Lecture 2: Online Learning and Multi-armed Bandits

Akshay Krishnamurthy
akshay@cs.umass.edu

February 1, 2022

1 Introduction

Recap on generalization. In the last lecture, we studied concentration inequalities and saw a basic uniform convergence result. The main algorithmic principle here was *empirical risk minimization*. The result showed that given labeled data and a function class $\mathcal{F}$ we could competes with the best predictor in the class.

One salient feature of our result, which we did not discuss last week, is that there are essentially no restrictions on the complexity of the instance space $\mathcal{X}$, which could be images, text, DNA sequences, or anything else. The sample complexity, or convergence rate, has nothing to do with how complicated $\mathcal{X}$ is, and this means that we can still learn well even though we haven't seen the vast majority of the instances. So this demonstrates that in the supervised learning setting, we can actually generalize in the sense that we discussed in the introduction.

While the complexity of $\mathcal{X}$ is irrelevant, the complexity of the function class $\mathcal{F}$ appears prominently in the bound. This is more desirable since it allows us to encode our domain knowledge or capture inductive biases when designing the class $\mathcal{F}$. However, one thing to note it is that if you (a) you know nothing about your problem, and (b) you want to compete with the globally optimal function, then you might need to choose $\mathcal{F}$ to be all functions from $\mathcal{X} \to \mathcal{Y}$. Observe that in this case $\log|\mathcal{F}| \asymp |\mathcal{X}|$, so we will pay for the complexity of the instance space. So there is no free lunch, but the bound above gives us some more flexibility.

A new capability: exploration As we discussed, in addition to generalizing across inputs, RL agents also have to explore their environment, so that they can collect useful information to learn. Exploration, and reinforcement learning more broadly, is a natural sequential process, so we first need to understand how to make predictions in a sequential or online manner. Studying online learning is not strictly necessary for us, but it will showcase what are known as *version space* algorithms in a simple setting, and we will see these algorithms in future lectures. After online learning, we can turn to make sequential predictions while exploring the environment. Here we'll see two algorithmic techniques: *importance weighting* and *optimism*.

2 Online Learning

Online learning, which involves making predictions in a sequential manner, is a vibrant research area with many beautiful results. While the sequential nature is a natural connection to reinforcement learning, it is practically relevant in time-series prediction and elsewhere. As motivation, consider the following example:

Example 1. *Suppose we would like to make weather forecasts. Let us assume that we are only interested in predicting* $\{$*rain, no rain*$\}$ *on each day, so we are making binary decisions. Every night we make a prediction about the next day (perhaps based on whatever information we have available), and the next day we experience the weather so we can verify our prediction. Let's say we'd like to predict well over the course of the year.*

This task is fundamentally different from the supervised learning problem we saw in the last lecture. Indeed, here there is no training data at all. Instead, on every example (day), we must make a prediction without seeing the label, and only afterwards is the true label revealed to us, allowing us to update our model. So in some sense, every example is first a test example and then a training example. Another important difference is that modeling the days as iid would be quite unrealistic, since the weather changes dramatically with the seasons.

Formalism. Online learning provides a framework for designing algorithms suitable for such problems. We have all of the same ingredients as in offline supervised learning: there is an instance space $\mathcal{X}$, a label space $\mathcal{Y}$, a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ and a function class $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$. Here, we often referred to the function class as "experts." We play a T round game with an adversary as follows. At each round $t \in [T]$:

1. Adversary chooses (x_t, y_t) based on all information in the past. Adversary reveals x_t to the learner.
2. Learner makes a prediction $\hat{y}_t$.
3. Adversary reveals y_t and learner incurs loss $\ell(\hat{y}_t, y_t)$.

Naturally, we would like the learner to incur low loss, so we will try to bound the cumulative loss $\sum_t \ell(\hat{y}_t, y_t)$. In the most challenging setting, we do not place any restrictions on the adversary whatsoever. It can choose (x_t, y_t) however it likes based on all information in the past.

