CMSC 858G: Bandits, Experts and Games 10/17/16

Lecture 7: Full feedback and adversarial rewards (part I)

Instructor: Alex Slivkins Scribed by: Khanh Nguyen

In today lecture, we will shift our focus from problems with bandit feedback to problems with full feedbacks and adversarial rewards. In a full-feedback setting, in the end of each round, we observe rewards of not only the arm we chose to play, but of all other arms as well.

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

1. Algorithm picks arm $a \in [K]$.
2. Algorithm incurs cost $c_t(a)$ for the chosen arm.
3. Also, the costs of all arms, $c_t(a): a \in [K]$, are revealed.

Multi-armed bandits with full feedback

A real-life example is the investment problem. Each morning, we choose a stock to invest. At the end of the day, we observe not only the price of our target stock but prices of all stocks. Based on this kind of "full" feedback, we determine which stock to invest for the next day.

A motivating special case of "bandits with full feedback" can be framed as a question-answering problem with experts advice. Concretely, suppose we are assisted with a committee of experts to answer questions. In each round, a question arrives and each expert suggests an answer for it. In the stock example, a question would be "which stock to invest today?"; in binary prediction with experts advice, typical questions are "which label to assign to this example?". After receiving the question, we (or our algorithms) listen to the experts and pick an answer to respond. We then observe the correct answer and costs (penalties) of all other answers. Our goal is to do approximately as well as the best expert.

Such a process can be described by the following protocol:

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

1. Question x_t arrives.
2. K experts give answers $z_{1,t}, \ldots, z_{K,t}$.
3. Algorithm picks expert i.
4. Correct answer y_t^* is observed.
5. Also, the costs for all submitted answers, $c(z_{j,t}, y_t^*)$, are revealed.

Q&A with expert advice

Because of this motivation, the bandit problem with full feedback is also called the *best-expert problem*. Since in the context of Q&A it is natural to talk about *costs* (as penalties for incorrect answers), full-feedback bandit problems are usually defined in terms of *costs* rather than *rewards*. We will follow this convention, too.

1 Full feedback and i.i.d. rewards

When costs of all arms are realized, there is no need to explore. If we apply a naive strategy such as playing arm with the lowest average cost, one can prove that this strategy achieves an upper regret bound of $O\left(\sqrt{T\log(KT)}\right)$ and a lower regret bound of $\Omega(\sqrt{T})$. The upper bound can be proved by a simple application of clean event/confidence radius technique that we have seen in previous lectures. The lower bound follows from the same argument as the bandit lower bound for 2 arms (it turns out we can just assume full feedback in that proof).

Remark 1.1. The $\sqrt{T}$ is still present in the upper and lower bounds, even with full feedback. We will later see a simple special case with a lower bound $\Omega(\min(T, \log K))$. So the best lower bound we have for full feedback and i.i.d. rewards can be summarized as $\Omega(\sqrt{T}, \min(T, \log K))$.

2 Full feedback and adversarial rewards

What does "adversarial rewards" mean? It does not neccessarily mean that there is a real adversary that tries hard to make our algorithm fail. This term denotes situations when the costs change over time. For example, at a news website, everyday an algorithm decides which news to show to the users. At some point, since people are mostly interested in the presidential election, the cost of showing election may be low. However, in the future, people may be attracted to other events (e.g. the Olympics), the cost of the election will get higher and the cost of the more interesting events will become lower.

In general, the problem can be casted as playing a game with the adversary:

Each round $t \in [T]$:

1. Adversary chooses costs $c_t(a)$ for each arm $a \in [K]$.
2. Algorithm picks arm a_t.
3. Algorithm receives cost $c_t(a_t)$ for the chosen arm.
4. Algorithm observes $c_t(a)$ for all arms a.

Best-expert problem with adversarial rewards.

An adversary is called *oblivious* if the chosen costs at each round t do not depend on the algorithm's choices before round t. Otherwise, it is called *adaptive*. Also, an adversary is called *randomized* if in each round it chooses distribution over the cost vectors $(c_t(a) : a \in [K])$, and then draws the cost vector independently from this distribution.

The total cost of each arm a is defined as $\texttt{cost}(a) = \sum_{t=1}^{T} c_t(a)$. Intuitively, the "best arm" is an arm with a lowest cost. However, defining the "best arm" and "regret" precisely is a little tricky, depending on the "type" of an adversary:

- *Deterministic-oblivious adversary*: equivalently, the entire "cost table" $c_t(a)$ for all arms and all rounds is chosen before the first round. Then, naturally, the best arm is defined as $a^* \in \text{argmin}_{a\in[K]} \texttt{cost}(a)$, and regret is defined as

  $$R(T) = \texttt{cost}(ALG) - \min_{a\in[K]} \texttt{cost}(a), \tag{1}$$

  where $\texttt{cost}(ALG)$ denotes the total cost incurred by the algorithm.

  One drawback of considering this adversary is that it does not model i.i.d. costs (because there is no randomness in costs), whereas we'd like to think if i.i.d. rewards as a simple special case of adversarial rewards.

