm stops when the total consumption of some resource i exceeds its respective budget B_i.

[1]If the values are independent across products, *i.e.,* $v_{t,1}, \ldots, v_{t,d}$ are mutually independent random variables, then the problem can be decoupled into d separate per-product problems. However, in general the values may be correlated.

Problem protocol: Bandits with Knapsacks (`BwK`)

Parameters: K arms, d resources with respective budgets $B_1, \ldots, B_d \in [0,T]$.
In each round $t = 1,2,3 \ldots$:
1. Algorithm chooses an arm $a_t \in [K]$.
2. Outcome vector $\vec{o}_t = (r_t; c_{t,1}, \ldots, c_{t,d}) \in [0,1]^{d+1}$ is observed,
   where r_t is the algorithm's reward, and $c_{t,i}$ is consumption of each resource i.

Algorithm stops when the total consumption of some resource i exceeds its budget B_i.

In each round, an 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. As a technical assumption, the reward and consumption of each resource in each round lie in $[0,1]$. We posit the "IID assumption", which now states that for each arm a the outcome vector is sampled independently from a fixed distribution over outcome vectors. Formally, an instance of `BwK` is specified by parameters T, K, d, budgets $B_1, \ldots, B_d$, and a mapping from arms to distributions over outcome vectors. The algorithm's goal is to maximize its *adjusted total reward*: the total reward over all rounds but the very last one.

The name "bandits with knapsacks" comes from 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, the number of items sold). However, in `BwK` the "value" and "size" of a given action are not known in advance.

Remark 10.1. The special case $B_1 = \ldots = B_d = T$ is just "bandits with IID rewards", as in Chapter 1.

Remark 10.2. An algorithm can continue while there are sufficient resources to do so, even if it almost runs out of some resources. Then the algorithm should only choose "safe" arms if at all possible, where an arm is called "safe" if playing this arm in the current round cannot possibly cause the algorithm to stop.

Remark 10.3. We posit that one of the arms, called the *null arm*, brings no reward and consumes no resource except the "time resource". Playing this arm is tantamount to skipping a round. This assumption is essential for several key steps in the analysis.

Remark 10.4. Without loss of generality, the outcome vectors are chosen as follows. In each round t, the *outcome matrix* $\mathbf{M}_t \in [0,1]^{K\times(d+1)}$ is drawn from some fixed distribution. Rows of $\mathbf{M}_t$ correspond to arms: for each arm $a \in [K]$, the a-th row of $\mathbf{M}_t$ is

$$\mathbf{M}_t(a) = (r_t(a);\, c_{t,1}(a), \ldots, c_{t,d}(a)),$$

so that $r_t(a)$ is the reward and $c_{t,i}(a)$ is the consumption of each resource i. The round-t outcome vector is defined as the a_t-th row of this matrix: $\vec{o}_t = \mathbf{M}_t(a_t)$. Only this row is revealed to the algorithm.

Examples. We illustrate the generality of `BwK` with several examples. In all these examples, "time resource" is the last component of the outcome vector.

- *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

$$\vec{o}_t = \begin{cases} (p,1,1) & \text{if price } p \text{ is accepted} \\ (0,0,1) & \text{otherwise.} \end{cases} \tag{10.1}$$

- *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

$$\vec{o}_t = \begin{cases} (1,p,1) & \text{if price } p \text{ is accepted} \\ (0,0,1) & \text{otherwise.} \end{cases} \tag{10.2}$$

- *Pay-per-click ad allocation.* 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:

$$\vec{o}_t = \begin{cases} (r_a,r_a,1) & \text{if ad } a \text{ is clicked} \\ (0,0,1) & \text{otherwise.} \end{cases} \tag{10.3}$$

- *Repeated auctions.* 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.

  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 is:

$$\vec{o}_t = \begin{cases} (p_t,1,1) & \text{if an item is sold at price } p_t \\ (0,0,1) & \text{otherwise.} \end{cases}$$

  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 is

$$\vec{o}_t = \begin{cases} (1, p_t, 1) & \text{if the bidder wins the item and pays } p_t \\ (0,0,1) & \text{otherwise.} \end{cases}$$

  The payment p_t is determined by the chosen bid b_t, other bids, and the rules of the auction.

Discussion. Let us compare BwK with the "standard" bandit problem with IID rewards. BwK is more complicated in several ways. First, resource consumption during exploration may limit the algorithm's ability to exploit in the future rounds. A stark consequence is that Explore-first algorithm fails if the budgets are too small, see Exercise 10.1(a). Second, per-round expected reward is no longer the right objective. An arm with high per-round expected reward may be undesirable because of high resource consumption. Instead, one needs to think about the *total* expected reward over the entire time horizon. Finally, learning the best arm is no longer the right objective! Instead, one is interested in the best fixed *distribution* over arms. A fixed distribution over arms can perform much better than the best fixed arm in many examples.

To illustrate the last point, consider the following example. There are two arms $i \in \{1,2\}$ and two resources $j \in \{1,2\}$ (other than time). In each round, each arm i yields reward 1, and consumes $\mathbf{1}_{\{i=j\}}$ units of each resource j. Both resources have budget B, and time horizon is $T = 2B$. Then playing a fixed action gives the total reward of B, whereas alternating the two actions gives the total reward of $2B$. 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.

Benchmark. We compete with a very strong benchmark: the best all-knowing algorithm, *i.e.,* the best algorithm tailored to the specific problem instance $\mathcal{I}$:

$$\texttt{OPT} \triangleq \sup_{\text{algorithms } \texttt{ALG}} \mathbb{E}[\texttt{REW}(\texttt{ALG} \mid \mathcal{I})], \tag{10.4}$$

where $\texttt{REW}(\texttt{ALG} \mid \mathcal{I})$ is the adjusted total reward of algorithm ALG on problem instance $\mathcal{I}$. We shall see that competing with OPT is essentially the same as competing with the best fixed distribution over arms.

10.2 LagrangeBwK: a game-theoretic algorithm for BwK

We present an algorithm for BwK, called LagrangeBwK, which builds on the zero-sum games framework developed in Chapter 9. On a high level, our approach consists of four steps:

Linear relaxation We consider a relaxation of BwK in which a fixed distribution D over arms is used in all rounds, and outcomes are equal to their expected values. This relaxation can be expressed as a linear program for optimizing D, whose per-round value is denoted $\texttt{OPT}_{\texttt{LP}}$ We prove that $\texttt{OPT}_{\texttt{LP}} \geq \texttt{OPT}/T$.

Lagrange game We consider the Lagrange function $\mathcal{L}$ associated with this linear program. We focus on the *Lagrange game*: a zero-sum game, where one player chooses an arm a, the other player chooses a resource i, and the payoff is $\mathcal{L}(a, i)$. We prove that the value of this game is $\texttt{OPT}_{\texttt{LP}}$.

Repeated Lagrange game We consider a repeated version of this game. In each round t, the payoffs are given by Lagrange function $\mathcal{L}_t$, which is defined by this round's outcomes in a similar manner as $\mathcal{L}$ is defined by the expected outcomes. Each player is controlled by a regret-minimizing algorithm. The analysis from Chapter 9 connects the average play in this game with its Nash equilibrium.

Reward at the stopping time The final step argues that the reward at the stopping time is large compared to the "relevant" value of the Lagrange function (which in turn is large enough because of the Nash property). Interestingly, this step only relies on the definition of $\mathcal{L}$, and holds for any algorithm.

Conceptually, these steps connect BwK to the linear program, to the Lagrange game, to the repeated game, and back to BwK, see Figure 10.1. We flesh out the details in what follows.

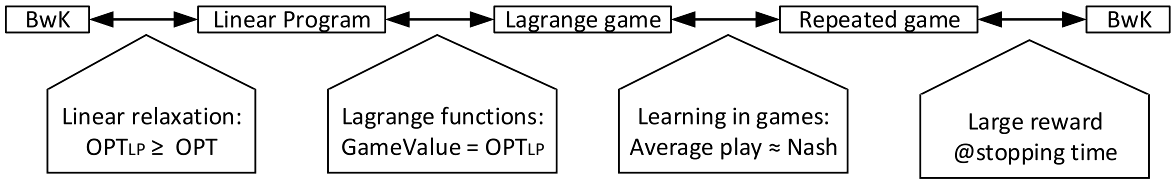

BwK
Linear Program
Lagrange game
Repeated game
BwK
Linear relaxation: OPTLP ≥ OPT
Lagrange functions: GameValue = OPTLP
Learning in games: Average play ≈ Nash
Large reward @stopping time

Figure 10.1: The conceptual layout of LagrangeBwK.

Preliminaries

Let $r(a)$ be the expected per-round reward of arm a, and $c_i(a)$ be the expected per-round consumption of a given resource i. The sets of rounds, arms and resources are denoted $[T]$, $[K]$ and $[d]$, respectively. The distributions over arms and resources are denoted Δ_K and Δ_d.

Let $B = \min_{i\in[d]} B_i$ be the smallest budget. Without loss of generality, we rescale the problem so that all budgets are B. For this, divide the per-round consumption of each resource i by B_i/B. In particular, the per-round consumption of the time resource is now B/T.

The adjusted total reward of algorithm ALG is denoted REW(ALG).

Step 1: Linear relaxation

Let us define a relaxation of BwK as follows. Fix some distribution D over arms. Suppose this distribution is used in a given round, *i.e.,* the arm is drawn independently from D. Let $r(D)$ be the expected reward, and $c_i(D)$ be the expected per-round consumption of each resource i:

$$r(D) = \textstyle\sum_{a\in[K]} D(a)\, r(a) \quad \text{and} \quad c_i(D) = \textstyle\sum_{a\in[K]} D(a)\, c_i(a).$$

In the relaxation distribution D is used in each round, and the reward and resource-i consumption are deterministically equal to $r(D)$ and $c_i(D)$, respectively. We are only interested in distributions D such that the algorithm does not run out of resources until round T. The problem of choosing D so as to maximize the per-round reward in the relaxation can be formulated as a linear program:

$$
\begin{array}{ll}
\text{maximize} & r(D) \\
\text{subject to} & \\
& D \in \Delta_K \\
& T \cdot c_i(D) \le B \qquad \forall i \in [d].
\end{array}
\tag{10.5}
$$

The value of this linear program is denoted $\texttt{OPT}_{\texttt{LP}}$. We claim that the corresponding total reward, $T \cdot \texttt{OPT}_{\texttt{LP}}$, is an upper bound on $\texttt{OPT}$, the best expected reward achievable in $\texttt{BwK}$.