Mistake bound setting. In the simpler setting called the *mistake bound model*, we typically consider binary classification, meaning that $\mathcal{Y} = \{-1, +1\}$ and the loss function is $\ell(y, y') = \mathbf{1}\{y \neq y'\}$, and we will constrain the adversary to ensure that one of the experts $f^\star \in \mathcal{F}$ incurs 0 loss in total. Since we know that one of our experts in $\mathcal{F}$ incurs zero loss, it is quite natural to eliminate any expert that has made a mistake so far. Such algorithms are called *version space* algorithms. So this will be our "learning rule" and what remains to decide is how should we make predictions. For this, it turns out that the majority vote will be quite effective. This algorithm is the *Halving algorithm*. Start with $\mathcal{F}_0 = \mathcal{F}$ and for each t:

1. Receive x_t.
2. Predict $\hat{y}_t = \text{argmax}_{r \in \mathcal{Y}} |\{f \in \mathcal{F}_{t-1} : f(x_t) = r\}|$.
3. Receive label y_t and upate $\mathcal{F}_t = \{f \in \mathcal{F}_{t-1} : f(x_t) = y_t\}$.

Theorem 1. *In the mistake bound model, the halving algorithm incurs total loss $\sum_t \ell(\hat{y}_t, y_t) \leq \log_2(|\mathcal{F}|)$.*

Proof. The proof uses a "potential function" argument, where we track $|\mathcal{F}_t|$. By assumption, we have $|\mathcal{F}_t| \geq 1$ (since we will never eliminate $f^\star$), and $|\mathcal{F}_t| \leq |\mathcal{F}_1| = |\mathcal{F}|$. The main observation is that, since we predict by the majority vote, every time we make a mistake we must eliminate at least $1/2$ of the surviving functions. So we get

$$1 \leq |\mathcal{F}_T| \leq |\mathcal{F}_{T-1}| 2^{-\mathbf{1}\{\hat{y}_T \neq y_T\}} \leq |\mathcal{F}_0| 2^{-\sum_t \ell(\hat{y}_t, y_t)} = |\mathcal{F}| 2^{-\sum_t \ell(\hat{y}_t, y_t)}.$$

Taking logarithms of this equation proves the theorem. □

General setting. In the general setting, we should not assume that one expert is perfect, since this is quite a strong assumption that is unlikely to hold in practice. Instead, similar to what we saw in supervised learning, we want to compete with the best expert/predictor in our class. This is formalized by minimizing the notion of *regret*:

$$\text{Regret}_T := \sum_{t=1} \ell(\hat{y}_t, y_t) - \min_{f \in \mathcal{F}} \sum_{t=1}^{T} \ell(f(x_t), y_t)$$

This is quite similar to what we tried to control in the offline setting, except remember that now we are making no assumptions on the data sequence (x_t, y_t). So instead of competing with the best predictor on the distribution, we say that we are competing with the best predictor *in hindsight*. The goal is to obtain regret that scales sublinearly with T, this is referred to as *no-regret*. The ideas is that $o(T)$ regret demonstrates that, if the game is long enough, we can in fact compete with the best predictor in hindsight.

There are two key principles for regret minimization. The first is related to the version space approach we saw in the Halving algorithm: we want to keep track of which experts are doing well and which experts are doing poorly. However we want to do this in a softer manner, since an expert that starts out doing poorly can later turn out to be the best one. So we should not actually do strict eliminations, but rather something much softer.

The second observation is that, rather than taking a majority vote, we need to randomize our predictions. If we don't randomize, then the all powerful adversary knows what we are going to predict so it can force us to make

a mistake. On the other hand as long as our class contains the constant predictors $f_- : x \mapsto -1$ and $f_+ : x \mapsto +1$, one of these two will make at most $T/2$ mistakes so we will have $\Omega(T)$ regret.

