Chapter 2

Bandits with IID Rewards

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

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

2.1 Model and examples

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

Multi-armed bandits

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

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

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

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

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

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

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

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

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

2. **Ad selection**: In website advertising, a user visits a webpage, and a learning algorithm selects one of many possible ads to display. If ad a is displayed, the website observes whether the user clicks on the ad, in which case the advertiser pays some amount $v_a \in [0,1]$. So each ad is an arm, and the paid amount is the reward.

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

3. **Medical Trials:** a patient visits a doctor and the doctor can proscribe one of several possible treatments, and observes the treatment effectiveness. Then the next patient arrives, and so forth. For simplicity of this example, the effectiveness of a treatment is quantified as a number in $[0,1]$. So here each treatment can be considered as an arm, and the reward is defined as the treatment effectiveness. As an idealized assumption, each patient is drawn independently from a fixed distribution over patients, so the effectiveness of a given treatment is i.i.d.

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

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

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

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

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

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

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

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

2.2 Simple algorithms: uniform exploration

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

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

Algorithm 1: Explore-First with parameter N.

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

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

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

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

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

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

Remark 2.2. With this approach, one does not need to worry about probability in the rest of the proof. Indeed, the probability has been taken care of by defining the clean event and observing that (2.2) holds! And we do not need to worry about the bad event either — essentially, because its probability is so tiny. We will use this "clean event" approach in many other proofs, to help simplify the technical details. The downside is that it usually leads to worse constants that can be obtained by a proof that argues about probabilities more carefully.

For simplicity, let us start with the case of $K = 2$ arms. Consider the clean event. We will show that if we chose the worse arm, it is not so bad because the expected rewards for the two arms would be close.

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

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

Re-arranging the terms, it follows that

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

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

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

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

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

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

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

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

For $K > 2$ arms, we have to apply the union bound for (2.2) over the K arms, and then follow the same argument as above. Note that the value of T is greater than K, since we need to explore each arm at least once. For the final regret computation, we will need to take into account the dependence on K: specifically, regret accumulated in exploration phase is now upper-bounded by KN. Working through the proof, we obtain $R(T) \leq NK + O(\sqrt{\frac{\log T}{N}} \times T)$. As before, we approximately minimize it by approximately minimizing the two summands. Specifically, we plug in $N = (T/K)^{2/3} \cdot O(\log T)^{1/3}$. Completing the proof same way as in (2.3), we obtain:

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

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

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

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

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

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

2.3 Advanced algorithms: adaptive exploration

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

Let's start with the case of $K = 2$ arms. One natural idea is to alternate them until we find that one arm is much better than the other, at which time we abandon the inferior one. But how to define "one arm is much better" exactly?

2.3.1 Clean event and confidence bounds

To flesh out the idea mentioned above, and to set up the stage for some other algorithms in this class, let us do some probability with our old friend Hoeffding Inequality.

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

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

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

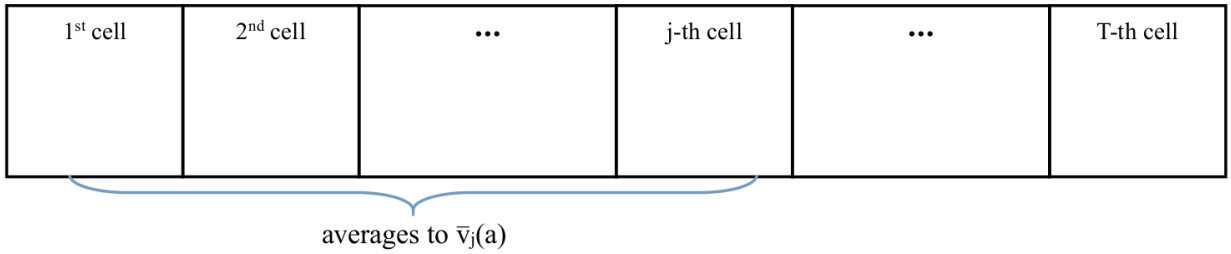

Figure 2.1: the j-th cell contains the reward of the j-th time we pull arm a, *i.e.,* reward of arm a when $n_t(a) = j$

variables, (2.4) would follow immediately if $n_t(a)$ were fixed in advance. However, $n_t(a)$ is itself is a random variable. So we need a slightly more careful argument, presented below.

