ECE 8803: Online Decision Making in Machine Learning **Fall 2022**

Lecture 15: October 20

Lecturer: Bhuvesh Kumar

In today's lecture, we will switch gears to our next course module, which introduces the setting of learning with *limited-information-feedback*. In contrast to the previous module, we will now focus on stochastic (independent and identically distributed data). Despite the setting seemingly being easier, we will show that limited-information feedback brings new challenges to decision-making, even when the data is stochastic.

15.1. Limited-information v.s. full-information feedback

Until now, we received *full-information feedback* at every decision-making round. In other words, we receive not only the actual loss that we incur, but we can also reason about the losses that we would have incurred had we been able to make other decisions instead. For example:

- In sequence prediction, we made a prediction $\widehat{X}_t$ at round t, observed the sequence X_t, and incurred a loss $\ell(\widehat{X}_t, X_t)$. Because we observe the sequence value X_t after prediction (and not just the loss), we could reason about the prediction that we should have made instead at round t.

- In online linear optimization, we made a prediction $\boldsymbol{w}_t$ at round t and observed the *full loss vector* $\boldsymbol{\ell}_t$. Thus, not only can we infer the actual loss we incurred, given by $\langle \boldsymbol{w}_t, \boldsymbol{\ell}_t \rangle$, but we can infer the loss we would incur for any other decision $\boldsymbol{w}$ that we may have made.

- In online convex optimization, we made a prediction $\boldsymbol{w}_t$ and observed the *full loss function* $\ell(\cdot)$. Thus, not only can we infer the actual loss we incurred, given by $\ell(\boldsymbol{w}_t)$, but we can also infer the loss we could incur for any other decision $\boldsymbol{w}$ that we may have made.

It is worth noting that this ability to immediately look back and infer the loss we could have incurred in hindsight had we picked a different decision is critically used in *all* of the algorithms that we have considered thus far. For example, FTL picks the decision that *would have* minimized total running loss until now (in all cases). And we have seen that MWA/FTL/FTPL all use this principle from FTL to a partial extent, but add regularization of some kind to ensure stability against a possibly adversarial environment.

In contrast, in the *limited-information-feedback* setting we only observe the actual loss we incur at that round. For example:

Table 15.1: Comparison between full-information and limited-information feedback

Decision problem	Full-inf. feedback at round t	Limited-inf. feedback at round t
Sequence prediction	X_t and $\ell(\widehat{X}_t, X_t)$	$\ell(\widehat{X}_t, X_t)$ only
OLO	$\boldsymbol{\ell}_t$ and $\langle \boldsymbol{\ell}_t, \boldsymbol{w}_t \rangle$	$\boldsymbol{\ell}_t$ only
OCO	$\ell_t(\cdot)$ and $\ell_t(\boldsymbol{w}_t)$	$\ell_t(\cdot)$ only

- In sequence prediction, we would only observe $\ell(\widehat{X}_t, X_t)$ without observing X_t. Consider an example of *ternary* sequence prediction with 0-1 loss, where the sequence can take values in $\{0, 1, 2\}$. Suppose that we predict $\widehat{X}_t = 0$ at round t. Then, if we incur a loss of 0, we would know that $X_t = 0$; however, if we incur a loss of 1, the sequence could either be $X_t = 1$ or $X_t = 2$. We would know which of the two it is. As we will see shortly, this limited feedback makes a stark difference to decision-making even in the stochastic case.

- In online linear/convex optimization, we will only observe $\ell_t(\boldsymbol{w}_t)$, not the loss vector/function itself. This means that we cannot reason about the loss we would have made had we chosen a different decision $\boldsymbol{w}$ in hindsight.

This distinction between limited-information and full-information feedback is summarized in Table 15.1. Motivated by the real-world applications of limited-information feedback, we will consider the flipped problem of *reward maximization* for the rest of this course.

15.1.1 A real-world example

There are several real-world examples in which this kind of limited-information feedback model persists. One of the historical examples involves *clinical trials* for drug discovery. Suppose that we have two candidate drugs, Drug A and Drug B, that we want to decide to recommend for treatment of a disease. We might have some initial indication of their theoretical potential, but we now need to test their efficacy in practice. The way we will do this is by picking a different drug to test on a different patient at regular intervals, and use the data collected to report the superior drug.

Concretely, this becomes an online decision-making problem where "rounds" represent patients, and "actions" represent the drug of choice. In particular, we choose to administer drug (take action) $A_t = A$ or $A_t = B$ to patient (round) t based on past observations. After this decision, we receive a *reward*, denoted by G_{t,A_t}, which reflects how effective the drug was on patient t.