Claim 10.5. $T \cdot \texttt{OPT}_{\texttt{LP}} \ge \texttt{OPT}$.

Proof. Fix some algorithm $\texttt{ALG}$ for $\texttt{BwK}$. Let τ be the round after which this algorithm stops. Without loss of generality, if $\tau < T$ then $\texttt{ALG}$ continues to play the null arm in all rounds $\tau, \tau+1, \ldots, T$.

Let $X_t(a) = \mathbf{1}_{\{a_t = a\}}$ be the indicator variable of the event that $\texttt{ALG}$ chooses arm a in round t. Let $D \in \Delta_K$ be the expected average play distribution: $D(a) = \mathbb{E}\left[\frac{1}{T}\sum_{t\in[T]} X_t(a)\right]$.

First, we claim that the expected total adjusted reward is $\mathbb{E}[\texttt{REW}(\texttt{ALG})] = r(D)$. Indeed,

$$
\begin{aligned}
\mathbb{E}[r_t] &= \textstyle\sum_{a\in[K]} \Pr[a_t = a] \cdot \mathbb{E}[r_t \mid a_t = a] \\
&= \textstyle\sum_{a\in[K]} \mathbb{E}[X_t(a)] \cdot r(a). \\
\mathbb{E}[\texttt{REW}] &= \sum_{t\in[T]} \mathbb{E}[r_t] = \sum_{a\in[K]} r(a) \cdot \sum_{t\in[T]} \mathbb{E}[X_t(a)] = \sum_{a\in[K]} r(a) \cdot T \cdot D(a) = T \cdot r(D).
\end{aligned}
$$

Similarly, the expected total consumption of each resource i is $\sum_{t\in[T]} \mathbb{E}[c_{t,i}] = T \cdot c_i(D)$.

Since the (modified) algorithm does not stop until time T, we have $\sum_{t\in[T]} c_{i,t} \le B$, and consequently $c_i(D) \le B/T$. Therefore, D is a feasible solution for the linear program (10.5). It follows that $\mathbb{E}[\texttt{REW}(\texttt{ALG})] = r(D) \le \texttt{OPT}_{\texttt{LP}}$. □

Step 2: Lagrange functions

Consider the Lagrange function associated with the linear program (10.5). For our purposes, this function inputs a distribution D over arms and a distribution λ over resources,

$$
\mathcal{L}(D, \lambda) := r(D) + \sum_{i\in[d]} \lambda_i \left[1 - \tfrac{T}{B}\, c_i(D)\right]. \tag{10.6}
$$

Define the *Lagrange game*: a zero-sum game, where the *primal player* chooses an arm a, the *dual player* chooses a resource i, and the payoff is given by the Lagrange function:

$$
\mathcal{L}(a, i) = r(a) + 1 - \tfrac{T}{B}\, c_i(a). \tag{10.7}
$$

The primal player receives this number as a reward, and the dual player receives it as cost. The two players move simultaneously, without observing one another.

Remark 10.6. The terms *primal player* and *dual player* are inspired by the duality in linear programming. For each linear program (LP), a.k.a. *primal* LP, there is an associated *dual* LP. Variables in (10.5) correspond to arms, and variables in its dual LP correspond to resources. Thus, the primal player chooses among the variables in the primal LP, and the dual player chooses among the variables in the dual LP.

The Lagrangian game is related to the linear program, as expressed by the following lemma.

Lemma 10.7. *Suppose (D^*, λ^*) is a mixed Nash equilibrium for the Lagrangian game. Then*

(a) $1 - \frac{T}{B}\, c_i(D^*) \geq 0$ *for each resource i, with equality if $\lambda^*_i > 0$.*

(b) D^* is an optimal solution for the linear program (10.5).

(c) The minimax value of the Lagrangian game equals the LP value: $\mathcal{L}(D^*, \lambda^*) = \texttt{OPT}_{\texttt{LP}}$.

Remark 10.8. Lagrange function is a standard notion in mathematical optimization. For an arbitrary linear program (with at least one solution and a finite LP value), the function satisfies a max-min property:

$$\min_{\lambda\in\mathbb{R}^d,\, \lambda\geq 0}\ \max_{D\in\Delta_K} \mathcal{L}(D,\lambda) = \max_{D\in\Delta_K}\ \min_{\lambda\in\mathbb{R}^d,\, \lambda\geq 0} \mathcal{L}(D,\lambda) = \texttt{OPT}_{\texttt{LP}}.$$

Because of the special structure of (10.5), we obtain the same property with $\lambda \in \Delta_d$.

Remark 10.9. In what follows, we only use part (c) of Lemma 10.7. Parts (ab) serve to prove (c), and are stated for intuition only. The property in part (a) is known as complementary slackness. The proof of Lemma 10.7 is a standard linear programming argument, not something about multi-armed bandits.

Proof of Lemma 10.7. By the definition of the mixed Nash equilibrium,

$$\mathcal{L}(D^*,\lambda) \geq \mathcal{L}(D^*,\lambda^*) \geq \mathcal{L}(D,\lambda^*) \qquad \forall D\in\Delta_K, \lambda\in\Delta_d. \tag{10.8}$$

Part (a). First, we claim that $Y_i := 1 - \frac{T}{B}\, c_i(D^*) \geq 0$ for each resource i with $\lambda^*_i = 1$.

To prove this claim, assume i is not the time resource (otherwise $c_i(D^*) = B/T$, and we are done). Fix any arm a, and consider the distribution D over arms which assigns probability 0 to arm a, puts probability $D^*(\texttt{null}) + D^*(a)$ on the null arm, and coincides with D^* on all other arms. Using Equation (10.8),

$$\begin{aligned}
0 &\leq \mathcal{L}(D^*,\lambda^*) - \mathcal{L}(D,\lambda^*) \\
&= [r(D^*) - r(D)] - \tfrac{T}{B}\,[c_i(D^*) - c_i(D)] \\
&= D^*(a)\left[r(a) - \tfrac{T}{B}\, c_i(a)\right] \\
&\leq D^*(a)\left[1 - \tfrac{T}{B}\, c_i(a)\right], \quad \forall \in [K].
\end{aligned}$$

Summing over all arms, we obtain $Y_i \geq 0$, claim proved.

Second, we claim that $Y_i \geq 0$ all resources i. For a contradiction, focus on resource i with the smallest $Y_i < 0$; note that $\lambda^*_i < 1$ by the first claim. Consider distribution $\lambda \in \Delta_d$ which puts probability 1 on resource i. Then $\mathcal{L}(D^*,\lambda) < \mathcal{L}(D^*,\lambda^*)$, contradicting (10.8).

Third, assume that $\lambda^*_i > 0$ and $Y_i > 0$ for some resource i. Then $\mathcal{L}(D^*,\lambda^*) > r(D^*)$. Now, consider distribution λ which puts probability 1 on the dummy resource. Then $\mathcal{L}(D^*,\lambda) = r(D^*) < \mathcal{L}(D^*,\lambda^*)$, contradicting (10.8). Thus, $\lambda^*_i = 0$ implies $Y_i > 0$.

Part (bc). By part (a), D^* is a feasible solution to (10.5), and $\mathcal{L}(D^*,\lambda^*) = r(D^*)$. Let D be some other feasible solution for (10.5). Plugging in the feasibility constraints for D, we have $\mathcal{L}(D,\lambda^*) \geq r(D)$. Then

$$r(D^*) = \mathcal{L}(D^*,\lambda^*) \geq \mathcal{L}(D,\lambda^*) \geq r(D).$$

So, D^* is an optimal solution to the $\texttt{LP}$. In particular, $\texttt{OPT}_{\texttt{LP}} = r(D^*) = \mathcal{L}(D^*,\lambda^*)$. □

Step 3: Repeated Lagrange game

The round-t outcome matrix $\mathbf{M}_t$, as defined in Remark 10.4, defines the respective Lagrange function $\mathcal{L}_t$:

$$\mathcal{L}_t(a,i) = r_t(a) + 1 - \tfrac{T}{B}\, c_{t,i}(a), \quad a \in [K],\ i \in [d]. \tag{10.9}$$

Note that $\mathbb{E}[\mathcal{L}_t(a,i)] = \mathcal{L}(a,i)$, so we will refer to $\mathcal{L}$ as the *expected* Lagrange function.

Remark 10.10. The function defined in (10.9) is a Lagrange function for the appropriate "round-t version" of the linear program (10.5). Indeed, consider the expected outcome matrix $\mathbf{M}_{\texttt{exp}} := \mathbb{E}[\mathbf{M}_t]$, which captures the expected per-round rewards and consumptions,[2] and therefore defines the LP. Let us instead plug in an *arbitrary* outcome matrix $\mathbf{M} \in [0,1]^{K\times(d+1)}$ instead of $\mathbf{M}_{\texttt{exp}}$. Formally, let $\texttt{LP}_{\mathbf{M}}$ be a version of (10.5) when $\mathbf{M}_{\texttt{exp}} = \mathbf{M}$, and let $\mathcal{L}_{\mathbf{M}}$ be the corresponding Lagrange function. Then $\mathcal{L}_t = \mathcal{L}_{\mathbf{M}_t}$ for each round t, *i.e.,* $\mathcal{L}_t$ is the Lagrange function for the version of the LP induced by the round-t outcome matrix $\mathbf{M}_t$.

The *repeated Lagrange game* is a game between two algorithms, the *primal algorithm* $\texttt{ALG}_1$ and the *dual algorithm* $\texttt{ALG}_2$, which proceeds over T rounds. In each round t, the primal algorithm chooses arm a_t, the dual algorithm chooses a resource i_t, and the payoff — primal player's reward and dual player's cost — equals $\mathcal{L}_t(a_t, i_t)$. The two algorithms make their choices simultaneously, without observing one another.