Since we are going to randomize, we can reformulate the problem slightly, which will be useful for the bandit setting. We eliminate both the instance and label spaces altogether and we number the experts $1, \dots, N$. Instead, the adversary simply chooses a loss vector $\ell_t \in [0,1]^N$ and the learner chooses a distribution $p_t \in \Delta(N)$ over the experts. Then the learner incurs loss $\langle p_t, \ell_t \rangle$, while expert i incurs loss $\ell_t(i)$. Regret is measured as

$$\text{Regret}_T := \sum_{t=1}^{T} \langle p_t, \ell_t \rangle - \min_{i \in [N]} \sum_{t=1}^{T} \ell_t(i)$$

Exponential Weights. Let us instantiate a soft elimination procecdure with randomized predictions. This procedure has many names, including "Exponential Weights" and "Hedge." We start with weights $w_1 = (1, \dots, 1) \in \mathbb{R}^N$ and we have a learning rate η. Then for each t:

1. Set $p_t \propto w_t$ so $p_t(i) = w_t(i)/Z_t$ where $Z_t = \sum_j w_t(j)$. Predict with p_t.
2. Observe ℓ_t, incur loss $\langle p_t, \ell_t \rangle$. Update $w_{t+1}(i) \leftarrow w_t(i) \cdot \exp(-\eta \ell_t(i))$.

Theorem 2. *Assume that $\ell_t \in [0,1]^N$ for all t. Then for any $\eta \in (0, 1]$ we have*

$$\text{Regret}_T \le \frac{\eta}{2} \sum_{t=1}^{T} \langle p_t, \ell_t^2 \rangle + \frac{\log(N)}{\eta}. \tag{1}$$

With $\eta = \sqrt{2\log(N)/T}$, we obtain $\text{Regret}_T \le \sqrt{2T\log(N)}$.

Proof. As in the proof for the Halving algorithm, we use a potential function argument. Here we track $\log(Z_t)$ the log-sum of the weights of the experts. First observe that $\log(Z_1) = \log(N)$ and that for any expert $i \in [N]$ we have

$$\log(Z_{T+1}) = \log \sum_j \exp(-\eta \sum_t \ell_t(j)) \ge -\eta \sum_t \ell_t(i)$$

The key step is to relate the change in potential to the learner's loss

$$\begin{aligned} \log \frac{Z_{t+1}}{Z_t} &= \log \left(\sum_i \frac{w_t(i) \exp(-\eta \ell_t(i))}{Z_t} \right) = \log \left(\sum_i p_t(i) \exp(-\eta \ell_t(i)) \right) \\ &\le \log \left(1 - \eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \langle p_t, \ell_t^2 \rangle \right) \le -\eta \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \langle p_t, \ell_t^2 \rangle \end{aligned}$$

Here we used two analytical inequalities. The first is the upper bound $e^{-x} \le 1 - x + x^2/2$, which is obtain by Taylor's expansion since the third order term is negative. We apply this to $\exp(-\eta \ell_t(i))$. The second is that $\log(1+x) \le x$ for $x > -1$. This is where we obtain the restrictions on ℓ_t and η.

Now, we use a telescoping argument

$$-\eta \sum_t \ell_t(i) \le \log Z_{T+1} = \sum_{t=1}^{T} \log \frac{Z_{t+1}}{Z_t} + \log Z_1 \le -\eta \sum_t \langle p_t, \ell_t \rangle + \frac{\eta^2}{2} \sum_t \langle p_t, \ell_t^2 \rangle + \log(N)$$

Re-arranging this inequality and dividing by η proves the theorem. □

3 Adversarial Multi-armed bandits

Now that we have some basic understanding of how to make predictions in a sequential manner, let us connect this setting back to reinforcement learning and introduce the challenge of exploration. In RL, we typically call the experts "actions" and number them $\{1, \dots, A\}$. Then, the loss vector ℓ_t associates a loss to each action. The main difference is that our agent should choose a single action, execute it in the environment, and then receive a loss for *just that action*. So instead of observing the entire loss vector, we should only observe the loss for the action that we select. This protocol is formalized via the *multi-armed bandit* framework. We still play a T round game where in each round t:

1. Nature/Adversary chooses loss vector $\ell_t \in [0,1]^A$
2. Learner chooses action $a_t \in [A]$
3. Learner suffers loss $\ell_t(a_t)$ and only observes $\ell_t(a_t)$.

We still measure performance via the notion of regret. But now, the main challenge is that we do not see the losses for the actions that we don't play. So we must explore to learn about the performance of the other actions.

