CMSC 858G: Bandits, Experts and Games 09/12/16

Lecture 2: Multi-arm Bandits with i.i.d Rewards

Instructor: Alex Slivkins Scribed by: Amr Sharaf, Liqian Zhang

1 Administrivia

In the literature of bandits, experts, and games, there are hundreds of papers, and it will be impossible to cover everything in details. Thus, this course will cover some of the most interesting aspects of the field. We will cover one line of work every lecture, trying to cover the essential concepts, giving higher priority for more intiuitive results rather than complicated techniques.

2 Introduction

Last lecture we had a probability theory recap, including concentration of measure inequalities. Today, we're going to cover the basic model of multi-arm bandits with i.i.d rewards. We will introduce several techniques for solving the problem, and analyze there performance in terms of regret guarantees.

3 Multi-arm Bandits: Mathematical Model

In the multi-arm bandit problem with iid rewards, the learner selects an arm $a \in \mathcal{A}$ at every time step t. Learning proceeds in rounds, and we assume that the number of rounds is fixed, and indexed by $t = 1 \cdots T$. At each round, the algorithm chooses one action a_t (we'll use the arms and actions interchangeably to mean the same thing). After taking the action, a reward for this action is realized and observed by the algorithm. The process is repeated until the end of time horizon T is reached. The goal of the algorithm is to gather as much commulative reward as possible.

It's important to emphasize that the algorithm observes only the reward for the selected action, not all the other actions that could have been selected, that is why it is called a bandit feedback setting.

There is also the iid assumption, where the reward for each action is assumed be i.i.d (independent and identically distributed). More precisely, for each action a, there is a distribution $\mathcal{D}_a$ over real numbers (called "reward distribution"); for simplicity, all rewards will be in the interval $[0, 1]$. Every time this action is chosen, the reward is sampled independently from this distribution. It is crucial that this distribution is unknown to the algorithm, and does not change over time.

Notation: the mean (expected) reward of action a is denoted $\mu(a)$; the time horizon is T; the number of arms is K. Actions are denoted a. Let's use these conventions throughout the course.

Perhaps the simplest reward distribution is the Bernoulli distribution, when the reward of each arm a can be either 0 or 1. ("Success or failure", "heads or tails".) This distribution is parameterized by the probability of success $\mu(a)$, which also defines the mean of the distribution. Note that the problem instance is then completely described by the time horizon T and the vector $(\mu(a) : a \in \mathcal{A})$.

3.1 Some (stylized) examples for MAB with IID rewards

1. **News**: in a very stylized news application, a user visits a news site, the site presents it with a header for an article, and a user either clicks on this header or not. The goal of the website is to maximize #clicks. So each possible header is an arm in a bandit problem, and clicks are the rewards. Note that the 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. **Selecting Ads**: In website advertising, a user visits a webpage, and a learning algorithm has to select one of many possible ads to display. In this setting, each ad can be considered an arm. The ad is displayed, it is observed whether or not the user clicks on the ad. If the ad a is clicked, the advertiser pays some amount $v_a \in [0,1]$ which is fixed in advance and known to the algorithm. (This amount depends only on the chosen ad, but does not change over time.) So the paid amount is considered as the observed reward.

   Note that here rewards can be arbitrary numbers, but for each arm a, the reward can take only two values: 0 or v_a.

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.

   Note that for each arm, the reward can, in principle, take arbitrarily many different values.

3.2 Regret

While algorithm's goal is to maximize reward, we have to be more precise about defining what it means. There are several notions for defining reward maximization. One standard notion is **regret**; we will use this notion (or versions thereof) throughout most of this course.

To define regret, we will look at two quantities: the reward accumulated by the best arm, and the average reward accumulated by the algorithm, we define the difference between the two quantities to be the regret:

$$R(t) = \mu^* \times t - \sum_{s=1}^{t} \mu(a_s), \tag{1}$$

where $R(t)$ is the regret after t-time steps, and $\mu^* = \max_{a \in \mathcal{A}} \mu(a)$ is the expected reward for the best arm. Note that the arm a_t chosen by the algorithm is a random quantity, as it may depend on the (random) rewards and also possibly on the internal randomness of the algorithm. Hence, we will typically talk about "expected regret" $\mathbb{E}[R(T)]$.