Our algorithm, called `LagrangeBwK`, is very simple: it is a repeated Lagrangian game in which the primal algorithm receives bandit feedback, and the dual algorithm receives full feedback. The pseudocode, summarized in Algorithm 10.1, is self-contained: it specifies the algorithm even without defining repeated games and Lagrangian functions. The algorithm is *implementable*, in the sense that the outcome vector $\vec{o}_t$ revealed in each round t of the `BwK` problem suffices to generate the feedback for both $\texttt{ALG}_1$ and $\texttt{ALG}_2$.

```
Given      : time horizon $T$, budget $B$, number of arms $K$, number of resources $d$.
             Bandit algorithm $\texttt{ALG}_1$: action set $[K]$, maximizes rewards, bandit feedback.
             Bandit algorithm $\texttt{ALG}_2$: action set $[d]$, minimizes costs, full feedback.
for round $t = 1, 2, \dots$ (until stopping) do
    $\texttt{ALG}_1$ returns arm $a_t \in [K]$, algorithm $\texttt{ALG}_2$ returns resource $i_t \in [d]$.
    Arm $a_t$ is chosen, outcome vector $\vec{o}_t = (r_t(a_t); c_{t,1}(a_t), \dots, c_{t,d}(a_t)) \in [0,1]^{d+1}$ is observed.
    The payoff $\mathcal{L}_t(a_t, i_t)$ from (10.9) is reported to $\texttt{ALG}_1$ as reward, and to $\texttt{ALG}_2$ as cost.
    The payoff $\mathcal{L}_t(a_t, i)$ is reported to $\texttt{ALG}_2$ for each resource $i \in [d]$.
end
```

Algorithm 10.1: Algorithm `LagrangeBwK`

Let us apply the machinery from Section 9 to the repeated Lagrangian game. For each algorithm $\texttt{ALG}_j$, $j \in \{1,2\}$, we are interested in its hindsight regret $R_j(T)$ relative to the best observed action, as defined in Chapter 5.1. We will call it *adversarial regret* throughout this chapter, to distinguish from the regret in `BwK`. For each round $\tau \in [T]$, let $\bar{a}_\tau = \frac{1}{\tau}\sum_{t\in[\tau]} a_t$ and $\bar{\imath}_\tau = \frac{1}{\tau}\sum_{t\in[\tau]} i_t$ be the average play distribution of $\texttt{ALG}_1$ and $\texttt{ALG}_2$, respectively. Now, Exercise 9.4, applied for the time horizon $\tau \in [T]$, implies the following:

Lemma 10.11. *For each round $\tau \in [T]$, the average play $(\bar{a}_\tau, \bar{\imath}_\tau)$ forms a δ_τ-approximate Nash equilibrium for the expected Lagrange game defined by $\mathcal{L}$, where*

$$\tau \cdot \delta_\tau = R_1(\tau) + R_2(\tau) + \texttt{err}_\tau, \textit{ with error term } \texttt{err}_\tau := \left| \textstyle\sum_{t\in[\tau]} \mathcal{L}_t(i_t, j_t) - \mathcal{L}(i_t, j_i) \right|.$$

Corollary 10.12. $\mathcal{L}(\bar{a}_\tau, i) \geq \texttt{OPT}_{\texttt{LP}} - \delta_\tau$ *for each resource i.*

[2]The a-th row of $\mathbf{M}_{\texttt{exp}}$ is $(r(a);\ c_1(a), \dots, c_d(a))$, for each arm $a \in [K]$.

Step 4: Reward at the stopping time

We focus on the *stopping time* τ, the first round when the total consumption of some resource i exceeds its budget; call i the *stopping resource*. We argue that $\texttt{REW}$ is large compared to $\mathcal{L}(\bar{a}_\tau, i)$ (which plugs nicely into Corollary 10.12). In fact, this step holds for any $\texttt{BwK}$ algorithm.

Lemma 10.13. *For an arbitrary* $\texttt{BwK}$ *algorithm* $\texttt{ALG}$,

$$\texttt{REW}(\texttt{ALG}) \geq \tau \cdot \mathcal{L}(\bar{a}_\tau, i) + (T - \tau) \cdot \texttt{OPT}_{\texttt{LP}} - \texttt{err}^*_{\tau,i},$$

where $\texttt{err}^*_{\tau,i} := \left|\tau \cdot r(\bar{a}_\tau) - \sum_{t\in[\tau]} r_t\right| + \frac{T}{B}\left|\tau \cdot c_i(\bar{a}_\tau) - \sum_{t\in[\tau]} c_{i,t}\right|$ *is the error term.*

Proof. Note that $\sum_{t\in[\tau]} c_{t,i} > B$ because of the stopping. Then:

$$\begin{aligned}\tau \cdot \mathcal{L}(\bar{a}_\tau, i) &= \tau \cdot (r(\bar{a}) + \tau - c_i(\bar{a})) \\ &\leq \sum_{t\in[\tau]} r_t + \tau - \tfrac{T}{B}\sum_{t\in[\tau]} c_{i,t} + \texttt{err}^*_{\tau,i} \\ &\leq \texttt{REW} + \tau - T + \texttt{err}^*_{\tau,i}.\end{aligned}$$

The Lemma follows since $\texttt{OPT}_{\texttt{LP}} \leq 1$. □

Plugging in Corollary 10.12, the analysis is summarized as follows:

$$\texttt{REW}(\texttt{LagrangeBwK}) \geq T \cdot \texttt{OPT}_{\texttt{LP}} - \left[\, R_1(\tau) + R_2(\tau) + \texttt{err}_\tau + \texttt{err}^*_{\tau,i}\,\right], \tag{10.10}$$

where τ is the stopping time and i is the stopping resource.

Wrapping up

It remains to bound the error/regret terms in (10.10). Since the payoffs in the Lagrange game lie in the range $[a, b] := [1 - \frac{T}{B}, 2]$, all error/regret terms are scaled up by the factor $b - a = 1 + \frac{T}{B}$.

Fix an arbitrary failure probability $\delta > 0$. An easy application of Azuma-Hoeffding Bounds implies that

$$\texttt{err}_\tau + \texttt{err}_{\tau,i} \leq O(b-a) \cdot \sqrt{TK \log\left(\tfrac{dT}{\delta}\right)} \qquad \forall \tau \in [T],\, i \in [d],$$

with probability at least $1 - \delta$. (Apply Azuma-Hoeffding to each $\texttt{err}_\tau$ and $\texttt{err}_{\tau,i}$ separately.)

Let use use algorithms $\texttt{ALG}_1$ and $\texttt{ALG}_2$ that admit high-probability upper bounds on adversarial regret. For $\texttt{ALG}_1$, we use algorithm $\texttt{EXP3.P.1}$ from Auer et al. (2002b), and for $\texttt{ALG}_2$ we use a version of algorithm $\texttt{Hedge}$ from Chapter 5.3. With these algorithms, with probability at least $1-\delta$ it holds that, for each $\tau \in [T]$,

$$\begin{aligned}R_1(\tau) &\leq O(b-a) \cdot \sqrt{TK \log\left(\tfrac{T}{\delta}\right)}, \\ R_2(\tau) &\leq O(b-a) \cdot \sqrt{T \log\left(\tfrac{dT}{\delta}\right)}.\end{aligned}$$

Plugging this into (10.10), we obtain the main result for $\texttt{LagrangeBwK}$.

Theorem 10.14. *Suppose algorithm* $\texttt{LagrangeBwK}$ *is used with* $\texttt{EXP3.P.1}$ *as* $\texttt{ALG}_1$*, and* $\texttt{Hedge}$ *as* $\texttt{ALG}_2$*. Then the following regret bound is achieved, with probability at least* $1 - \delta$*:*

$$\texttt{OPT} - \texttt{REW} \leq O\left(\tfrac{T}{B}\right) \cdot \sqrt{TK \ln\left(\tfrac{dT}{\delta}\right)}.$$

This regret bound is optimal in the worst case, up to logarithmic factors, in the regime when $B = \Omega(T)$. This is because of the $\Omega(\sqrt{KT})$ lower bound for the "standard" bandit problem with IID rewards.

10.3 Optimal algorithms and regret bounds (no proofs)

The regret bound in Theorem 10.14 is not the best possible for $\texttt{BwK}$. The optimal regret bound is as follows:

$$\texttt{OPT} - \mathbb{E}[\texttt{REW}] \leq \tilde{O}\left(\sqrt{K \cdot \texttt{OPT}} + \texttt{OPT}\sqrt{K/B}\right). \tag{10.11}$$

(As in Section 10.2, we assume that all budgets are B by rescaling.) The first summand is essentially regret from bandits with IID rewards, and the second summand is due to the global constraints. This regret bound is optimal, in the worst case, for any given triple (K, B, T). More formally, there is a lower bound which states that for any algorithm and any given triple (K, B, T), there exists a problem instance of $\texttt{BwK}$ with K arms, budget B, and time horizon T such that this algorithm suffers regret

$$\texttt{OPT} - \mathbb{E}[\texttt{REW}] \geq \Omega\left(\min\left(\texttt{OPT},\ \sqrt{K \cdot \texttt{OPT}} + \texttt{OPT}\sqrt{K/B}\right)\right). \tag{10.12}$$

However, the lower bound is proved for a particular family of problem instances (designed for the purpose of proving this lower bound), which does not rule out better regret bounds for some interesting special cases.

Below we outline two algorithms that achieve the optimal regret bound in (10.11).[3] These algorithms build on techniques from IID bandits: resp., Successive Elimination and Optimism under Uncertainty (see Chapter 1). We omit their analyses, which are very detailed and not as lucid as the one for $\texttt{LagrangeBwK}$.

Prepwork

We consider outcome matrices: formally, these are matrices in $[0,1]^{K\times(d+1)}$. The round-t outcome matrix $\mathbf{M}_t$ is defined as per Remark 10.4, and the expected outcome matrix is $\mathbf{M}_{\texttt{exp}} = \mathbb{E}[\mathbf{M}_t]$.

Both algorithms maintain a *confidence region* on $\mathbf{M}_{\texttt{exp}}$: a set of outcome matrices which contains $\mathbf{M}_{\texttt{exp}}$ with probability at least $1 - T^{-2}$. In each round t, the confidence region $\texttt{ConfRegion}_t$ is recomputed based on the data available in this round. More specifically, a confidence interval is computed separately for each "entry" of $\mathbf{M}_{\texttt{exp}}$, and $\texttt{ConfRegion}_t$ is defined as a product set of these confidence intervals.

Given a distribution D over arms, consider its value in the linear program (10.5):

$$\texttt{LP}(D \mid B, \mathbf{M}_{\texttt{exp}}) = \begin{cases} r(D) & \text{if } c_i(D) \leq B/T \text{ for each resource } i, \\ 0 & \text{otherwise.} \end{cases}$$

We use a flexible notation which allows to plug in arbitrary outcome matrix $\mathbf{M}_{\texttt{exp}}$ and budget B.

Algorithm I: Successive Elimination with Knapsacks

We look for optimal *distributions* over arms. A distribution D is called *potentially optimal* if it optimizes $\texttt{LP}(D \mid B, \mathbf{M})$ for some $\mathbf{M} \in \texttt{ConfRegion}_t$. 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. We settle for something almost as good: we choose an arm a uniformly at random, and then explore it as much as possible, in the sense that we choose a potentially optimal distribution D that maximizes $D(a)$. See Algorithm 10.2 for the pseudocode.

