Contents

Chapter 1

Bandits

The **multi-armed bandits** (MAB) setting is a simple but powerful setting for studying the basic challenges of RL. In this setting, an agent repeatedly chooses from a fixed set of actions, called **arms**, each of which has an associated reward distribution. The agent's goal is to maximize the total reward it receives over some time period.

States	Actions	Rewards
None	Finite	Stochastic

In particular, we'll spend a lot of time discussing the **Exploration-Exploitation Tradeoff**: should the agent choose new actions to learn more about the environment, or should it choose actions that it already knows to be good?

Example 1.0.1: Online advertising

Let's suppose you, the agent, are an advertising company. You have K different ads that you can show to users; For concreteness, let's suppose there's just a single user. You receive 1 reward if the user clicks the ad, and 0 otherwise. Thus, the unknown *reward distribution* associated to each ad is a Bernoulli distribution defined by the probability that the user clicks on the ad. Your goal is to maximize the total number of clicks by the user.

Example 1.0.2: Clinical trials

Suppose you're a pharmaceutical company, and you're testing a new drug. You have K different dosages of the drug that you can administer to patients. You receive 1 reward if the patient recovers, and 0 otherwise. Thus, the unknown *reward distribution* associated to each dosage is a Bernoulli distribution defined by the probability that the patient recovers. Your goal is to maximize the total number of patients that recover.

In this chapter, we will introduce the multi-armed bandits setting, and discuss some of the challenges that arise when trying to solve problems in this setting. We will also introduce some of the key concepts that we will use throughout the book, such as regret and exploration-exploitation tradeoffs.

1.1 Introduction

The name "multi-armed bandits" comes from slot machines in casinos, which are often called "one-armed bandits" since they have one arm (the lever) and take money from the player.

Let K denote the number of arms. We'll label them $0, \dots, K-1$ and use *superscripts* to indicate the arm index; since we seldom need to raise a number to a power, this hopefully won't cause much confusion. For simplicity, we'll assume rewards are *bounded* between 0 and 1. Then each arm has an unknown reward distribution $\nu^k \in \Delta([0,1])$ with mean $\mu^k = \mathbb{E}_{r\sim\nu^k}[r]$.

In pseudocode, the agent's interaction with the MAB environment can be described by the following process:

for $t = 0, \dots, T$ **do**
  Agent chooses $a_t \in [K]$
  Agent receives $r_t \sim \nu^{a_t}$
  Agent updates its internal state
end for

What's the *optimal* strategy for the agent, i.e. the one that achieves the highest expected reward? Convince yourself that the agent should try to always pull the arm with the highest expected reward $\mu^\star := \max_{k\in[K]} \mu^k$.

The goal, then, can be rephrased as to minimize the **regret**, defined below:

Definition 1.1.1: Regret

The agent's **regret** after T timesteps is defined as

$$\text{Regret}_T := \sum_{t=0}^{T-1} \mu^\star - \mu^{a_t} \tag{1.1}$$

Note that this depends on the *true means* of the pulled arms, *not* the actual observed rewards. We typically think of this as a random variable where the randomness comes from the agent's strategy (i.e. the sequence of actions $a_0, \dots, a_{T-1}$).

Throughout the chapter, we will try to upper bound the regret of various algorithms in two different senses:

1. Upper bound the *expected* regret, i.e. show $\mathbb{E}[\text{Regret}_T] \le M_T$.
2. Find a high-probability upper bound on the regret, i.e. show $\mathbb{P}(\text{Regret}_T \le M_{T,\delta}) \ge 1-\delta$.

Note that these two different approaches say very different things about the regret. The first approach says that the *average* regret is at most M_T. However, the agent might still achieve higher regret on many runs. The second approach says that, *with high probability*, the agent will achieve regret at most $M_{T,\delta}$. However, it doesn't say anything about the regret in the remaining δ fraction of runs, which might be arbitrarily high.

We'd like to achieve **sublinear regret** in expectation, i.e. $\mathbb{E}[\text{Regret}_T] = o(T)$. That is, as we learn more about the environment, we'd like to be able to exploit that knowledge to achieve higher rewards.

The rest of the chapter comprises a series of increasingly sophisticated MAB algorithms.

1.1.1 O notation

Throughout this chapter and the rest of the book, we will describe the asymptotic behavior of a function using O notation.

For two functions $f(t)$ and $g(t)$, we say that $f(t) \leq O(g(t))$ if f is asymptotically upper bounded by g. Formally, this means that there exists some constant $C > 0$ such that $f(t) \leq C \cdot g(t)$ for all t past some point t_0.