So why is it called regret? Because it is how much the algorithm "regrets" not knowing what is the best arm!

Remark 3.1. One can view the $\mu^* \times t$ term as a 'benchmark' to which the performance of the algorithm is compared. Hence, it is sometimes called the "best arm" benchmark. While it is the

standard benchmark in most work on bandits, in some settings other benchmarks make sense (and sometimes make *more* sense).

Remark 3.2 (Terminology). Since our definition of regret sums over all rounds, we will sometimes call it *cumulative* regret. When/if we need to highlight the distinction between $R(T)$ and $\mathbb{E}[R(T)]$, we will say *realized regret* and *expected regret*; but most of the time, we will just say "regret" and the meaning will be clear from the context. The quantity $\mathbb{E}[R(T)]$ is sometimes called *pseudo-regret* in the literature.

Regret can depend on many different parameters, we care mainly about the dependence on the time horizon, number of arms, and the average reward for each arm (μ). We will usually use big-O notation to focus on the growth rate dependence on the different parameters instead of keeping track of all the constants.

4 Multi-arm bandit Algorithms

In this section, we cover several algorithms for solving the multi-arm bandit problem. For the simplicity of presentation, we start with the simple case where we have only two arms with zero/one rewards. Later, we extend the algorithms for more than two arms with bounded rewards.

4.1 Explore-First

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

Algorithm 1: Explore-First

Algorithm 1 describes the explore-first algorithm: we explore for N time steps, and then select the arm with the highest average reward. The parameter N is fixed in advance; it will be chosen later in the analysis as function of the time horizon and #arms. In the remainder of this subsection, we analyze the algorithm in terms of regret guarantees.

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 from last lecture 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:

$$\Pr\left\{|\bar{\mu}(a) - \mu(a)| \leq r(a)\right\} \geq 1 - \frac{1}{T^4} \tag{2}$$

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

Remark 4.1. 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) 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 with a more technical proof that argues about probabilities more carefully.

Let us start with 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 we choose the other arm $a \neq a^*$. But why did we choose arm 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

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

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

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

Since we can select any value for N (as long as it is known to the algorithm before the first round), we can optimize the right-hand side to get the tightest upper bound. Noting that the two summands are, resp., monotonically increasing and monotonically decreasing in N, we set N so that they are (approximately) equal. For $N = T^{2/3}$, we get the following:

$$\begin{aligned} R(T) &\leq T^{2/3} + O(\sqrt{\frac{\log T}{T^{2/3}}} \times T) \\ &\leq O(\sqrt{\log T} \times T^{2/3}). \end{aligned}$$

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

$$\begin{aligned} \mathbb{E}[R(T)] &= \mathbb{E}[R(T)|\text{clean event}] \times \Pr[\text{clean event}] \ + \ \mathbb{E}[R(T)|\text{bad event}] \times \Pr[\text{bad event}] && (3) \\ &\leq \mathbb{E}[R(T)|\text{clean event}] + T \times O(T^{-4}) && (4) \\ &\leq O(\sqrt{\log T} \times T^{2/3}). && (5) \end{aligned}$$

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

For $K > 2$ arms, we have to apply the union bound for (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, the confidence radius is now $r(a) = \sqrt{\frac{2\log T}{N/K}}$. Working through the proof, we obtain $R(T) \leq N + O(\sqrt{\frac{\log T}{N/K}} \times T)$. Plugging in $N = T^{2/3} \times O(K \log T)^{1/3}$, and completing the proof same way as in (3), we obtain:

Theorem 4.2. *Explore-first algorithm achieves regret* $\mathbb{E}[R(T)] \leq T^{2/3} \times O(K \log T)^{1/3}$*, where* K *is the number of arms.*

4.2 Epsilon Greedy

One problem with Explore-first is that the "losses" are concentrated in the initial exploration phase. It may be better to have a more uniform exploration over time. This is done in the epsilon-greedy algorithm.

```
input : Exploration probability ε
Toss a coin with probability of success = ε;
if success then
    explore: choose an arm uniformly at random
else
    exploit: choose the arm with the highest average reward so far
```

Algorithm 2: Epsilon-Greedy

Note that the exploration is uniform, which is similar to the "round-robin" exploration in explore-first. 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 analysis for this algorithm may appear on the homework.

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 class, we will talk about two algorithms that implement adaptive exploration and achieve better regret.