[3]More precisely, Algorithm 10.2 achieves the regret bound (10.11) with an extra multiplicative term of $\sqrt{d}$, whereas the optimal dependence on d is logarithmic.

```
for round $t = 1, 2, \ldots$ (until stopping) do
    $S_t \leftarrow$ the set of all potentially optimal distributions over arms.
    Pick arm $b_t$ uniformly at random.
    Pick a distribution $D = D_t$ so as to maximize $D(b_t)$ over all $D \in S_t$.
    Pick arm $a_t \sim D_t$.
end
```

Algorithm 10.2: Successive Elimination with Knapsacks

Remark 10.15. The step of maximizing $D(b_t)$ does not have a computationally efficient implementation (more precisely, such implementation is not known for the general case of BwK).

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.

Algorithm II: Optimism under Uncertainty for BwK

For each round t and each distribution D over arms, define the Upper Confidence Bound as

$$\texttt{UCB}_t(D \mid B) = \sup_{\mathbf{M} \in \texttt{ConfRegion}_t} \texttt{LP}(D \mid B, \mathbf{M}). \tag{10.13}$$

The algorithm is very simple: in each round, the algorithm picks distribution D which maximizes the 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. This trick ensures that the algorithm does not run out of resources too soon due to randomness in the outcomes or to the fact that the distributions D_t do not quite achieve the optimal value for $\texttt{LP}(D)$. Thus, the algorithm is as follows:

```
Rescale the budget: $B' \leftarrow B(1-\epsilon)$, where $\epsilon = \tilde{\Theta}(\sqrt{K/B})$
Initialization: pull each arm once.
for all subsequent rounds $t$ do
    In each round $t$, pick distribution $D = D_t$ with highest $\texttt{UCB}_t(\cdot \mid B')$.
    Pick arm $a_t \sim D_t$.
end
```

Algorithm 10.3: Optimism under Uncertainty for BwK.

Remark 10.16. For a given distribution D, the supremum in (10.13) is obtained when upper confidence bounds are used for rewards, and lower confidence bounds are used for resource consumption:

$$\texttt{UCB}_t(D \mid B') = \begin{cases} r^{\texttt{UCB}}(D) & \text{if } c_i^{\texttt{LCB}}(D) \leq B'/T \text{ for each resource } i, \\ 0 & \text{otherwise.} \end{cases}$$

Accordingly, choosing a distribution with maximal UCB can be implemented by a linear program:

$$\begin{array}{ll} \text{maximize} & r^{\texttt{UCB}}(D) \\ \text{subject to} & \\ & D \in \Delta_K \\ & c_i^{\texttt{LCB}}(D) \leq B'/T. \end{array} \tag{10.14}$$

10.4 Bibliographic remarks and further directions

The general setting of BwK is introduced in Badanidiyuru et al. (2018), along with an optimal solution (10.11), the lower bound (10.12), and a detailed discussion of motivational examples. LagrangeBwK, the algorithm this chapter focuses on, is from a subsequent paper (Immorlica et al., 2018).[4] The lower bound is also implicit in an earlier paper (Devanur et al., 2019).

The optimal regret bound in (10.11) has been achieved by three different algorithms. Algorithm 10.2 is from (Badanidiyuru et al., 2018), and Algorithm 10.3 is from a follow-up paper (Agrawal and Devanur, 2014). Both algorithms are modified slightly compared to their original versions, in a way that does not affect their respective analyses. The third algorithm, also from (Badanidiyuru et al., 2018), is a "primal-dual" algorithm superficially similar to LagrangeBwK. It decouples into two online learning algorithms: a "primal" algorithm which chooses among arms, and a "dual" algorithm similar to $\texttt{ALG}_2$, which chooses among resources. However, the two algorithms are not playing a repeated game in any meaningful sense. The primal algorithm is very problem-specific: it interprets the dual distribution as a vector of costs over resources, and chooses arms with largest reward-to-cost ratios, estimated using "optimism under uncertainty".

Some of the key ideas in BwK trace back to earlier work.[5] First, focusing on total rather than per-round expected rewards, approximating the total rewards with a linear program, and using "optimistic" estimates of the LP values in a UCB-style algorithm goes back to Babaioff et al. (2015). They studied the special case of dynamic pricing with limited supply, and applied these ideas to fixed arms (not distributions over arms). Second, repeated Lagrange games, in conjunction with regret minimization in zero-sum games, have been used as an algorithmic tool to solve convex optimization problems (different from BwK), application domains range from differential privacy to algorithmic fairness to learning from revealed preferences (Rogers et al., 2015; Hsu et al., 2016; Roth et al., 2016; Kearns et al., 2018; Agarwal et al., 2017a; Roth et al., 2017). All these papers deal with deterministic games (*i.e.,* same game matrix in all rounds). Most related are (Roth et al., 2016, 2017), where a repeated Lagrangian game is used as a subroutine (the "inner loop") in an online algorithm; the other papers solve an offline problem. Third, estimating an optimal "dual" vector from samples and using this vector to guide subsequent "primal" decisions is a running theme in the work on *stochastic packing* problems (Devanur and Hayes, 2009; Agrawal et al., 2014; Devanur et al., 2019; Feldman et al., 2010; Molinaro and Ravi, 2012). These are full information problems in which the costs and rewards of decisions in the past and present are fully known, and the only uncertainty is about the future. Particularly relevant is the algorithm of Devanur et al. (2011), in which the dual vector is adjusted using multiplicative updates, as in LagrangeBwK and the primal-dual algorithm from Badanidiyuru et al. (2018).

[4] Rivera et al. (2018), in a simultaneous and independent work, put forward a similar algorithm, and analyze it for "bandit convex optimization with knapsacks" (see below) under full feedback.

[5] In addition to the zero-sum games machinery from Chapter 9 and the IID bandit techniques from Chapter 1.

Extensions and special cases

Generalized resources. Agrawal and Devanur (2014) consider an extension to a more abstract version of resources. First, they remove the distinction between rewards and resource consumption. Instead, in each round t there is an outcome vector $o_t \in [0,1]^d$, and the "final outcome" is the average $\bar{o}_T = \frac{1}{T}\sum_{t\in[T]} o_t$. The total reward is determined by $\bar{o}_T$, in a very flexible way: it is $T \cdot f(\bar{o}_T)$, for an arbitrary Lipschitz concave function $f : [0,1]^d \mapsto [0,1]$ known to the algorithm. Second, the resource constraints can be expressed by an arbitrary convex set $S \subset [0,1]^d$ which $\bar{o}_T$ must belong to. Third, they allow *soft constraints*: rather than requiring $\bar{o}_t \in S$ for all rounds t (*hard constraints*), they upper-bound the distance between $\bar{o}_t$ and S. They obtain regret bounds that scale as $\sqrt{T}$, with distance between $\bar{o}_t$ and S scaling as $1/\sqrt{T}$. Their results extend to the hard-constraint version by rescaling the budgets, as in Algorithm 10.3, as long as the constraint set S is downward closed.

Bandits with one knapsack. BwK with only one constrained resource (and unlimited number of rounds) tends to be an easier problem, avoiding much of the complexity of the general case. In patricular, the optimal all-knowing time-invariant policy is the best fixed arm, rather than the best fixed distribution over arms. A line of work György et al. (2007); Tran-Thanh et al. (2010, 2012); Ding et al. (2013) obtains instance-dependent polylog(T) regret bounds under various assumptions.

From bandits to BwK. LagrangeBwK algorithm presented in Section 10.2 provides a general template for extensions. Indeed, it allows for an arbitrary "primal" algorithm $\texttt{ALG}_1$, and turns it into an algorithm for BwK. To obtain an extension to a particular setting, *e.g.,* contextual bandits, $\texttt{ALG}_1$ should be an algorithm for this setting that achieves a high-probability bound on its "adversarial regret" $R_1(\cdot)$ (as defined before Lemma 10.11). This regret bound then plugs into Equation (10.10) in the analysis.

Immorlica et al. (2018) use this approach to obtain algorithms and regret bounds for several scenarios: contextual bandits, combinatorial semi-bandits, bandit convex optimization, and full feedback. (These scenarios are discussed in detail below.) The resulting regret bounds are usually optimal in the regime when $\min(B, \texttt{OPT}) > \Omega(T)$. Importantly, these extensions take little or no extra work.

While successful, this approach comes with several important caveats. First, $\texttt{ALG}_1$ needs a high-probability regret bound against an adaptive adversary, even if for BwK we only need an expected regret bound against an oblivious adversary. Second, this regret bound needs to hold for the rewards specified by the Lagrange functions, rather than the rewards in the underlying BwK problem (and the latter may have some useful properties that do not carry over to the former). Third, the resulting regret bounds tend to be suboptimal when B or OPT are much smaller than T.

Contextual bandits with knapsacks. Contextual bandits with knapsacks (*cBwK*) is a common generalization of BwK and contextual bandits (see Chapter 8). In each round t, an algorithm observes context x_t before it chooses an arm. The pair $(x_t, \mathbf{M}_t)$, where $\mathbf{M}_t \in [0,1]^{K\times(d+1)}$ is the round-t outcome matrix, is chosen independently from some fixed distribution (which is included in the problem instance).

The algorithm is given a set Π of policies (mappings from arms to actions), as in Section 8.4. The benchmark is the best all-knowing algorithm restricted to policies in Π: $\texttt{OPT}_\Pi$ is the expected total reward of such algorithm, similarly to (10.4). The following regret bound can be achieved:

$$\texttt{OPT}_\Pi - \mathbb{E}[\texttt{REW}] \le \tilde{O}(1 + \texttt{OPT}_\Pi/B)\,\sqrt{KT\log|\Pi|}. \tag{10.15}$$

The $\sqrt{KT\log(|\Pi|)}$ dependence on K, T and Π is optimal for contextual bandits, whereas the $(1+\texttt{OPT}_\Pi/B)$ term is due BwK. In particular, this regret bound is optimal in the regime $B > \Omega(\texttt{OPT}_\Pi)$.