We say $f(t) < o(g(t))$ if asymptotically f grows strictly slower than g. Formally, this means that for *any* scalar $C > 0$, there exists some t_0 such that $f(t) \leq C \cdot g(t)$ for all $t > t_0$. Equivalently, we say $f(t) < o(g(t))$ if $\lim_{t\to\infty} f(t)/g(t) = 0$.

$f(t) = \Theta(g(t))$ means that f and g grow at the same rate asymptotically. That is, $f(t) \leq O(g(t))$ and $g(t) \leq O(f(t))$.

Finally, we use $f(t) \geq \Omega(g(t))$ to mean that $g(t) \leq O(f(t))$, and $f(t) > \omega(g(t))$ to mean that $g(t) < o(f(t))$.

We also use the notation $\widetilde{O}(g(t))$ to hide logarithmic factors. That is, $f(t) = \widetilde{O}(g(t))$ if there exists some constant C such that $f(t) \leq C \cdot g(t) \cdot \log^k(t)$ for some k and all t.

Occasionally, we will also use $O(f(t))$ (or one of the other symbols) as shorthand to manipulate function classes. For example, we might write $O(f(t)) + O(g(t)) = O(f(t) + g(t))$ to mean that the sum of two functions in $O(f(t))$ and $O(g(t))$ is in $O(f(t) + g(t))$.

1.2 Pure exploration (random guessing)

A trivial strategy is to always choose arms at random (i.e. "pure exploration").

Definition 1.2.1: Pure exploration

for $t \leftarrow 0$ to $T - 1$ **do**
  Choose $a_t \sim \text{Unif}([K])$
  Observe $r_t \sim \nu^{a_t}$
end for

Note that

$$\mathop{\mathbb{E}}_{a_t \sim \text{Unif}([K])}[\mu^{a_t}] = \bar{\mu} = \frac{1}{K}\sum_{k=1}^{K} \mu^k$$

so the expected regret is simply

$$\begin{aligned}\mathbb{E}[\text{Regret}_T] &= \sum_{t=0}^{T-1} \mathbb{E}[\mu^\star - \mu^{a_t}] \\ &= T(\mu^\star - \bar{\mu}) > 0.\end{aligned}$$

This scales as $\Theta(T)$, i.e. *linear* in the number of timesteps T. There's no learning here: the agent doesn't use any information about the environment to improve its strategy.

1.3 Pure greedy

How might we improve on pure exploration? Instead, we could try each arm once, and then commit to the one with the highest observed reward. We'll call this the **pure greedy** strategy.

Definition 1.3.1: Pure greedy

for $k \leftarrow 0$ to $K-1$ **do** ▷ Exploration phase
  Observe $r^k \sim \nu^k$
end for
$\widehat{k} \leftarrow \arg\max_{k \in [K]} r^k$
for $t \leftarrow K$ to $T-1$ **do** ▷ Exploitation phase
  Observe $r_t \sim \nu^{\widehat{k}}$
end for

Note we've used superscripts r^k during the exploration phase to indicate that we observe exactly one reward for each arm. Then we use subscripts r_t during the exploitation phase to indicate that we observe a sequence of rewards from the chosen greedy arm $\widehat{k}$.

How does the expected regret of this strategy compare to that of pure exploration? We'll do a more general analysis in the following section. Now, for intuition, suppose there's just $K = 2$ arms, with Bernoulli reward distributions with means $\mu^0 > \mu^1$.

Let r^0 be the random reward from the first arm and r^1 be the random reward from the second. If $r^0 > r^1$, then we achieve constant regret of $\mu^0 - \mu^1$ (from the one time we pull arm 1 during exploration). Otherwise, we achieve regret $T \cdot (\mu^0 - \mu^1)$. Thus, the expected regret is simply:

$$\begin{aligned}\mathbb{E}[\text{Regret}_T] &= \mathbb{P}(r^0 < r^1) \cdot T \cdot (\mu^0 - \mu^1) + c \\ &= (1 - \mu^0)\mu^1 \cdot T \cdot (\mu^0 - \mu^1) + c\end{aligned}$$

Which is still $\Theta(T)$, the same as pure exploration! Can we do better?

1.4 Explore-then-commit

We can improve the pure greedy algorithm as follows: let's reduce the variance of the reward estimates by pulling each arm $N_{\text{explore}} > 1$ times before committing. This is called the **explore-then-commit** strategy.