Let us imagine there is a tape of length T for each arm a, with each cell independently sampled from $\mathcal{D}_a$, as shown in Figure 2.1. Without loss of generality, this table encodes rewards as follows: the j-th time a given arm a is chosen by the algorithm, its reward is taken from the j-th cell in this arm's tape. Let $\bar{v}_j(a)$ represent the average reward at arm a from first j times that arm a is chosen. Now one can use Hoeffding Inequality to derive that

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

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

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

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

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

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

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

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

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

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

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

2.3.2 Successive Elimination algorithm

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

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

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

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

Let t be the last round when we did *not* invoke the stopping rule, i.e., when the confidence intervals of the two arms still overlap (see Figure 2.2). Then

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

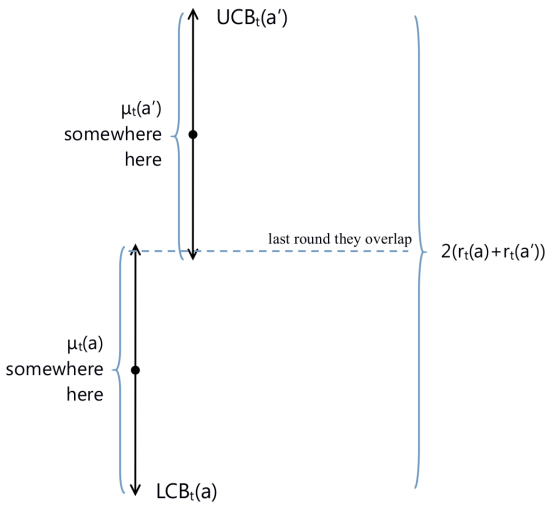

Figure 2.2: t is the last round that the two confidence intervals still overlap

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

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

Then total regret accumulated till round t is

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

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

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

We proved the following:

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

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

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

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

Algorithm 4: Successive Elimination algorithm

To analyze the performance of this algorithm, it suffices to focus on the clean event (2.6); as in the case of $k = 2$ arms, the contribution of the "bad event" $\bar{\mathcal{E}}$ can be neglected.

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

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

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

We have proved the following crucial property:

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

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

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

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

Recall that $\mathcal{A}$ denotes the set of all K arms, and let $\mathcal{A}^+ = \{a : \mu(a) < \mu(a^*)\}$ be the set of all arms that contribute to regret. Then:

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

Since $f(x) = \sqrt{x}$ is a real concave function, and $\sum_{a \in \mathcal{A}} n_t(a) = t$, by Jensen's Inequality (Theorem A.1) we have

$$\frac{1}{K} \sum_{a \in \mathcal{A}} \sqrt{n_t(a)} \leq \sqrt{\frac{1}{K} \sum_{a \in \mathcal{A}} n_t(a)} = \sqrt{\frac{t}{K}}.$$

Plugging this into (2.8), we see that $R(t) \leq O(\sqrt{Kt \log T})$. Thus, we have proved:

Theorem 2.8. *Successive Elimination algorithm achieves regret*

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

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

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

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

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

Theorem 2.9. *Successive Elimination algorithm achieves regret*

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

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

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

Remark 2.12. It is instructive to derive Theorem 2.8 in a different way: starting from the logarithmic regret bound in (2.10). Informally, we need to get rid of arbitrarily small $\Delta(a)$'s in the denominator. Let us fix some $\epsilon > 0$, then regret consists of two parts:

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

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

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

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

2.3.3 UCB1 Algorithm

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

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

Algorithm 5: UCB1 Algorithm

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

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

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

It follows that

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

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

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

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

2.4 Further remarks

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

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

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

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

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

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

Bandits with predictions. While the standard goal for bandit algorithms is to maximize cumulative reward, an alternative goal is to output a prediction a_t^* after each round t. The algorithm is then graded only on the quality of these predictions. In particular, it does not matter how much reward is accumulated. There are two standard ways to make this objective formal: (i) minimize instantaneous regret $\mu^* - \mu(a_t^*)$, and (ii) maximize the probability of choosing the best arm: $\Pr[a_t^* = a^*]$. Essentially, good algorithms for cumulative regret, such as Successive Elimination and UCB1, are also good for this version (more on this in the exercises). However, improvements are possible in some regimes (*e.g.,* Mannor and Tsitsiklis, 2004; Even-Dar et al., 2006; Bubeck et al., 2011a; Audibert et al., 2010).

2.5 Exercises

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

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

Hint: Use a version of Hoeffding Inequality with ranges.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Exercise 2.5 (Doubling trick). Take any bandit algorithm $\mathcal{A}$ for fixed time horizon T. Convert it to an algorithm $\mathcal{A}_\infty$ which runs forever, in phases $i = 1, 2, 3, \ldots$ 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.)