Badanidiyuru et al. (2014) achieve (10.15), with an extra factor of $\sqrt{d}$, unifying Algorithm 10.2 and the *policy elimination* algorithm for contextual bandits (Dudík et al., 2011). Their algorithm does not come with a computationally efficient implementation. Subsequently, Agrawal et al. (2016) obtained (10.15) using an "oracle-efficient" algorithm which builds on the contextual bandit algorithm from (Agarwal et al., 2014), see Section 8.4 for background and motivation. Their algorithm uses uses the classification oracle for policy class Π as a subroutine, and calls this oracle only $\tilde{O}(d\sqrt{KT\log|\Pi|})$ times.

LagrangeBwK algorithm can be applied, achieving regret bound

$$\texttt{OPT}_{\Pi} - \mathbb{E}[\texttt{REW}] \leq \tilde{O}(T/B)\,\sqrt{KT\log|\Pi|}.$$

This regret rate matches (10.15) when $\texttt{OPT}_{\Pi} > \Omega(T)$, and is optimal in the regime $B > \Omega(T)$. The "primal" algorithm $\texttt{ALG}_1$ is Exp4.P from Beygelzimer et al. (2011), the high-probability version of algorithm Exp4 from Section 6.4. Like Exp4, this algorithm is not computationally efficient.

In a stark contrast with BwK, non-trivial regret bounds are not necessarily achievable when B and $\texttt{OPT}_{\Pi}$ are small. Indeed, $o(\texttt{OPT})$ worst-case regret is impossible in the regime $\texttt{OPT}_{\Pi} \leq B \leq \sqrt{KT}/2$ (Badanidiyuru et al., 2014). Whereas $o(\texttt{OPT})$ regret bound holds for BwK whenever $B = \omega(1)$, as per (10.11).

Agrawal and Devanur (2016) consider an extension of BwK to *linear* contextual bandits (see Section 8.3), where the expected outcome matrix is linear in the context x_t, and all policies are allowed. Formally, each context is a matrix: $x_t \in [0,1]^{K\times m}$, where rows correspond to arms, and each arm's context has dimension m. The linear assumption states that $\mathbb{E}[\mathbf{M}_t \mid x_t] = x_t\,\mathbf{W}$, for some matrix $\mathbf{W} \in [0,1]^{(d+1)\times m}$ that is fixed over time, but not known to the algorithm. Agrawal and Devanur (2016) achieve regret bound

$$\texttt{OPT} - \mathbb{E}[\texttt{REW}] \leq \tilde{O}(m\sqrt{T})(1+\texttt{OPT}/B) \qquad \text{in the regime } B > mT^{3/4}. \tag{10.16}$$

The $\tilde{O}(m\sqrt{T})$ dependence is the best possible for linear bandits (Dani et al., 2008), whereas the $(1+\texttt{OPT}/B)$ term and the restriction to $B > mT^{3/4}$ is due to BwK. In particular, this regret bound is optimal, up to logarithmic factors, in the regime $B > \max(\Omega(\texttt{OPT}),\, m\,T^{3/4})$.

Combinatorial semi-bandits with knapsacks. In the extension to combinatorial semi-bandits (see Section 7.3), there is a finite set S of *atoms*, and a collection $\mathcal{F}$ of feasible subsets of S. Arms correspond to the subsets in $\mathcal{F}$. When an arm $a = a_t \in \mathcal{F}$ is chosen in some round t, each atom $i \in a$ collects a reward and consumes resources, with its outcome vector $v_{t,i} \in [0,1]^{d+1}$ chosen independently from some fixed but unknown distribution. The "total" outcome vector $\vec{o}_t$ is a sum over atoms: $\vec{o}_t = \sum_{i\in a} v_{t,i}$. We have *semi-bandit feedback*: the outcome vector $v_{t,i}$ is revealed to the algorithm, for each atom $i \in a$. The central theme is that, while the number of arms, $K = |\mathcal{F}|$, may be exponential in the number of atoms, $m = |S|$, one can achieve regret bounds that are polynomial in m.

Combinatorial semi-bandits with knapsacks is a special case of linear cBwK, as defined above, where the context x_t is the same in all rounds, and defines the collection $\mathcal{F}$. (For each arm $a \in \mathcal{F}$, the a-th row of x_t is a binary vector that represents a.) Thus, the regret bound (10.16) for linear cBwK applies. Sankararaman and Slivkins (2018) achieve an improved regret bound when the set system $\mathcal{F}$ is a matroid. They combine Algorithm 10.3 with the *randomized rounding* techniques from approximation algorithms. In the regime when $\min(B,\texttt{OPT}) > \Omega(T)$, these two regret bounds become, respectively, $\tilde{O}(m\sqrt{T})$ and $\tilde{O}(\sqrt{mT})$. LagrangeBwK algorithm (with a suitable "primal" algorithm $\texttt{ALG}_1$) matches the latter regret bound for this regime, and achieves $\tilde{O}(T/B)\sqrt{mT}$ regret in general, without making the matroid assumption. The $\sqrt{mT}$ regret is optimal, up to constant factors, even for combinatorial semi-bandits "without knapsacks" (Kveton et al., 2014).

Bandit convex optimization (BCO) with knapsacks. BCO is a version of multi-armed bandits where the action set is a convex set $\mathcal{X} \subset \mathbb{R}^K$, and in each round t, there is a concave function $f_t : \mathcal{X} \to [0,1]$ such that the reward for choosing action $\vec{x} \in \mathcal{X}$ in this round is $f_t(\vec{x})$. BCO is a prominent topic in bandits, starting from (Kleinberg, 2004; Flaxman et al., 2005), with important recent advances (Bubeck et al., 2015; Hazan and Levy, 2014; Bubeck et al., 2017).

We are interested in a common generalization of BCO and BwK, called *BCO with knapsacks*. For each round t and each resource i, one has convex functions $g_{t,i} : \mathcal{X} \to [0,1]$, such that the consumption of this resource for choosing action $\vec{x} \in \mathcal{X}$ in this round is $g_{t,i}(\vec{x})$. The tuple of functions $(f_t; g_{t,1}\,,\; \dots\;, g_{t,d})$ is sampled independently from some fixed distribution (which is not known to the algorithm). Immorlica et al. (2018) provide a regret bound of the form $\frac{T}{B}\sqrt{T} \cdot \text{poly}(K \log T)$, building on the recent breakthrough in BCO in Bubeck et al. (2017).[6] The algorithm is LagrangeBwK, where the "primal" algorithm $\texttt{ALG}_1$ is the BCO algorithm from Bubeck et al. (2017). Rivera et al. (2018) obtain a similar regret bound for the full feedback version, in a simultaneous and independent work.

BwK with full feedback. In the full-feedback version of BwK, the entire outcome matrix $\mathbf{M}_t$ is revealed after each round t. The following regret bound can be achieved:

$$\texttt{OPT} - \mathbb{E}[\texttt{REW}] \leq O\left(\sqrt{\log(dKT)}\left(\sqrt{\texttt{OPT}} + \texttt{OPT}/\sqrt{B}\right)\right). \tag{10.17}$$

Essentially, this is the regret bound (10.11) for BwK, where K is replaced with $\log K$. This regret bound can be obtained, *e.g.,* Algorithm 10.3 with a slightly modified analysis. LagrangeBwK achieves regret $O(T/B) \cdot \sqrt{T \log(dKT)}$, matching (10.17) when $\min(B, \texttt{OPT}) > \Omega(T)$. In this application, the "primal" algorithm $\texttt{ALG}_1$ is Hedge from Section 5.3.

Adversarial bandits with knapsacks

In the adversarial version of BwK, each outcome matrices $\mathbf{M}_t$ is chosen by an adversary. Let us focus on the oblivious adversary, so that the entire sequence $\mathbf{M}_1, \mathbf{M}_2\,,\; \dots\;, \mathbf{M}_T$ is fixed before round 1. All results below are from Immorlica et al. (2018), unless mentioned otherwise. The version with IID outcomes that we've considered before will be called *Stochastic BwK.*

Hardness of the problem. Adversarial BwK is a much harder problem compared to the stochastic version. The new challenge is that the algorithm needs to decide how much budget to save for the future, without being able to predict it. An algorithm must compete, during any given time segment $[1, \tau]$, with a distribution D_τ over arms that maximizes the total reward on this time segment. However, these distributions may be very different for different τ. For example, one distribution D_τ may exhaust some resources by time τ, whereas another distribution $D_{\tau'}$, $\tau' > \tau$ may save some resources for later.

Due to this hardness, one can only approximate optimal rewards up to a multiplicative factor, whereas sublinear regret is no longer possible. To state this point more concretely, consider the ratio $\texttt{OPT}_{\texttt{FD}} / \mathbb{E}[\texttt{REW}]$, called *competitive ratio*, where $\texttt{OPT}_{\texttt{FD}}$ is the expected total reward of the best fixed distribution over arms. A very strong lower bound holds: no algorithm can achieve competitive ratio better than $O(\log T)$ on all problem instances. The lower-bounding construction involves only two arms and only one resource, and forces the algorithm to make a huge commitment without knowing the future.

It is instructive to consider a simple example in which the competitive ratio is at least $\frac{5}{4} - o(1)$ for any algorithm. There are two arms and one resource with budget $\frac{T}{2}$. Arm 1 has zero rewards and zero consumption. Arm 2 has consumption 1 in each round, and offers reward $\frac{1}{2}$ in each round of the first half-time ($\frac{T}{2}$ rounds). In the second half-time, it offers either reward 1 in all rounds, or reward 0 in all rounds. Thus, there are two problem instances that coincide for the first half-time and differ in the second half-time.

[6]The regret bound is simply $O(\frac{T}{B})$ times the state-of-art regret bound from Bubeck et al. (2017).

The algorithm needs to choose how much budget to invest in the first half-time, without knowing what comes in the second. Any choice leads to competitive ratio at least $\frac{5}{4}$ on one of the two instances.

The best fixed distribution is arguably the right benchmark for this problem. The best all-knowing algorithm benchmark is *too harsh*: there are simple examples when $\texttt{OPT}/\,\mathbb{E}[\texttt{REW}] \geq T/B^2$ for some problem instance, for any given B, T. And the best fixed arm benchmark, denote it $\texttt{OPT}_{\texttt{FA}}$, is *uninteresting*: there is a lower bound $\texttt{OPT}_{\texttt{FA}}/\,\mathbb{E}[\texttt{REW}] \geq \Omega(K)$, matched by a trivial algorithm that chooses one arm uniformly at random and plays it forever.