Definition 1.4.1: Explore-then-commit

Input: $N_{\text{explore}} \leq T/K$
for $k \leftarrow 0$ to $K-1$ **do** ▷ Exploration phase
  for $i \leftarrow 0$ to $N_{\text{explore}} - 1$ **do**
    $r_i^k \sim \nu^k$
  end for
  $\widehat{\mu}^k \leftarrow \frac{1}{N_{\text{explore}}} \sum_{i=0}^{N_{\text{explore}}-1} r_i^k$
end for
$\widehat{k} \leftarrow \arg\max_k(\widehat{\mu}^k)$
for $t \leftarrow N_{\text{explore}}K$ to $T-1$ **do** ▷ Exploitation phase
  $r_t \sim \nu^{\widehat{k}}$
end for

(Note that the "pure greedy" strategy is just the special case where $N_{\text{explore}} = 1$.)

1.4.1 ETC regret analysis

Let's analyze the expected regret of this strategy by splitting it up into the exploration and exploitation phases.

Exploration phase. This phase takes $N_{\text{explore}}K$ timesteps. Since at each step we incur at most 1 regret, the total regret is at most $N_{\text{explore}}K$.

Exploitation phase. This will take a bit more effort. We'll prove that for any total time T, we can choose N_{explore} such that with arbitrarily high probability, the regret is sublinear. We know the regret from the exploitation phase is

$$T_{\text{exploit}}(\mu^\star - \mu^{\widehat{k}}) \qquad \text{where} \qquad T_{\text{exploit}} := T - N_{\text{explore}}K.$$

So we'd like to bound $\mu^\star - \mu^{\widehat{k}} < o(1)$ (as a function of T) in order to achieve sublinear regret. How can we do this?

Let's define $\Delta^k = \widehat{\mu}^k - \mu^k$ to denote how far the mean estimate for arm k is from the true mean. How can we bound this quantity? We'll use the following useful inequality for i.i.d. bounded random variables:

Theorem 1.4.1: Hoeffding's inequality

Let $X_0, \ldots, X_{n-1}$ be i.i.d. random variables with $X_i \in [0,1]$ almost surely for each $i \in [n]$. Then for any $\delta > 0$,

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i])\right| > \sqrt{\frac{\ln(2/\delta)}{2n}}\right) \leq \delta. \tag{1.2}$$

(The proof of this inequality is beyond the scope of this book.) We can apply this directly to the rewards for a given arm k, since the rewards from that arm are i.i.d.:

$$\mathbb{P}\left(|\Delta^k| > \sqrt{\frac{\ln(2/\delta)}{2N_{\text{explore}}}}\right) \leq \delta. \tag{1.3}$$

But note that we can't apply this to arm $\widehat{k}$ directly since $\widehat{k}$ is itself a random variable. Instead, we need to "uniform-ize" this bound across *all* the arms, i.e. bound the error across all the arms simultaneously, so that the resulting bound will apply *no matter what* $\widehat{k}$ "crystallizes" to.

The **union bound** provides a simple way to do this:

Theorem 1.4.2: Union bound

Consider a set of events $A_0, \ldots, A_{n-1}$. Then

$$\mathbb{P}(\exists i \in [n].A_i) \leq \sum_{i=0}^{n-1}\mathbb{P}(A_i).$$

In particular, if $\mathbb{P}(A_i) \geq 1-\delta$ for each $i \in [n]$, we have

$$\mathbb{P}(\forall i \in [n].A_i) \geq 1 - n\delta.$$

Exercise: Prove the second statement above.

Applying the union bound across the arms for the l.h.s. event of 1.3, we have

$$\mathbb{P}\left(\forall k \in [K], |\Delta^k| \leq \sqrt{\frac{\ln(2/\delta)}{2N_{\text{explore}}}}\right) \geq 1 - K\delta$$

Then to apply this bound to $\widehat{k}$ in particular, we can apply the useful trick of "adding zero":

$$\begin{aligned}
\mu^{k^\star} - \mu^{\widehat{k}} &= \mu^{k^\star} - \mu^{\widehat{k}} + (\widehat{\mu}^{k^\star} - \widehat{\mu}^{k^\star}) + (\widehat{\mu}^{\widehat{k}} - \widehat{\mu}^{\widehat{k}}) \\
&= \Delta^{\widehat{k}} - \Delta^{k^*} + \underbrace{(\widehat{\mu}^{k^\star} - \widehat{\mu}^{\widehat{k}})}_{\leq 0 \text{ by definition of } \widehat{k}} \\
&\leq 2\sqrt{\frac{\ln(2K/\delta')}{2N_{\text{explore}}}} \text{ with probability at least } 1-\delta'
\end{aligned}$$

where we've set $\delta' = K\delta$. Putting this all together, we've shown that, with probability $1 - \delta'$,

$$\text{Regret}_T \leq N_{\text{explore}} K + T_{\text{exploit}} \cdot \sqrt{\frac{2\ln(2K/\delta')}{N_{\text{explore}}}}.$$