This application of drug discovery also highlights *why* this problem is online in nature: each of our data points is a human patient, and so the setting is very high-stakes. In traditional, low-stakes ML, we have the luxury of a large amount of training data that we can, at least initially, make several errors on. Here, we do not have this luxury: a more natural goal is to try and *maximize* the long-term reward over all patients. This is the central objective of the *multi-armed bandit* problem, that we now introduce formally.

15.1.2 The multi-armed bandit problem

The multi-armed bandit problem is a classic online decision-making problem that involves K actions or "arms" of choice at every round. Specifically, at round t we pick (a possibly randomized) action A_t, and receive the reward given by G_{t,A_t}. Importantly, we will assume that the rewards corresponding to each action are independently and identically distributed. In other words, $G_{t,a}$ will be iid across $t \geq 1$ for each value of $a \in \{1, \ldots, K\}$. Note that this assumption is well-motivated in the context of our drug discovery example: each drug will have similar efficacy on each patient, but there is also random noise that influences the efficacy from patient-to-patient. We denote the mean of the reward corresponding to action a by μ_a, i.e. $\mathbb{E}[G_{t,a}] = \mu_a$ for all values of t. *Of course, we do not know these means in advance: our goal is to learn them.*

Our goal is maximize the expected long-term reward over T rounds, given by:

$$\mathbb{E}[G_T] := \mathbb{E}\left[\sum_{t=1}^{T} G_{t,A_t}\right].$$

Just like in the online prediction/optimization module, the natural question arises of how to benchmark our performance. This actually turns out to have a very intuitive answer in the stochastic setting that we consider. Recall that if we knew the mean rewards $\{\mu_a\}$ in advance, our choice would be very easy: we would simply pick the action a^* that maximizes mean reward, i.e. $a^* = \arg\max_{a \in \{1,\ldots,K\}} \mu_a$. The expected reward that we would have gotten had we picked this action on all T rounds would be $T\mu_{a^*}$. For the drug discovery example, this benchmark would represent how well we could have done if we had consistently picked the better of the two drugs for the population.

This forms a natural benchmark for our overall performance, and our learning objective is to get as close to it as possible. Accordingly, all of the performance guarantees we will derive will minimize a quantity called *pseudo-regret*:

$$\overline{R}_T := T\mu_{a^*} - \mathbb{E}[G_T].$$

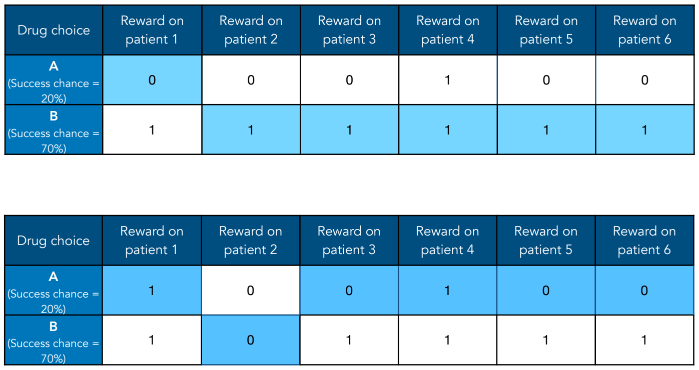
15.2. Challenges in limited-information feedback

Despite the seeming simplicity of this stochastic setting, the limited-information nature of the feedback poses foundational challenges. We now describe the crux of these challenges through our simple drug-discovery problem (which is an example of the 2-armed bandit problem).

15.2.1 The perils of *pure exploitation*

First, we consider the first and most intuitive choice for decision-making, which is akin to FTL. We consider each action, look at the previous rounds on which we sampled it, and calculate the average reward that was obtained. Then, we simply pick the action whose average reward was the largest. Concretely, at every round we can define these *sample means*

$$\widehat{\mu}_{a,t} := \frac{\sum_{s=1}^{t-1} \mathbb{I}[A_s = a] G_{s,a}}{\sum_{s=1}^{t-1} \mathbb{I}[A_s = a]}, \tag{15.1}$$



Drug choice	Reward on patient 1	Reward on patient 2	Reward on patient 3	Reward on patient 4	Reward on patient 5	Reward on patient 6
A (Success chance = 20%)	0	0	0	1	0	0
B (Success chance = 70%)	1	1	1	1	1	1

Drug choice	Reward on patient 1	Reward on patient 2	Reward on patient 3	Reward on patient 4	Reward on patient 5	Reward on patient 6
A (Success chance = 20%)	1	0	0	1	0	0
B (Success chance = 70%)	1	0	1	1	1	1

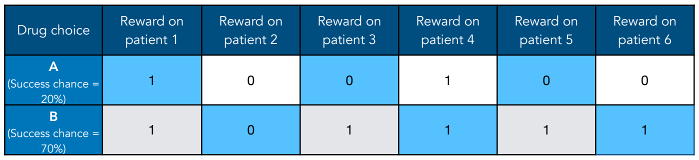
Figure 15.1: Depiction of the drugs that would be chosen by a “greedy” strategy on two possible realizations of the rewards of drugs A and B. Blue denotes the identity of the drug that was picked on each patient.

and we can simply pick A_t as the action that maximizes $\widehat{\mu}_{a,t}$. On the first K rounds, we simply pick the actions in a round-robin fashion so that we have one sample of each.