Algorithmic results. One can achieve a near-optimal competitive ratio,

$$\frac{\texttt{OPT}_{\texttt{FD}} - \texttt{reg}}{\mathbb{E}[\texttt{REW}]} \leq (d+1)^2 \lceil \log T \rceil, \tag{10.18}$$

up to a regret term $\texttt{reg} = O(1 + \frac{\texttt{OPT}_{\texttt{FD}}}{dB})\sqrt{TK\log(Td)}$. Surprisingly, this result is achieved using an algorithm for Stochastic BwK. Namely, the algorithm is a version of `LagrangeBwK`, with two important differences: the time resource is not included in the outcome matrices, and the $\frac{T}{B}$ ratio in the Lagrange function (10.9) is replaced by a parameter γ, sampled at random from some exponential scale. Essentially, when γ is sampled close to $\texttt{OPT}_{\texttt{FD}}/B$, the algorithm obtains a constant competitive ratio. A completely new analysis of `LagrangeBwK` is needed, as the zero-games framework from Chapter 9 is no longer applicable.

One can also achieve a similar competitive ratio with high probability:

$$\Pr\left[\frac{\texttt{OPT}_{\texttt{FD}} - \texttt{reg}}{\texttt{REW}} \leq 2(d+1)^4 \lceil \log T \rceil\right] \geq 1 - T^{-2},$$

with a somewhat larger regret term `reg`. This result uses `LagrangeBwK` as a subroutine, and its analysis as a key lemma. The algorithm is considerably more involved: instead of guessing the parameter γ upfront, the guess is iteratively refined over time.

As discussed previously, `LagrangeBwK` provides a general template for extensions. We immediately obtain extensions to the same settings as before: contextual bandits with knapsacks, combinatorial semi-bandits with knapsacks, bandit convex optimization with knapsacks, and BwK with full feedback. For each setting, one obtains the competitive ratio stated above, with some problem-specific regret term `reg`.

Bandits with one knapsack. Rangi et al. (2018) consider the special case when there is only one constrained resource, including time. They attain sublinear regret, *i.e.,* a regret bound that is sublinear in T. They assume a known lower bound $c_{\min} > 0$ on realized per-round consumption of each resource, and their regret bound scales as $1/c_{\min}$. They also achieve polylog(T) instance-dependent regret for the stochastic version using the same algorithm (matching results from prior work on the stochastic version).

OCO with constraints. A related line of work concerns online convex optimization with constraints (Mahdavi et al., 2012, 2013; Chen et al., 2017; Neely and Yu, 2017; Chen and Giannakis, 2018). The setting is similar to Adversarial BwK, but differs in several important respects. First, the action set is a convex subset of $\mathbb{R}^K$ (and the algorithms rely on the power to choose arbitrary actions in this set). In particular, there is no immediate way to handle discrete action sets under bandit feedback. Second, convexity/concavity is assumed on the rewards and resource consumption. Third, more feedback is assumed: full feedback is observed for the resource consumption, and (except in Chen and Giannakis (2018)) one also observes either full feedback on rewards or the rewards gradient around the chosen action. Fourth, an algorithm only needs to satisfy the supply/budget constraints at the time horizon (whereas in BwK the constraints hold for all rounds). Fifth, the fixed-distribution benchmark is weaker: essentially, its time-averaged consumption must be small enough at each round t. Due to these differences, this setting admits sublinear regret.

Dynamic pricing with limited supply

A paradigmatic application of BwK is dynamic pricing with limited supply. In the basic version, the algorithm is a seller with B identical copies of some product. In each round t, the algorithm chooses some price $p_t \in [0,1]$ and offers one copy for sale at this price. The outcome is either a sale or no sale; the corresponding outcome vectors are shown in Equation (10.1). The customer response is summarized by the probability of making a sale at a given price p, denoted $S(p)$, which is assumed to be the same in all rounds, and non-increasing in p.[7] The function $S(\cdot)$ is the *demand curve*, a well-known notion in Economics.

This problem can be solved via *discretization*: choose a finite subset $P \subset [0,1]$ of prices, and run a generic BwK algorithm with action set P. The generic guarantees for BwK provide a regret bound against $\texttt{OPT}(P)$, the expected total reward of the best all-knowing algorithm restricted to prices in P. One needs to choose P so as to balance the BwK regret bound (which scales with $\sqrt{|P|}$) and the *discretization error* $\texttt{OPT} - \texttt{OPT}(P)$ (or a similar difference in the LP-values, whichever is more convenient). Bounding the discretization error is a new, non-trivial step, separate from the analysis of the BwK algorithm. With this approach, Badanidiyuru et al. (2018) achieve regret bound $\tilde{O}(B^{2/3})$ against OPT. Note that, up to logarithmic factors, this regret bound is driven by B rather than T. This regret rate is optimal for dynamic pricing, for any given B, T: a complimentary lower bound has been proved in Babaioff et al. (2015). Interestingly, the optimal regret rate is attained using a generic BwK algorithm.

Consider the same problem against a weaker benchmark of the best fixed price. The $\tilde{O}(B^{2/3})$ regret rate is still optimal, as the $\Omega(B^{2/3})$ lower bound from holds against the best fixed arm. However, $\tilde{O}(\sqrt{B})$ regret rate can ba achieved under *regular demands*, a standard assumption in theoretical economics which states that $p \cdot S(p)$, the expected revenue at price p, is a concave function of p. This regret rate is optimal under regular demands, for any given B, T pair, due to a complimentary lower bound. All there results are from Babaioff et al. (2015). The $\tilde{O}(\sqrt{T})$ result under regular demands, for the special case $B > \Omega(T)$, has also appeared in a simultaneous and independent paper (Wang et al., 2014).

The study of dynamic pricing with limited supply has been initiated in Besbes and Zeevi (2009). They considered the case of $B > \Omega(T)$, and achieved regret $\tilde{O}(T^{3/4})$ against the best fixed price, and $\tilde{O}(T^{2/3})$ regret under regular demands. They used a non-adaptive exploration algorithm based on Explore-first technique (see Chapter 1). The unconstrained case ($B = T$), has been introduced in Kleinberg and Leighton (2003), building on the earlier work (Blum et al., 2003). They provide optimal solutions, both for the worst case and for regular demands, along with matching lower bounds. In fact, the lower bounds in Babaioff et al. (2015) are proved by reduction to the ones in Kleinberg and Leighton (2003), which do much of the heavy lifting. The algorithm in Kleinberg and Leighton (2003) fairly straightforward, see Exercise 6.2.

Dynamic pricing with $d \geq 2$ products in the inventory is less understood. As in the single-product case, one can use discretization and run an optimal BwK algorithm on a pre-selected finite subset P of price vectors. If the demand curve is Lipschitz, one can bound the discretization error, and achieve regret rate on the order of $T^{(d+1)/(d+2)}$ (see Exercise 10.3(a)). However, it is unclear how to bound the discretization error without Lipschitz assumptions. The only known result in this direction is when there are multiple feasible bundles of goods, and in each round the algorithm chooses a bundle and a price. Then the technique from the single-product case applies, and one obtains a regret bound of $\tilde{O}(d\, B^{2/3}\, (N\ell)^{1/3})$, where N is the number of bundles, each bundle consists of at most ℓ items, and prices are in the range $[0, \ell]$ (Badanidiyuru et al., 2018). The problem with $d \geq 2$ is pioneered in (Besbes and Zeevi, 2012). They provided a non-adaptive

[7]In particular, suppose one customer arrives at time t, with private value v_t, and buys if and only if $v_t \geq p_t$. If v_t is drawn independently from some distribution D, then $S(p) = \Pr_{v \sim D}[v \geq p]$. Considering the sales probability directly is more general, *e.g.,* it allows for multiple customers to be present at a given round.

exploration algorithm for the regime when all budgets are $\Omega(T)$ and the demands are Lipschitz, attaining regret bound $\tilde{O}(T^{1-1/(d+3)})$.

While this discussion focuses on regret-minimizing formulations of dynamic pricing, Bayesian and parametric formulations versions have a rich literature in Operations Research and Economics (Boer, 2015).

10.5 Exercises and Hints

(Assume Stochastic BwK unless specified otherwise.)

Exercise 10.1 (Explore-first algorithm for BwK). Fix time horizon T and budget B. Consider an algorithm ALG which explores uniformly at random for the first N steps, where N is fixed in advance, then chooses some distribution D over arms and draws independently from this distribution in each subsequent rounds.

(a) Assume $B < \sqrt{T}$. Prove that there exists a problem instance on which ALG suffers linear regret:

$$\texttt{OPT} - \mathbb{E}[\texttt{REW}] > \Omega(T).$$

Hint: Posit one resource other than time, and three arms:

- the *bad arm*, with deterministic reward 0 and consumption 1;
- the *good arm*, with deterministic reward 1 and expected consumption $\frac{B}{T}$;
- the *decoy arm*, with deterministic reward 1 and expected consumption $2\frac{B}{T}$.

Use the following fact: given two coins with expectations $\frac{B}{T}$ and $\frac{B}{T} + c/\sqrt{N}$, for a sufficiently low absolute constant c, after only N tosses of each coin, for any algorithm it is a constant-probability event that this algorithm cannot tell one coin from another.

(b) Assume $B > \Omega(T)$. Choose N and D so that ALG achieves regret $\texttt{OPT} - \mathbb{E}[\texttt{REW}] < \tilde{O}(T^{2/3})$.

Hint: Choose D as a solution to the "optimistic" linear program (10.14), with rescaled budget $B' = B(1 - \sqrt{K/T})$. Compare D to the value of the original linear program (10.5) with budget B', and the latter to the value of (10.5) with budget B.

Exercise 10.2 (Best distribution vs. best fixed arm). Let $\texttt{OPT}_{\texttt{FA}}$ and $\texttt{OPT}_{\texttt{FD}}$ be, resp., the total expected reward of the best fixed arm and that of the best fixed distribution. Let d be the number of resources, including time.

(a) Construct an example such that $\texttt{OPT}_{\texttt{FD}} \geq d \cdot \texttt{OPT}_{\texttt{FA}} - o(\texttt{OPT}_{\texttt{FA}})$.

Hint: Extend the $d = 2$ example from Section 10.1.

(b) Prove that $\texttt{OPT} \leq d \cdot \texttt{OPT}_{\texttt{FA}}$.

Hint: (10.5) has an optimal solution with support size at most d. Use the pigeonhole principle!

Exercise 10.3. (Discretization in dynamic pricing) Consider dynamic pricing with limited supply of d products: actions are price vectors $p \in [0,1]^d$, and $c_i(p) \in [0,1]$ is the expected per-round amount of product i sold at price vector p. Let $P_\epsilon := [0,1]^d \cap \epsilon\, \mathbb{N}^d$ be a uniform mesh of prices with step $\epsilon \in (0,1)$. Let $\texttt{OPT}(P_\epsilon)$ and $\texttt{OPT}_{\texttt{FA}}(P_\epsilon)$ be the resp. benchmark restricted to the price vectors in P_ϵ.