Note that it suffices for N_{explore} to be on the order of $\sqrt{T}$ to achieve sublinear regret. In particular, we can find the optimal N_{explore} by setting the derivative of the r.h.s. to zero:

$$0 = K - T_{\text{exploit}} \cdot \frac{1}{2}\sqrt{\frac{2\ln(2K/\delta')}{N_{\text{explore}}^3}}$$

$$N_{\text{explore}} = \left(T_{\text{exploit}} \cdot \frac{\sqrt{\ln(2K/\delta')/2}}{K} \right)^{2/3}$$

Plugging this into the expression for the regret, we have (still with probability $1 - \delta'$)

$$\begin{aligned} \text{Regret}_T &\leq 3T^{2/3} \sqrt[3]{K\ln(2K/\delta')/2} \\ &\leq \widetilde{O}(T^{2/3} K^{1/3}). \end{aligned}$$

The ETC algorithm is rather "abrupt" in that it switches from exploration to exploitation after a fixed number of timesteps. In practice, it's often better to use a more gradual transition, which brings us to the *epsilon-greedy* algorithm.

1.5 Epsilon-greedy

Instead of doing all of the exploration and then all of the exploitation separately – which additionally requires knowing the time horizon beforehand – we can instead interleave exploration and exploitation by, at each timestep, choosing a random action with some probability. We call this the **epsilon-greedy** algorithm.

Definition 1.5.1: Epsilon-greedy

To ensure that the sample means S^k/N^k are well-defined, we initially pull each arm once. Afterwards, at each timestep t, we choose a random arm with probability $\epsilon(t)$ and the greedy arm with probability $1 - \epsilon(t)$. Note that we let ϵ vary over time. In particular we might want to gradually *decrease* ϵ as we learn more about the reward distributions over time.

Input: $\epsilon : \mathbb{N} \to [0, 1]$
for $k \in [K]$ **do** ▷ Initialization
  $r_k \sim \nu^k$
  $S^k \leftarrow r_k$ ▷ Total reward for arm k

$N^k \leftarrow 1$ ▷ Number of pulls for arm k
end for
for $t \leftarrow K+1$ to T **do**
  if random() $< \epsilon(t)$ **then**
    $k \sim$ Unif($[K]$) ▷ Exploration
  else
    $k \leftarrow \arg\max_k \left(\frac{S^k}{N^k}\right)$ ▷ Exploitation
  end if
  $r_t \sim \nu^k$
  $S^k \leftarrow S^k + r_t$
  $N^k \leftarrow N^k + 1$
end for

Setting $\epsilon_t = \sqrt[3]{K \ln(t)/t}$ also achieves a regret of $\widetilde{O}(t^{2/3}K^{1/3})$ (ignoring the logarithmic factors), the same as ETC. (We omit the proof here.) Why might we choose epsilon-greedy over ETC? In ETC, we had to set N_{explore} based on the total number of timesteps T. But the epsilon-greedy algorithm actually handles the exploration *automatically*: the regret rate holds for *any* t, and doesn't depend on the final horizon T.

But the way these algorithms explore is rather naive: we've been exploring *uniformly* across all the arms. But what if we could be smarter about it, and explore *more* for arms that we're less certain about?

1.6 Upper Confidence Bound (UCB)

To quantify how *certain* we are about the mean of each arm, we'll compute *confidence intervals* for our estimators, and then choose the arm with the highest *upper confidence bound*. This operates on the principle of **the benefit of the doubt (i.e. optimism in the face of uncertainty)**: we'll choose the arm that we're most optimistic about.

In particular, for each arm k at time t, we'd like to compute some upper confidence bound M_t^k such that $\widehat{\mu}_t^k \leq M_t^k$ with high probability, and then choose $a_t := \arg\max_{k\in[K]} M_t^k$. But how should we compute M_t^k?

In subsection 1.4.1, we were able to compute this bound using Hoeffding's inequality, which assumes that the number of samples is *fixed*. This was the case in ETC (where we pull each arm N_{explore} times), but in UCB, the number of times we pull each arm depends on the agent's actions, which in turn depend on the random rewards and are therefore stochastic. So we *can't* use Hoeffding's inequality directly.