Clearly, this strategy has an intuitive interpretation that is similar to Follow-the-Leader (where we now normalize by the number of times each action was picked, as we are not able to observe samples of the reward on the other rounds). It turns out to be highly suboptimal in the limited-information feedback setting for entirely different reasons. The principal issue is that the randomness in initial rounds can influence this “greedy” decision-making long-term. Consider how greedy decisions would be made in our drug discovery example. Figure 15.1 illustrates an example where Drug A has 20% treatment efficacy, while Drug B has 70% treatment efficacy. The rewards that would be observed on each round would be the indicator of whether the respective drug was effective or not, and these are iid Bernoulli random variables (with success probability 0.2 and 0.7 respectively). Clearly, we would ideally like to converge towards selecting Drug B eventually. However, we can see from Figure 15.1 that whether the greedy algorithm actually does this depends on how the rewards *on the first two rounds* were realized. The top panel shows a favorable situation under which Drug A is not effective but Drug B is, consistent with the actual success probabilities. The bottom panel, on the other hand, shows a flipped situation under which *we got unlucky with Drug B, and lucky with Drug A*. This initial unluckiness leads the greedy algorithm to *always* pick Drug A thereafter, which is obviously a very undesirable outcome. It is a worthwhile exercise to verify this behavior for yourself and internalize it.

One might hope that the bottom panel situation is unlikely. While it is less likely than the top panel situation, it turns out to occur with a constant probability. Thus, in the formal multi-armed bandit framework with K arms, the greedy strategy can be shown to accumulate pseudo-regret that is *linear* in T, which is a highly suboptimal guarantee.

15.2.2 The perils of *pure exploration*



Drug choice	Reward on patient 1	Reward on patient 2	Reward on patient 3	Reward on patient 4	Reward on patient 5	Reward on patient 6
A (Success chance = 20%)	1	0	0	1	0	0
B (Success chance = 70%)	1	0	1	1	1	1

Figure 15.2: Depiction of the drugs that would be chosen by a uniformly exploring strategy on the second realization of the rewards of drugs A and B. Blue denotes the identity of the drug that was picked on each patient.

The exercise above showed us that it is essential to collect sufficiently many samples of each action before making a decision. The other extreme in decision-making would be to simply pick an action uniformly at random at every round. This could involve picking A_t according to the uniform distribution over all K arms, or in a round-robin fashion (i.e. not just in the first K rounds).

It is somewhat easier to see that this type of "pure exploration" strategy is also highly suboptimal, as it consistently keeps sampling the suboptimal actions, even after we have learned that they are suboptimal. Figure 15.2 depicts what would happen as a result of it in the drug discovery example. It is a worthwhile exercise to verify that the pseudo-regret incurred on this example would be equal to $0.5(0.7T - 0.2T) = 0.25T$. Thus, the pure exploration strategy also incurs linear in T pseudo-regret, but for a different reason.

15.3. Heuristics that trade off exploration and exploitation

The example above shows that strategies of *pure exploitation* and *pure exploration* both suffer a linear pseudo-regret, but for different reasons. To be able to do better than this, we will need to trade off exploration and exploitation in our decision-making. We will examine principled ways of doing this next lecture, but in the meantime present two heuristics that incorporate aspects of exploration and exploitation in decision-making.

15.3.1 ϵ-greedy

One natural way to incorporate both exploration and exploitation in decision-making is to pick the greedy action with some probability, and randomize otherwise. This leads to the popular ϵ-greedy algorithm, whose choice is formally described below. At every round t, for some randomization parameter ϵ_t, we pick:

$$A_t = \begin{cases} \arg\max_a [\widehat{\mu}_{a,t}] \text{ with probability } 1 - \epsilon_t \\ \text{uniformly at random with probability } \epsilon_t. \end{cases}$$

The choice of the parameter ϵ_t is critical here, as it determines to what extent we will randomize at time t. Intuitively, we would want to randomize less after we have played more rounds, as we may have seen more samples by then and would like to trust the greedy decision more. Accordingly, ϵ-greedy is usually run with ϵ_t being a decaying sequence in t. In the upcoming HW, you will explore one such choice of decaying sequence.

Intuitively, incorporating randomization allows us to fix some of the earlier issues in the greedy strategy being led onto the wrong track in the long run: the randomization ensures that we will sample both options sufficiently often initially, allowing us to get a true measure of their performance.