(a) Assume that $c_i(p)$ is Lipschitz in p, for each product i:

$$|c_i(p) - c_i(p')| \leq L \cdot \|p - p\|_1, \quad \forall p, p' \in [0,1]^d.$$

Prove that the discretization error is $\texttt{OPT} - \texttt{OPT}(P_\epsilon) \leq O(\epsilon d L)$. Using an optimal $\texttt{BwK}$ algorithm with appropriately action set P_ϵ, obtain regret rate $\texttt{OPT} - \mathbb{E}[\texttt{REW}] \sim T^{(d+1)/(d+2)}$.

Hint: To bound the discretization error, use the approach from Exercise 10.1; now the deviations in rewards/consumptions are due to the change in p.

(b) For the single-product case ($d = 1$), consider the fixed-arm benchmark, and prove that the resp. discretization error is $\texttt{OPT}_{\texttt{FA}} - \texttt{OPT}_{\texttt{FA}}(P_\epsilon) \leq O(\epsilon \sqrt{B})$.

Hint: Consider "approximate total reward" at price p as $V(p) = p \cdot \min(B,\, T \cdot S(p))$. Prove that the expected total reward for always using price p lies between $V(p) - \tilde{O}(\sqrt{B})$ and $V(p)$.

Exercise 10.4. ($\texttt{LagrangeBwK}$ for adversarial bandits) Consider adversarial bandits, as a special case of Adversarial BwK with $d \geq 2$ resources and zero resource consumption. Prove that $\texttt{LagrangeBwK}$ achieves regret $\tilde{O}(\sqrt{KT})$.

Hint: One can no longer rely on the machinery from Chapter 9. Use the regret bound for the primal algorithm, and the fact that there exists an optimal solution for (10.5) with support size 1.

Bibliography

Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *25th Advances in Neural Information Processing Systems (NIPS)*, pages 2312–2320, 2011.

Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In *17th ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 49–55, 2005.

Alekh Agarwal, Miroslav Dudík, Satyen Kale, John Langford, and Robert E. Schapire. Contextual bandit learning with predictable rewards. In *15th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 19–26, 2012.

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, Luong Hoang, John Langford, Lihong Li, Dan Melamed, Gal Oshri, Siddhartha Sen, and Aleksandrs 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`.

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. *Fairness, Accountability, and Transparency in Machine Learning (FATML)*, 2017a.

Alekh Agarwal, Sarah Bird, Markus Cozowicz, Luong Hoang, John Langford, Stephen Lee, Jiaji Li, Dan Melamed, Gal Oshri, Oswaldo Ribas, Siddhartha Sen, and Alex Slivkins. Making contextual decisions with low technical debt, 2017b. Techical report at `arxiv.org/abs/1606.03966`.

Rajeev Agrawal. The continuum-armed bandit problem. *SIAM J. Control and Optimization*, 33(6):1926–1951, 1995.

Shipra Agrawal and Nikhil R. Devanur. Bandits with concave rewards and convex knapsacks. In *15th ACM Conf. on Economics and Computation (ACM EC)*, 2014.

Shipra Agrawal and Nikhil R. Devanur. Linear contextual bandits with knapsacks. In *29th Advances in Neural Information Processing Systems (NIPS)*, 2016.

Shipra Agrawal and Navin Goyal. Analysis of Thompson Sampling for the multi-armed bandit problem. In *25nd Conf. on Learning Theory (COLT)*, 2012.

Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. In *16th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 99–107, 2013.

Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014.

Shipra Agrawal, Nikhil R. Devanur, and Lihong Li. An efficient algorithm for contextual bandits with knapsacks, and an extension to concave objectives. In *29th Conf. on Learning Theory (COLT)*, 2016.

Kareem Amin, Michael Kearns, and Umar Syed. Bandits, query learning, and the haystack dimension. In *24th Conf. on Learning Theory (COLT)*, 2011.

J.-Y. Audibert, R. Munos, and Cs. Szepesvári. Exploration-exploitation trade-off using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410:1876–1902, 2009.

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 and Chao-Kai Chiang. An algorithm with nearly optimal pseudo-regret for both stochastic and adversarial bandits. In *29th Conf. on Learning Theory (COLT)*, 2016.

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.

Peter Auer, Ronald Ortner, and Csaba Szepesvári. Improved Rates for the Stochastic Continuum-Armed Bandit Problem. In *20th Conf. on Learning Theory (COLT)*, pages 454–468, 2007.

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.

Mohammad Gheshlaghi Azar, Alessandro Lazaric, and Emma Brunskill. Online stochastic optimization under correlated bandit feedback. In *31th Intl. Conf. on Machine Learning (ICML)*, pages 1557–1565, 2014.

Moshe Babaioff, Yogeshwer Sharma, and Aleksandrs Slivkins. Characterizing truthful multi-armed bandit mechanisms. *SIAM J. on Computing (SICOMP)*, 43(1):194–230, 2014. Preliminary version in *10th ACM EC*, 2009.

Moshe Babaioff, Shaddin Dughmi, Robert D. Kleinberg, and Aleksandrs Slivkins. Dynamic pricing with limited supply. *ACM Trans. on Economics and Computation*, 3(1):4, 2015. Special issue for *13th ACM EC*, 2012.

Ashwinkumar Badanidiyuru, John Langford, and Aleksandrs Slivkins. Resourceful contextual bandits. In *27th Conf. on Learning Theory (COLT)*, 2014.

Ashwinkumar Badanidiyuru, Robert Kleinberg, and Aleksandrs Slivkins. Bandits with knapsacks. *J. of the ACM*, 65 (3), 2018. Preliminary version in *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.

Omar Besbes and Assaf Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations Research*, 57:1407–1420, 2009.

Omar Besbes and Assaf J. Zeevi. Blind network revenue management. *Operations Research*, 60(6):1537–1550, 2012.

Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandit algorithms with supervised learning guarantees. In *14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Avrim Blum, Vijay Kumar, Atri Rudra, and Felix Wu. Online learning in online auctions. In *14th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 202–204, 2003.

Arnoud V. Den Boer. Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science*, 20(1), June 2015.

Sébastien Bubeck. *Bandits Games and Clustering Foundations*. PhD thesis, Univ. Lille 1, 2010.

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.

Sébastien Bubeck, Gilles Stoltz, and Jia Yuan Yu. Lipschitz bandits without the lipschitz constant. In *22nd Intl. Conf. on Algorithmic Learning Theory (ALT)*, pages 144–158, 2011c.

Sébastien Bubeck, Ofer Dekel, Tomer Koren, and Yuval Peres. Bandit convex optimization: \(\sqrt{T}\) regret in one dimension. In *28th Conf. on Learning Theory (COLT)*, pages 266–278, 2015.

Sébastien Bubeck, Yin Tat Lee, and Ronen Eldan. Kernel-based methods for bandit convex optimization. In *49th ACM Symp. on Theory of Computing (STOC)*, pages 72–85. ACM, 2017.

Sébastien Bubeck, Michael B. Cohen, and Yuanzhi Li. Sparsity, variance and curvature in multi-armed bandits. In *29nd Intl. Conf. on Algorithmic Learning Theory (ALT)*, 2018.

Adam Bull. Adaptive-treed bandits. *Bernoulli J. of Statistics*, 21(4):2289–2307, 2015.

Alexandra Carpentier and Andrea Locatelli. Tight (lower) bounds for the fixed budget best arm identification bandit problem. In *29th Conf. on Learning Theory (COLT)*, pages 590–604, 2016.

Nicolò Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge Univ. Press, 2006.

Nicolò Cesa-Bianchi, Pierre Gaillard, Claudio Gentile, and Sébastien Gerchinovitz. Algorithmic chaining and the role of partial feedback in online nonparametric learning. In *30th Conf. on Learning Theory (COLT)*, pages 465–481, 2017.

Tianyi Chen and Georgios B Giannakis. Bandit convex optimization for scalable and dynamic iot management. *IEEE Internet of Things Journal*, 2018.

Tianyi Chen, Qing Ling, and Georgios B Giannakis. An online convex optimization approach to proactive network resource allocation. *IEEE Transactions on Signal Processing*, 65(24):6350–6364, 2017.

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.

Varsha Dani, Thomas P. Hayes, and Sham Kakade. Stochastic Linear Optimization under Bandit Feedback. In *21th Conf. on Learning Theory (COLT)*, pages 355–366, 2008.

Constantinos Daskalakis, Alan Deckelbaum, and Anthony Kim. Near-optimal no-regret algorithms for zero-sum games. *Games and Economic Behavior*, 92:327–348, 2015. Preliminary version in *ACM-SIAM SODA 2011*.

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.

Thomas Desautels, Andreas Krause, and Joel Burdick. Parallelizing exploration-exploitation tradeoffs with gaussian process bandit optimization. In *29th Intl. Conf. on Machine Learning (ICML)*, 2012.

Nikhil Devanur and Sham M. Kakade. The price of truthfulness for pay-per-click auctions. In *10th ACM Conf. on Electronic Commerce (EC)*, pages 99–106, 2009.

Nikhil R. Devanur and Thomas P. Hayes. The AdWords problem: Online keyword matching with budgeted bidders under random permutations. In *10th ACM Conf. on Electronic Commerce (EC)*, pages 71–78, 2009.

Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *12th ACM Conf. on Electronic Commerce (EC)*, pages 29–38, 2011.

Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. *J. ACM*, 66(1):7:1–7:41, 2019. Preliminary version in *ACM EC 2011*.

Wenkui Ding, Tao Qin, Xu-Dong Zhang, and Tie-Yan Liu. Multi-armed bandit with budget constraint and variable costs. In *27th AAAI Conference on Artificial Intelligence (AAAI)*, 2013.

Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal leanring for contextual bandits. In *27th Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2011.

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.

Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In *18th Annual European Symp. on Algorithms (ESA)*, pages 182–194, 2010.

Abraham Flaxman, Adam Kalai, and H. Brendan McMahan. Online Convex Optimization in the Bandit Setting: Gradient Descent without a Gradient. In *16th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 385–394, 2005.

Dylan J. Foster, Zhiyuan Li, Thodoris Lykouris, Karthik Sridharan, and Éva Tardos. Learning in games: Robustness of fast convergence. In *29th Advances in Neural Information Processing Systems (NIPS)*, pages 4727–4735, 2016.