Instead, we'll apply the same trick we used in the ETC analysis: we'll use the **union bound** to compute a *looser* bound that holds *uniformly* across all timesteps and arms. Let's introduce some notation to discuss this.

Let N_t^k denote the (random) number of times arm k has been pulled within the first t timesteps, and $\widehat{\mu}_t^k$ denote the sample average of those pulls. That is,

$$N_t^k := \sum_{\tau=0}^{t-1} \mathbf{1}\{a_\tau = k\}$$
$$\widehat{\mu}_t^k := \frac{1}{N_t^k} \sum_{\tau=0}^{t-1} \mathbf{1}\{a_\tau = k\} r_\tau.$$

To achieve the "fixed sample size" assumption, we'll need to shift our index from *time* to *number of samples from each arm*. In particular, we'll define $\widetilde{r}_n^k$ to be the nth sample from arm k, and $\widetilde{\mu}_n^k$ to be the sample average of the first n samples from arm k. Then, for a fixed n, this satisfies the "fixed sample size" assumption, and we can apply Hoeffding's inequality to get a bound on $\widetilde{\mu}_n^k$.

So how can we extend our bound on $\widetilde{\mu}_n^k$ to $\widehat{\mu}_t^k$? Well, we know $N_t^k \leq t$ (where equality would be the case if and only if we had pulled arm k every time). So we can apply the same trick as last time, where we uniform-ize across all possible values of N_t^k:

$$\mathbb{P}\left(\forall n \leq t, |\widetilde{\mu}_n^k - \mu^k| \leq \sqrt{\frac{\ln(2/\delta)}{2n}}\right) \geq 1 - t\delta.$$

In particular, since $N_t^k \leq t$, and $\widetilde{\mu}_{N_t^k}^k = \widehat{\mu}_t^k$ by definition, we have

$$\mathbb{P}\left(|\widehat{\mu}_t^k - \mu^k| \leq \sqrt{\frac{\ln(2t/\delta')}{2N_t^k}}\right) \geq 1 - \delta' \text{ where } \delta' := t\delta.$$

This bound would then suffice for applying the UCB algorithm! That is, the upper confidence bound for arm k would be

$$M_t^k := \widehat{\mu}_t^k + \sqrt{\frac{\ln(2t/\delta')}{2N_t^k}},$$

where we can choose δ' depending on how tight we want the interval to be. A smaller δ' would give us a larger yet higher-confidence interval, and vice versa. We can now use this to define the UCB algorithm.

Definition 1.6.1: Upper Confidence Bound (UCB)

Input: $\delta' \in (0, 1)$
for $t \leftarrow 0$ to $T - 1$ **do**
  $k \leftarrow \arg\max_{k' \in [K]} \frac{S^{k'}}{N^{k'}} + \sqrt{\frac{\ln(2t/\delta')}{2N^{k'}}}$
  $r_t \sim \nu^k$
  $S^k \leftarrow S^k + r_t$
  $N^k \leftarrow N^k + 1$
end for

Exercise: As written, this ignores the issue that we divide by $N^k = 0$ for all arms at the beginning. How should we resolve this issue?

Intuitively, UCB prioritizes arms where:

1. $\widehat{\mu}_t^k$ is large, i.e. the arm has a high sample average, and we'd choose it for *exploitation*, and
2. $\sqrt{\frac{\ln(2t/\delta')}{2N_t^k}}$ is large, i.e. we're still uncertain about the arm, and we'd choose it for *exploration*.

As desired, this explores in a smarter, *adaptive* way compared to the previous algorithms. Does it achieve lower regret?

1.6.1 UCB regret analysis

First we'll bound the regret incurred at each timestep. Then we'll bound the *total* regret across timesteps.

For the sake of analysis, we'll use a slightly looser bound that applies across the whole time horizon and across all arms. We'll omit the derivation since it's very similar to the above (walk through it yourself for practice).

$$\mathbb{P}\left(\forall k \leq K, t < T. |\widehat{\mu}_t^k - \mu^k| \leq B_t^k\right) \geq 1 - \delta''$$
$$\text{where} \quad B_t^k := \sqrt{\frac{\ln(2TK/\delta'')}{2N_t^k}}.$$