15.3.2 Explore-then-commit

Another heuristic that trades off exploration and exploitation is the *explore-then-commit* (ETC) algorithm. This algorithm simply samples all arms uniformly for $T_0 < T$ rounds, and thereafter picks the greedy action. Of course, the choice of the exploration period T_0 critically matters: if T_0 is too small, the algorithm will behave like the greedy algorithm; and if T_0 is too large, the algorithm will perform pure exploration.

It turns out that the ϵ-greedy and ETC, with respective appropriate choices of hyperparameter tuning, can get us a pseudo-regret guarantee that scales as $\mathcal{O}(T^{2/3})$. In fact, as we will see next lecture, we can design more principled approaches that trade off exploration and exploitation, and do even better.

15.4. Understanding pseudo-regret

Our goal is maximize the expected long-term reward over T rounds, given by:

$$\mathbb{E}[G_T] := \mathbb{E}\left[\sum_{t=1}^{T} G_{t,A_t}\right].$$

It is important to understand why we are taking an expectation here: it is two-fold:

1. For one, the rewards themselves are random, i.e. $\mathbb{E}[G_{T,a}] = \mu_a$ at any round t and for any action $a \in 1, \cdots, K$. Recall that these rewards in our drug example denoted the effect of a drug on a patient which we model as a random variable. Since all patients are from the same population, the rewards are iid.

2. For another, the actions A_t that are taken at time t in general will depend on the realizations of the rewards in past rounds. This means that the actions A_t will also be random variables that depend on past outcomes (although they are often deterministic conditioned on the past)

Recall that if we knew the mean rewards $\{\mu_a\}$ in advance, our choice would be very easy: we would simply pick the action a^* that maximizes mean reward, i.e. $a^* = \arg\max_{a\in\{1,\ldots,K\}} \mu_a$. The expected reward that we would have gotten had we picked this action on all T rounds would be $T\mu_{a^*}$. This leads to the quantity called *pseudo-regret*:

$$\overline{R}_T := T\mu_{a^*} - \mathbb{E}[G_T].$$

Note the "bar" that we have placed on top of the pseudo-regret quantity in order to distinguish it from the notions of regret that we studied previously in the class for online prediction and optimization with full-information feedback. In the stochastic MAB problem that we are studying, the benchmark is a learning-based one: in particular, how well we could have done had we known the means $(\mu_1, \cdots, \mu_K)$ beforehand (or just the identity of the optimal arm).

It turns out to be convenient to write the pseudo-regret directly in terms of the number of times each of the suboptimal arms is sampled. Let us say that our choice of algorithm sampled a suboptimal arm, denoted by a, $N_a(T)$ times, where $N_a(T)$ is a random variable that depends on the actual outcomes of the rewards. (For example, recall that in our drug discovery example from last time, $a = 1$ was the suboptimal arm. We saw that the greedy algorithm could yield two very different outcomes: one where we had $N_a(T) = 1$, and the other where we had $N_a(T) = T - 1$!). Concretely, we can show the following result:

Definition 1 *Let $\Delta_a := \mu^* - \mu_a$ for each suboptimal arm $a \neq a^*$ be the suboptimality gap. Then, the pseudo-regret of any algorithm can be equivalently written as:*

$$\overline{R}_T = \sum_{a \neq a^*} \Delta_a \cdot \mathbb{E}\left[N_a(T)\right], \tag{15.2}$$

Equation 15.2 has an intuitive meaning: the less number of times a suboptimal arm is pulled, the less the pseudo-regret will be. See Lemma 4.5 in Lattimore and Szepesvári (2020) for a formal proof of this fact. Henceforth, all of our intuitive discussion as well as formal proofs will use this definition of pseudo-regret.

15.5. Additional notes

- The explore-then-commit algorithm was first proposed by Robbins (1952), who called it the "win-stay, lose-shift" algorithm.

- The ϵ-greedy approach is suboptimal for bandits, but is still used extensively in reinforcement learning applications owing to its simplicity.

- One of the first bandit algorithms (called the *Gittins index policy*) was leveraged by Unilever for use in its clinical trials.

- Multi-armed bandits ideas are used extensively in A/B testing which has a wide variety of applications. One of the most interesting ones is by Barack Obama's campaign manager, Dan Siroker, who used bandit ideas to settle on the best homepage appearance for their fundraising website. Clearly, this is an online problem: all data being collected is high-stakes and influences the trajectory of the presidential campaign!

- For historical perspective on the origins of the multi-armed bandit problem, I enjoy reading Chapter 2 of the science journalist Brian Christian's book "Algorithms To Live By", titled "Explore/Exploit". (The book, more generally, is also a fantastic read!)

References

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms.* Cambridge University Press, 2020.

Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.