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

Aurélien Garivier and Olivier Cappé. The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond. In *24th Conf. on Learning Theory (COLT)*, 2011.

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, 2011.

Jean-Bastien Grill, Michal Valko, and Rémi Munos. Black-box optimization of noisy functions with unknown smoothness. In *28th Advances in Neural Information Processing Systems (NIPS)*, 2015.

Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low–distortion embeddings. In *44th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 534–543, 2003.

András György, Levente Kocsis, Ivett Szabó, and Csaba Szepesvári. Continuous time associative bandit problems. In *20th Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 830–835, 2007.

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. *Journal of Machine Learning Research*, 12: 1287–1311, 2011. Preliminary version published in *ACM-SIAM SODA 2009*.

Elad Hazan and Kfir Y. Levy. Bandit convex optimization: Towards tight bounds. In *27th Advances in Neural Information Processing Systems (NIPS)*, pages 784–792, 2014.

Elad Hazan and Nimrod Megiddo. Online Learning with Prior Information. In *20th Conf. on Learning Theory (COLT)*, pages 499–513, 2007.

Chien-Ju Ho, Aleksandrs Slivkins, and Jennifer Wortman Vaughan. Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems. *J. of Artificial Intelligence Research*, 55:317–359, 2016. Preliminary version appeared in *ACM EC 2014*.

Junya Honda and Akimichi Takemura. An asymptotically optimal bandit algorithm for bounded support models. In *23rd Conf. on Learning Theory (COLT)*, 2010.

Justin Hsu, Zhiyi Huang, Aaron Roth, and Zhiwei Steven Wu. Jointly private convex programming. In *27th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 580–599, 2016.

Nicole Immorlica, Karthik Abinav Sankararaman, Robert Schapire, and Aleksandrs Slivkins. Adversarial bandits with knapsacks, 2018. Working paper. Available at `https://arxiv.org/abs/1811.11881`.

D.R. Karger and M. Ruhl. Finding Nearest Neighbors in Growth-restricted Metrics. In *34th ACM Symp. on Theory of Computing (STOC)*, pages 63–66, 2002.

Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *23rd Intl. Conf. on Algorithmic Learning Theory (ALT)*, pages 199–213, 2012.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *J. of Machine Learning Research (JMLR)*, 17:1:1–1:42, 2016.

Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *35th Intl. Conf. on Machine Learning (ICML)*, pages 2564–2572, 2018.

Jon Kleinberg and Eva Tardos. *Algorithm Design*. Addison Wesley, 2005.

Jon Kleinberg, Aleksandrs Slivkins, and Tom Wexler. Triangulation and embedding using small sets of beacons. *J. of the ACM*, 56(6), September 2009. Subsumes conference papers in *IEEE FOCS 2004* and *ACM-SIAM SODA 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 and Aleksandrs Slivkins. Sharp dichotomies for regret minimization in metric spaces. In *21st ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2010.

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. *J. of the ACM*, 2019. To appear. Merged and revised version of conference papers in *ACM STOC 2008* and *ACM-SIAM SODA 2010*. Also available at `http://arxiv.org/abs/1312.1277`.

Robert D. Kleinberg and Frank T. Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *IEEE Symp. on Foundations of Computer Science (FOCS)*, 2003.

Levente Kocsis and Csaba Szepesvari. Bandit Based Monte-Carlo Planning. In *17th European Conf. on Machine Learning (ECML)*, pages 282–293, 2006.

Wouter M. Koolen, Manfred K. Warmuth, and Jyrki Kivinen. Hedging structured concepts. In *23rd Conf. on Learning Theory (COLT)*, 2010.

Andreas Krause and Cheng Soon Ong. Contextual gaussian process bandit optimization. In *25th Advances in Neural Information Processing Systems (NIPS)*, pages 2447–2455, 2011.

Branislav Kveton, Zheng Wen, Azin Ashkan, Hoda Eydgahi, and Brian Eriksson. Matroid bandits: Fast combinatorial optimization with learning. In Nevin L. Zhang and Jin Tian, editors, *Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 420–429, 2014.

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.

Tyler Lu, Dávid Pál, and Martin Pál. Showing Relevant Ads via Lipschitz Context Multi-Armed Bandits. In *14th Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Thodoris Lykouris, Vahab Mirrokni, and Renato Paes-Leme. Stochastic bandits robust to adversarial corruptions. In *50th ACM Symp. on Theory of Computing (STOC)*, 2018.

Mehrdad Mahdavi, Rong Jin, and Tianbao Yang. Trading regret for efficiency: online convex optimization with long term constraints. *J. of Machine Learning Research (JMLR)*, 13(Sep):2503–2528, 2012.

Mehrdad Mahdavi, Tianbao Yang, and Rong Jin. Stochastic convex optimization with multiple objectives. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1115–1123, 2013.

Odalric-Ambrym Maillard and Rémi Munos. Online Learning in Adversarial Lipschitz Environments. In *European Conf. on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, pages 305–320, 2010.

Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz. A finite-time analysis of multi-armed bandits problems with kullback-leibler divergences. In *24th Conf. on Learning Theory (COLT)*, 2011.

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.

Stanislav Minsker. Estimation of extreme values and associated level sets of a regression function via selective sampling. In *26th Conf. on Learning Theory (COLT)*, pages 105–121, 2013.

Marco Molinaro and R. Ravi. Geometry of online packing linear programs. In *39th Intl. Colloquium on Automata, Languages and Programming (ICALP)*, pages 701–713, 2012.

Rémi Munos. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *25th Advances in Neural Information Processing Systems (NIPS)*, pages 783–791, 2011.

Rémi Munos. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning*, 7(1):1–129, 2014.

Rémi Munos and Pierre-Arnaud Coquelin. Bandit algorithms for tree search. In *23rd Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2007.

Michael J Neely and Hao Yu. Online convex optimization with time-varying constraints. *arXiv preprint arXiv:1702.04783*, 2017.

Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for Taxonomies: A Model-based Approach. In *SIAM Intl. Conf. on Data Mining (SDM)*, 2007.

Alexander Rakhlin and Karthik Sridharan. Online learning with predictable sequences. In *26th Conf. on Learning Theory (COLT)*, pages 993–1019, 2013.

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

Alexander Rakhlin, Karthik Sridharan, and Ambuj Tewari. Online learning via sequential complexities. *J. of Machine Learning Research (JMLR)*, 16:155–186, 2015.

Anshuka Rangi, Massimo Franceschetti, and Long Tran-Thanh. Unifying the stochastic and the adversarial bandits with knapsack. *arXiv preprint arXiv:1811.12253*, 2018.

Adrian Rivera, He Wang, and Huan Xu. Online saddle point problem with applications to constrained online convex optimization. *arXiv preprint arXiv:1806.08301*, 2018.

Ryan Rogers, Aaron Roth, Jonathan Ullman, and Zhiwei Steven Wu. Inducing approximately optimal flow using truthful mediators. In *16th ACM Conf. on Electronic Commerce (EC)*, pages 471–488, 2015.

Aaron Roth, Jonathan Ullman, and Zhiwei Steven Wu. Watch and learn: Optimizing from revealed preferences feedback. In *48th ACM Symp. on Theory of Computing (STOC)*, pages 949–962, 2016.

Aaron Roth, Aleksandrs Slivkins, Jonathan Ullman, and Zhiwei Steven Wu. Multidimensional dynamic pricing for welfare maximization. In *18th ACM Conf. on Electronic Commerce (EC)*, pages 519–536, 2017.

Paat Rusmevichientong and John N. Tsitsiklis. Linearly parameterized bandits. *Mathematics of Operations Research*, 35(2):395–411, 2010.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Math. Oper. Res.*, 39(4):1221–1243, 2014.

Karthik Abinav Sankararaman and Aleksandrs Slivkins. Combinatorial semi-bandits with knapsacks. In *Intl. Conf. on Artificial Intelligence and Statistics (AISTATS)*, pages 1760–1770, 2018.

Manfred Schroeder. *Fractal, Chaos and Power Laws: Minutes from an Infinite Paradise*. W. H. Freeman and Co., 1991.

Yevgeny Seldin and Gabor Lugosi. A lower bound for multi-armed bandits with expert advice. In *13th European Workshop on Reinforcement Learning (EWRL)*, 2016.

Yevgeny Seldin and Gábor Lugosi. An improved parametrization and analysis of the EXP3++ algorithm for stochastic and adversarial bandits. In *30th Conf. on Learning Theory (COLT)*, 2017.

Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *31th Intl. Conf. on Machine Learning (ICML)*, 2014.

Ohad Shamir. On the complexity of bandit linear optimization. In *28th Conf. on Learning Theory (COLT)*, pages 1523–1551, 2015.

Aleksandrs Slivkins. Towards fast decentralized construction of locality-aware overlay networks. In *26th Annual ACM Symp. on Principles Of Distributed Computing (PODC)*, pages 89–98, 2007.

Aleksandrs Slivkins. Multi-armed bandits on implicit metric spaces. In *25th Advances in Neural Information Processing Systems (NIPS)*, 2011.

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.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *27th Intl. Conf. on Machine Learning (ICML)*, pages 1015–1022, 2010.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E. Schapire. Fast convergence of regularized learning in games. In *28th Advances in Neural Information Processing Systems (NIPS)*, pages 2989–2997, 2015.

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.

Kunal Talwar. Bypassing the embedding: Algorithms for low-dimensional metrics. In *36th ACM Symp. on Theory of Computing (STOC)*, pages 281–290, 2004.

William R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 1933.

Long Tran-Thanh, Archie Chapman, Enrique Munoz de Cote, Alex Rogers, and Nicholas R. Jennings. ϵ-first policies for budget-limited multi-armed bandits. In *24th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1211–1216, 2010.

Long Tran-Thanh, Archie Chapman, Alex Rogers, and Nicholas R. Jennings. Knapsack based optimal policies for budget-limited multi-armed bandits. In *26th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1134–1140, 2012.

Michal Valko, Alexandra Carpentier, and Rémi Munos. Stochastic simultaneous optimistic optimization. In *30th Intl. Conf. on Machine Learning (ICML)*, pages 19–27, 2013.

Zizhuo Wang, Shiming Deng, and Yinyu Ye. Close the gaps: A learning-while-doing algorithm for single-product revenue management problems. *Operations Research*, 62(2):318–331, 2014.

Chen-Yu Wei and Haipeng Luo. More adaptive algorithms for adversarial bandits. In *31st Conf. on Learning Theory (COLT)*, 2018.