Intuitively, B_t^k denotes the *width* of the CI for arm k at time t. Then, assuming the above uniform bound holds (which occurs with probability $1 - \delta''$), we can bound the regret at each timestep as follows:

$$\begin{aligned}
\mu^\star - \mu^{a_t} &\leq \widehat{\mu}_t^{k^*} + B_t^{k^*} - \mu^{a_t} && \text{applying UCB to arm } k^\star \\
&\leq \widehat{\mu}_t^{a_t} + B_t^{a_t} - \mu^{a_t} && \text{since UCB chooses } a_t = \arg\max_{k \in [K]} \widehat{\mu}_t^k + B_t^k \\
&\leq 2B_t^{a_t} && \text{since } \widehat{\mu}_t^{a_t} - \mu^{a_t} \leq B_t^{a_t} \text{ by definition of } B_t^{a_t}
\end{aligned}$$

Summing this across timesteps gives

$$\text{Regret}_T \leq \sum_{t=0}^{T-1} 2B_t^{a_t}$$

$$
\begin{aligned}
&= \sqrt{2\ln(2TK/\delta'')} \sum_{t=0}^{T-1} (N_t^{a_t})^{-1/2} \\
\sum_{t=0}^{T-1} (N_t^{a_t})^{-1/2} &= \sum_{t=0}^{T-1} \sum_{k=1}^{K} \mathbf{1}\left\{a_t = k\right\} (N_t^k)^{-1/2} \\
&= \sum_{k=1}^{K} \sum_{n=1}^{N_T^k} n^{-1/2} \\
&\leq K \sum_{n=1}^{T} n^{-1/2} \\
\sum_{n=1}^{T} n^{-1/2} &\leq 1 + \int_1^T x^{-1/2}\, \mathrm{d}x \\
&= 1 + (2\sqrt{x})_1^T \\
&= 2\sqrt{T} - 1 \\
&\leq 2\sqrt{T}
\end{aligned}
$$

Putting everything together gives

$$
\begin{aligned}
\text{Regret}_T &\leq 2K\sqrt{2T\ln(2TK/\delta'')} \qquad \text{with probability } 1-\delta'' \\
&= \widetilde{O}(K\sqrt{T})
\end{aligned}
$$

In fact, we can do a more sophisticated analysis to trim off a factor of $\sqrt{K}$ and show $\text{Regret}_T \leq \widetilde{O}(\sqrt{TK})$.

1.6.2 Lower bound on regret (intuition)

Is it possible to do better than $\Omega(\sqrt{T})$ in general? In fact, no! We can show that any algorithm must incur $\Omega(\sqrt{T})$ regret in the worst case. We won't rigorously prove this here, but the intuition is as follows.

The Central Limit Theorem tells us that with T i.i.d. samples from some distribution, we can only learn the mean of the distribution to within $\Omega(1/\sqrt{T})$ (the standard deviation). Then, since we get T samples spread out across the arms, we can only learn each arm's mean to an even looser degree.

That is, if two arms have means that are within about $1/\sqrt{T}$, we won't be able to confidently tell them apart, and will sample them about equally. But then we'll incur regret

$$\Omega((T/2)\cdot(1/\sqrt{T})) = \Omega(\sqrt{T}).$$

1.7 Thompson sampling and Bayesian bandits

So far, we've treated the parameters $\mu^0, \dots, \mu^{K-1}$ of the reward distributions as *fixed*. Instead, we can take a **Bayesian** approach where we treat them as random variables from some **prior distribution**. Then, upon pulling an arm and observing a reward, we can simply *condition* on this observation to exactly describe the **posterior distribution** over the parameters. This fully describes the information we gain about the parameters from observing the reward.

From this Bayesian perspective, the **Thompson sampling** algorithm follows naturally: just sample from the distribution of the optimal arm, given the observations!

Definition 1.7.1: Thompson sampling

Input: the prior distribution $\pi \in \Delta([0,1]^K)$
for $t \leftarrow 0$ to $T-1$ **do**
  $\boldsymbol{\mu} \sim \pi(\cdot \mid a_0, r_0, \dots, a_{t-1}, r_{t-1})$
  $a_t \leftarrow \arg\max_{k \in [K]} \mu^k$
  $r_t \sim \nu^{a_t}$ ▷ Observe reward
end for

In other words, we sample each arm proportionally to how likely we think it is to be optimal, given the observations so far. This strikes a good exploration-exploitation tradeoff: we explore more for arms that we're less certain about, and exploit more for arms that we're more certain about. Thompson sampling is a simple yet powerful algorithm that achieves state-of-the-art performance in many settings.

Example 1.7.1: Bayesian Bernoulli bandit

We've often been working in the Bernoulli bandit setting, where arm k yields a reward of 1 with probability μ^k and no reward otherwise. The vector of success probabilities $\boldsymbol{\mu} = (\mu^1, \dots, \mu^K)$ thus describes the entire MAB.