- *Randomized-oblivious adversary*: equivalently, the adversary fixes a distribution $\mathcal{D}$ over the cost tables before round one, and then draws a cost table from this distribution. Thus, $\texttt{cost}(a)$, the total cost of each arm a, is a random variable whose distribution is specified by $\mathcal{D}$. Accordingly, there are two natural ways to define the "best arm":

  - $\text{argmin}_a \texttt{cost}(a)$: this is the best arm *in hindsight* – after all costs have been observed. It is a natural notion of the best arm if we start from the deterministic-oblivious adversary.
  - $\text{argmin}_a \mathbb{E}[\texttt{cost}(a)]$: this is be best arm *in foresight* – and arm that you'd pick before round 1 if you only know the distribution $\mathcal{D}$ and you need to pick one arm to play in all rounds. This is a natural notion of "best arm" if we start from i.i.d. costs.

  Accordingly, one there are two natural notions of regret: w.r.t. the best-arm-in-hindsight (as in (1)), and w.r.t. best-arm-in-foresight:

  $$R(T) = \texttt{cost}(ALG) - \min_{a\in[K]} \mathbb{E}[\texttt{cost}(a)], \tag{2}$$

  (We used the second notion for bandits with i.i.d. rewards/costs.) We will refer to them as *hindsight regret* and *foresight regret*, respectively.[1]

  Foresight regret cannot exceed hindsight regret (because the best-arm-in-foresight is a weaker benchmark), and sometimes it allows for much stronger positive results. In particular, the $\log(T)$ regret bounds for i.i.d. rewards/costs do not extend to foresight regret.

- *Adaptive adversary* models scenarios when algorithm's actions may alter the environment that the algorithm operates in. For example:

  - an algorithm that adjusts the layout and text formatting of a website may cause users to permanently change their behavior, e.g., users may gradually used to a new design, and get dissatisfied with the old one.

[1]In the literature, the hindsight regret is usually referred to as *regret*, while the foresight regret is referred to as *weak regret* or *pseudo-regret*.

- Consider a bandit algorithm that selects news articles to show to users. A change in how the articles are selected may attract some users and repel some others, or perhaps cause the users to alter their reading preferences.
- if a dynamic pricing algorithm offers a discount on a new product, it may cause many people to buy this product and (eventually) grow to like it and spread the good word. Then more people would be willing to buy this product at full price.
- if a bandit algorithm adjusts the parameters of a repeated auction (e.g., a reserve price), auction participants may adjust their behavior over time, as they become more familiar with the algorithm.

Usually adaptive adversary is used to model non-malicious environment that adjusts itself in response to the algorithm's actions, rather than an actual adversary that strives to hurt us.

An adaptive adversary is assumed to be randomized by default, because the adversary can adapt the costs to the algorithm's choice of arms in the past, and the algorithm is usually randomized. Thus, the distinction between foresight and hindsight regret comes into play.

More importantly, which arm is best may depend on the algorithm's actions. For example, an algorithm that always chooses arm 1 may see that arm 2 is consistently much better, whereas *if the algorithm always played arm 2*, arm 1 may have been better. In this course we use a simple notion: best-in-hindsight-arm according to the costs as they are actually observed by the algorithm; we call it the *best observed arm.*

There are three good reasons to consider this notion. First, this notion *is* meaningful in some scenarios (but not in some others). Second, results for adaptive adversary and best observed arm often follow from the analysis of oblivious adversary with very little extra work. Third, best-realized-arm analysis of an adaptive adversary may be used as a tool to prove a statement about oblivious adversaries (e.g., next lecture).

3 Majority vote algorithm

We consider *binary prediction with experts advice*, a special case of "Q&A with expert advice", where the expert answers $z_{i,t}$ are binary. For example: is this image a face or not? Is it going to rain today or not?

Let us start with a simple example when there exists a *perfect expert* who never makes a mistake. We propose the following algorithm, called the *majority vote algorithm*:

In round t, pick the action chosen by the majority of the experts.

Theorem 3.1. *Consider binary prediction with experts advice. Assuming a perfect expert, the majority vote algorithm makes at most $\log_2 K$ mistakes, where K is the number of experts.*

Proof. Let S_t be the set of experts who make no mistakes up to the beginning of round t.

Let $W_t = |S_t|$. Since the perfect expert is always in S_t, $W_t \geq 1$.

If the algorithm makes a mistake at round t, then $W_{t+1} \leq W_t/2$ because the majority is wrong and thus excluded from S_t. Since $W_t \geq 1$, the algorithm cannot make more than $\log_2 K$ mistakes. □

While this proof is very simple, it introduces a general technique that will be essential in the subsequent proofs:

- Define a quantity W_t which measures the total remaining amount of "credibility" of the the experts. Make sure W_t decreases by definition as experts make mistakes (incur costs).
- In the analysis, connect W_t with the behavior of the algorithm. First, prove that W_t decreases by a constant factor whenever the algorithm makes mistake (incurs a cost). Second, derive a lower bound on W_T from the existence of a "good expert".

One can prove a simple lower bound for this problem:

Theorem 3.2. *Consider binary prediction with experts advice. Assume a perfect expert. For any algorithm, there is a problem instance such that the algorithm makes at least $\Omega(\min(T, \log K))$ mistakes, where T is the time horizon and K is the number of experts.*

Proof Sketch. Assume $\log_2 K \in \mathbb{N}$. For the first $\log_2 K$ rounds, consider all possible sequences of predictions, create one expert for each prediction. Pick the perfect expert at random. □