Exp3. The exponential weights algorithm can almost fit into the multi-armed bandit protocol by sampling $a_t \sim p_t$ on each round. In some sense, Exponential Weights already does some exploration, since it ensures $p_t(a) > 0$ for each a, t. This is a good sanity check since we won't entirely starve any action. However, we might play some actions with exponentially small probability, which could be bad.

To avoid this, we have to change the update rule. Note that we can't implement the original update rule anyway since we don't observe the losses for the actions we didn't play. And replacing them with, say zero, amounts to running Exponential Weights on a completely different loss sequence, so the regret bound doesn't make much sense.

We will ultimately replace the losses we didn't see with zero, but the key is to do this in an *unbiased* manner. For this, we introduce the concept of *importance weighting*. If at round t we play $a_t \sim p_t$ and observe $\ell_t(a_t)$, we construct the loss estimate

$$\tilde{\ell}_t(a) = \ell_t(a_t)\frac{\mathbf{1}\{a_t = a\}}{p_t(a)}$$

Intuitively, if we choose some action with low probability, but it incurs large loss, we inflate this dramatically to account for all the times we didn't play this action. Importance weighting satisfies a few critical properties:

Lemma 3. *The importance weighted estimates satisfy:*

$$\mathbb{E}_{a_t \sim p_t}[\tilde{\ell}_t(a)] = \ell_t(a), \quad \mathbb{E}_{a_t \sim p_t}[\tilde{\ell}_t^2(a)] = \ell_t(a)^2/p_t(a), \quad \mathbb{E}_{a \sim p_t}[1/p_t(a)] = A$$

Proof. The calculations are fairly straightforward:

$$\mathbb{E}_{a_t \sim p_t}[\tilde{\ell}_t(a)] = \sum_{a_t} p_t(a_t)\ell_t(a_t)\frac{\mathbf{1}\{a_t = a\}}{p_t(a)} = \ell_t(a)$$

$$\mathbb{E}_{a_t \sim p_t}[\tilde{\ell}_t^2(a)] = \sum_{a_t} p_t(a_t)\ell_t^2(a_t)\frac{\mathbf{1}\{a_t = a\}}{p_t^2(a)} = \ell_t(a)^2/p_t(a)$$

$$\mathbb{E}_{a \sim p_t}[1/p_t(a)] = \sum_a p_t(a)/p_t(a) = A$$

□

The Exp3 algorithm simply runs Exponential Weights using importance weighted loss estimates. By the above lemma, we immediately obtain the following theorem.

Theorem 4. *If $\ell_t \in [0,1]^A$ and $\eta = \sqrt{\frac{\log(A)}{AT}}$, the Exp3 algorithm has an expected regret bound of*

$$\mathbb{E}[\text{Regret}_T] \le \sqrt{2TA\log(A)}$$

Proof. If we think of the adversary as deploying the losses $\tilde{\ell}_t$ then the Exponential weights regret bound is

$$\sum_t \left\langle p_t, \tilde{\ell}_t \right\rangle - \sum_t \tilde{\ell}_t(a^\star) \le \frac{\eta}{2}\sum_t \left\langle p_t, \tilde{\ell}_t^2 \right\rangle + \frac{\log(A)}{\eta}$$

Here $a^\star \in \text{argmin}_a \sum_t \ell_t(a)$ is the best action on the original sequence. (This requires one small observation in the Exponential Weights proof, which is that even though $\tilde{\ell}_t$ may not be bounded, we do have $\left\langle p_t, \tilde{\ell}_t \right\rangle$ bounded, so we can still use the inequality $\log(1+x) \le x$.)

Taking expectation (over the random actions $a_{1:T}$) and using our bound from the previous lemma, we have

$$\mathbb{E}\left[\sum_t \ell_t(a_t) - \ell_t(a^\star)\right] \le \frac{\log(A)}{\eta} + \frac{\eta}{2}\mathbb{E}\left[\sum_t \sum_a p_t(a)\frac{\ell_t(a)^2}{p_t(a)}\right] \le \frac{\log(A)}{\eta} + \frac{\eta}{2}\cdot AT$$

With our choice of η we obtain the result. □