Under the Bayesian perspective, we think of $\boldsymbol{\mu}$ as a *random* vector drawn from some prior distribution $\pi(\boldsymbol{\mu})$. For example, we might have π be the Uniform distribution over the unit hypercube $[0,1]^K$, that is,

$$\pi(\boldsymbol{\mu}) = \begin{cases} 1 & \text{if } \boldsymbol{\mu} \in [0,1]^K \\ 0 & \text{otherwise} \end{cases}$$

Then, upon viewing some reward, we can exactly calculate the **posterior** distribution of $\boldsymbol{\mu}$ using Bayes's rule (i.e. the definition of conditional probability):

$$\begin{aligned} \mathbb{P}(\boldsymbol{\mu} \mid a_0, r_0) &\propto \mathbb{P}(r_0 \mid a_0, \boldsymbol{\mu})\, \mathbb{P}(a_0 \mid \boldsymbol{\mu})\, \mathbb{P}(\boldsymbol{\mu}) \\ &\propto (\mu^{a_0})^{r_0} (1-\mu^{a_0})^{1-r_0}. \end{aligned}$$

This is the PDF of the Beta$(1+r_0, 1+(1-r_0))$ distribution, which is a conjugate prior for the Bernoulli distribution. That is, if we start with a Beta prior on μ^k (note that

$\text{Unif}([0,1]) = \text{Beta}(1,1)$), then the posterior, after conditioning on samples from $\text{Bern}(\mu^k)$, will also be Beta. This is a very convenient property, since it means we can simply update the parameters of the Beta distribution upon observing a reward, rather than having to recompute the entire posterior distribution from scratch.

It turns out that asymptotically, Thompson sampling is optimal in the following sense. Lai and Robbins [2] prove an *instance-dependent* lower bound that says for *any* bandit algorithm,

$$\liminf_{T\to\infty} \frac{\mathbb{E}[N_T^k]}{\ln(T)} \geq \frac{1}{\mathsf{KL}(\mu^k \parallel \mu^\star)}$$

where

$$\mathsf{KL}(\mu^k \parallel \mu^\star) = \mu^k \ln \frac{\mu^k}{\mu^\star} + (1-\mu^k) \ln \frac{1-\mu^k}{1-\mu^\star}$$

measures the **Kullback-Leibler divergence** from the Bernoulli distribution with mean μ^k to the Bernoulli distribution with mean $\mu^\star$. It turns out that Thompson sampling achieves this lower bound with equality! That is, not only is the error *rate* optimal, but the *constant factor* is optimal as well.

1.8 Contextual bandits

In the above MAB environment, the reward distributions of the arms remain constant. However, in many real-world settings, we might receive additional information that affects these distributions. For example, in the online advertising case where each arm corresponds to an ad we could show the user, we might receive information about the user's preferences that changes how likely they are to click on a given ad. We can model such environments using **contextual bandits**.

Definition 1.8.1: Contextual bandit

At each timestep t, a new *context* x_t is drawn from some distribution ν_{x}. The learner gets to observe the context, and choose an action a_t according to some context-dependent policy $\pi_t(x_t)$. Then, the learner observes the reward from the chosen arm $r_t \sim \nu^{a_t}(x_t)$. The reward distribution also depends on the context.

Assuming our context is *discrete*, we can just perform the same algorithms, treating each context-arm pair as its own arm. This gives us an enlarged MAB of $K|\mathcal{X}|$ arms.

Exercise: Write down the UCB algorithm for this enlarged MAB. That is, write an expression for $\pi_t(x_t) = \arg\max_a \ldots$.

Recall that running UCB for T timesteps on an MAB with K arms achieves a regret bound of $\widetilde{O}(\sqrt{TK})$. So in this problem, we would achieve regret $\widetilde{O}(\sqrt{TK|\mathcal{X}|})$ in the contextual MAB, which has a polynomial dependence on $|\mathcal{X}|$. But in a situation where we have large, or even

infinitely many contexts, e.g. in the case where our context is a continuous value, this becomes intractable.

Note that this "enlarged MAB" treats the different contexts as entirely unrelated to each other, while in practice, often contexts are *related* to each other in some way: for example, we might want to advertise similar products to users with similar preferences. How can we incorporate this structure into our solution?

