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 Trade-off**: 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.

examples
nical trials,
ce, etc.

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 Multi-Armed Bandits

Remark 1.1.1: Namesake

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 $1, \dots, K$ and use *superscripts* to indicate the arm index; since we seldom need to raise values to a power, this 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 $\nu^k \in \triangle([0,1])$ with mean $\mu^k = \mathbb{E}_{r\sim\nu^k}[r]$.

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

```
# multi-armed bandits
for timestep in range(0, T):
    # Agent chooses an arm
    k = agent.choose_arm()

    # Environment generates a reward
    r = env.generate_reward(k)

    # Agent observes the reward
    agent.observe_reward(k, r)
```

Maybe switc
more traditi
pseudocode
Or set up so
Python "int
faces" near
start?

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 the difference between the total reward it observes and the total reward it *would* have received if it had always pulled the optimal arm:

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

Note that this depends on the *true means* of the pulled arms, *not* the observed rewards.

Often we consider the **expected regret** $\mathbb{E}[\text{Regret}_T]$, where the randomness comes

from the agent's strategy.

e Big O
ion in ap-
x

Ideally, we'd like to asymptotically achieve **zero regret**, i.e. $\mathbb{E}[\text{Regret}_T] = o(T)$.

1.1.1 Pure exploration (random guessing)

A trivial strategy is to always choose arms at random (i.e. "pure exploration"). What is the expected regret of this strategy?

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

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.

1.1.2 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.

```
# observed_rewards is an array of length K

# exploration phase
for k in range(K):
    observed_rewards[k] = env.generate_reward(k)
k_hat = argmax(observed_rewards)

# exploitation phase
for t in range(T - K):
    r = env.generate_reward(k_hat)
```

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 two arms, with Bernoulli reward distributions given by $\mu^1 > \mu^2$.

Let's let r^1 be the random reward from the first arm and r^2 be the random reward from the second. If $r^1 > r^2$, then we achieve zero regret. Otherwise, we achieve regret $T(\mu^1 - \mu^2)$. Thus, the expected regret is simply:

$$\begin{aligned}\mathbb{E}[\text{Regret}_T] &= \mathbb{P}(r^1 < r^2) \cdot T(\mu^1 - \mu^2) + c \\ &= (1 - \mu^1)\mu^2 \cdot T(\mu^1 - \mu^2) + c\end{aligned}$$

Which is still $\Theta(T)$, the same as pure exploration!

Can we do better? For one, we could reduce the variance of the reward estimates by pulling each arm *multiple times*. This is called the **explore-then-commit** strategy.

1.1.3 Explore-then-commit

Let's pull each arm N_{explore} times, and then commit to the arm with the highest observed average reward. What is the expected regret of this strategy?

```
# avg_reward is an array of length K

# exploration phase
for k in range(K):
    total = 0
    for i in range(N_explore):
        total += env.generate_reward(k)
    avg_reward[k] = total / N_explore
k_hat = argmax(avg_reward)

# exploitation phase
for t in range(T):
    r = env.generate_reward(k_hat)
```

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

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 ultimately prove that:

1. For any total time T,
2. We can choose N_{explore} such that
3. With arbitrarily high probability, the regret is sublinear.

Let $\hat{k} := \arg\max_{k \in [K]} \hat{\mu}^k$ be the arm we choose to exploit. We know the regret from the exploitation phase is

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

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

Let's use $\Delta^k = \hat{\mu}^k - \mu^k$ to denote how far the mean estimate for arm k is from the true mean. **Hoeffding's inequality** tells us that, for a given arm k, since the rewards from that arm are i.i.d.,

$$\mathbb{P}\left(|\Delta^k| > \sqrt{\frac{\ln(2/\delta)}{2N_{\text{explore}}}}\right) \leq \delta. \tag{1.2}$$

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

The **union bound** provides a simple way to do this: The probability of error (i.e. the l.h.s. of 1.2) for a *single* arm is at most δ, so the probability that *at least one* of the arms is far from the mean is at most $K\delta$. Setting $\delta' := K\delta$ and taking the complement of both sides, we have

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

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

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

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:

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

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

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

$$\text{Regret}_T \leq 3T^{2/3}\sqrt[3]{K\ln(2K/\delta')/2}$$

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.

1.1.4 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.

```
# epsilon-greedy
# random() samples from the uniform distribution on [0, 1]
for t in range(T):
    if random() < epsilon(t):
        # exploration
        k = random_choice(K)
    else:
        # exploitation
        # calculate averages using element-wise division
        k = argmax(total_reward / num_pulls)
    r = env.generate_reward(k)
    total_reward[k] += r
    num_pulls[k] += 1
```

Note that ϵ can vary over time. In particular we might want to gradually *decrease* ϵ as we learn more about the environment over time.

It turns out that setting $\epsilon_t = \sqrt[3]{K\ln(t)/t}$ also achieves a regret of $\widetilde{O}(t^{2/3}\sqrt[3]{K})$ (ignoring the logarithmic factors).

Insert optim epsilon and gret analysis high probab bound

In the ETC case, 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? In particular, what if we could explore more for arms that we're less certain about? This brings us to the **Upper Confidence Bound** (UCB) algorithm.

s revision
ructure

1.1.5 Upper Confidence Bound (UCB)

We'll estimate *confidence intervals* for the mean of each arm, 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, we'd like to compute some upper confidence bound M_t^k for arm k at time t and then choose $a_t := \arg\max_{k\in[K]} M_t^k$. But how should we compute M_t^k?

In our regret analysis for ETC, we were able to compute this bound using Hoeffding's inequality. Hoeffding's inequality assumes that the number of samples is *fixed*, which was true in ETC. However, 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 upper confidence bound that holds *uniformly* across time and across the different 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 $\hat{\mu}_t^k$ denote the sample average of those pulls. That is,

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

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 $\hat{\mu}_t^k$? Well, we know $N_t^k \leq t$ (which would be the case 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. In particular, we let $\delta' := t\delta$, giving us

$$\mathbb{P}\left(\forall n \leq t, |\widetilde{\mu}_n^k - \mu^k| \leq \sqrt{\frac{\ln(2t/\delta')}{2n}}\right) \geq 1 - \delta'$$

Now we can safely set $n := N_t^k$ to achieve

elaborate m on this

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

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

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

, where we can choose δ' depending on how tight we want the interval to be. A smaller δ' would give us a larger yet "more confident" interval, and vice versa.

Intuitively, this prioritizes arms where:

1. $\hat{\mu}_t^k$ 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?

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 similar to the above (walk through it yourself for practice).

$$\mathbb{P}\left(\forall k \leq K, t < T, |\hat{\mu}_t^k - \mu^k| \leq B_t^k\right) \geq 1 - \delta''$$

$$\text{where} \quad B_t^k := \sqrt{\frac{\ln(2TK/\delta'')}{2N_t^k}}.$$

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:

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

Summing this across timesteps gives

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

Putting everything together gives

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

e include? In fact, we can do a more sophisticated analysis to show $\text{Regret}_T = \widetilde{O}(\sqrt{TK})$.

1.2 Thompson sampling