1.8.1 Linear contextual bandits

We want to model the *mean reward* of arm k as a function of the context, i.e. $\mu^k(x)$. One simple model is the *linear* one: $\mu^k(x) = x^\top \theta^k$, where $x \in \mathcal{X} = \mathbb{R}^d$ and $\theta^k \in \mathbb{R}^d$ describes a *feature direction* for arm k. Recall that **supervised learning** gives us a way to estimate a conditional expectation from samples: We learn a *least squares* estimator from the timesteps where arm k was selected:

$$\widehat{\theta}_t^k = \arg\min_{\theta \in \mathbb{R}^d} \sum_{\{i \in [t] : a_i = k\}} (r_i - x_i^\top \theta)^2.$$

This has the closed-form solution known as the *ordinary least squares* (OLS) estimator:

$$\begin{aligned} \widehat{\theta}_t^k &= (A_t^k)^{-1} \sum_{\{i \in [t] : a_i = k\}} x_i r_i \\ \text{where} \quad A_t^k &= \sum_{\{i \in [t] : a_i = k\}} x_i x_i^\top. \end{aligned} \tag{1.4}$$

We can now apply the UCB algorithm in this environment in order to balance *exploration* of new arms and *exploitation* of arms that we believe to have high reward. But how should we construct the upper confidence bound? Previously, we treated the pulls of an arm as i.i.d. samples and used Hoeffding's inequality to bound the distance of the sample mean, our estimator, from the true mean. However, now our estimator is not a sample mean, but rather the OLS estimator above (1.4). Instead, we'll use **Chebyshev's inequality** to construct an upper confidence bound.

Theorem 1.8.1: Chebyshev's inequality

For a random variable Y such that $\mathbb{E}\, Y = 0$ and $\mathbb{E}\, Y^2 = \sigma^2$,

$$|Y| \leq \beta\sigma \quad \text{with probability} \geq 1 - \frac{1}{\beta^2}$$

Since the OLS estimator is known to be unbiased (try proving this yourself), we can apply Chebyshev's inequality to $x_t^\top(\widehat{\theta}_t^k - \theta^k)$:

$$x_t^\top \theta^k \leq x_t^\top \widehat{\theta}_t^k + \beta \sqrt{x_t^\top (A_t^k)^{-1} x_t} \quad \text{with probability} \geq 1 - \frac{1}{\beta^2}$$

Exercise: We haven't explained why $x_t^\top (A_t^k)^{-1} x_t$ is the correct expression for the variance of $x_t^\top \widehat{\theta}_t^k$. This result follows from some algebra on the definition of the OLS estimator (1.4).

The first term is exactly our predicted reward $\widehat{\mu}_t^k(x_t)$. To interpret the second term, note that

$$x_t^\top (A_t^k)^{-1} x_t = \frac{1}{N_t^k} x_t^\top (\Sigma_t^k)^{-1} x_t,$$

where

$$\Sigma_t^k = \frac{1}{N_t^k} \sum_{\{i \in [t] : a_i = k\}} x_i x_i^\top$$

is the empirical covariance matrix of the contexts (assuming that the context has mean zero). That is, the learner is encouraged to choose arms when x_t is *not aligned* with the data seen so far, or if arm k has not been explored much and so N_t^k is small.

We can now substitute these quantities into UCB to get the **LinUCB** algorithm:

Definition 1.8.2: LinUCB

Input: Regularization parameter $\lambda > 0$
Input: Confidence parameter $c : \mathbb{N} \to [0, \infty)$
for $t \in [T]$ **do**
  for $k \in [K]$ **do**
    $A_t^k \leftarrow \sum_{i=0}^{t-1} \mathbf{1}\{a_i = k\} x_i x_i^\top + \lambda I$
    $\theta_t^k \leftarrow (A_t^k)^{-1} \sum_{i=0}^{t-1} x_i r_i \mathbf{1}\{a_t = k\}$
  end for
  Given context x_t
  Choose $a_t = \arg\max_k x_t^\top \widehat{\theta}_t^k + c_t \sqrt{x_t^\top (A_t^k)^{-1} x_t}$
  Observe reward $r_t \sim \nu^{a_t}(x_t)$
end for

We include a λI regularization term to ensure that A_t^k is invertible. This is equivalent to solving a *ridge regression* problem instead of the unregularized least squares problem.

c_t is similar to the $\log(2t/\delta')$ term of UCB: It controls the width of the confidence interval. Here, we treat it as a tunable parameter, though in a theoretical analysis, it would depend on A_t^k and the probability δ with which the bound holds.

Using similar tools for UCB, we can also prove an $\widetilde{O}(\sqrt{T})$ regret bound. The full details of the analysis can be found in [1, Section 6.3].