Georgia Tech S'22 CS 6550/8803: Advanced Algorithms & Uncertainty

Lecture 10: Online Learning for Full and Partial Feedback

Lecturer: *Sahil Singla* Last updated: February 5, 2022

Today we will study one of the major success stories of Online Computation in Learning Theory. Let's start by a motivating example.

1 Motivating example: Prediction from Expert Advice

Now we briefly illustrate the general idea in a simple and concrete setting. This is known as the *Prediction from Expert Advice* problem.

Imagine the process of picking good times to invest in a stock. For simplicity, assume that there is a single stock of interest, and its daily price movement is modeled as a sequence of binary events: up/down. (Below, this will be generalized to allow non-binary events.) Each morning we try to predict whether the price will go up or down that day; if our prediction happens to be wrong we lose a dollar that day, and if it's correct, we lose nothing.

The stock movements can be *arbitrary* and even *adversarial*[1]. To balance out this pessimistic assumption, we assume that while making our predictions, we are allowed to watch the predictions of n "experts". These experts could be arbitrarily correlated, and they may or may not know what they are talking about.

What's the best guarantee you might hope for in this setting? You could, for instance, ask for a prediction that is correct at every single round. But OK, this is ridiculous to ask for. Maybe all the experts are wrong, or maybe they all know nothing. The next strongest guarantee you might hope for is a prediction that is correct at every round that at least one expert is correct. In other words, you want to be as good as the best expert every single round. But some quick thought shows that this is again a ridiculous guarantee to hope for: maybe the experts all know nothing and just guess randomly. Then surely one of them will happen to be correct every round, but you can't possibly expect to know the right answer with no information.

Instead, what you can ask for is that *on average* over T rounds, your average performance is at least as good as the best expert's *average performance* over T rounds. That is, there may be numerous rounds where some expert is correct and you are wrong. But if there's a single expert who's correct 70% of the time, then you are also correct (almost) 70% of the time, but the rounds in which you're correct could be the same or different.

So the algorithm's goal is to limit its cumulative losses (i.e., bad predictions) to roughly the same as the *best* of these experts. This difference between the algorithm's total loss and the best expert is known as *regret*. At first sight getting a small regret seems an impossible

[1]Note that finance experts have studied stock movements for over a century and fitted all kinds of stochastic models to them. But we are doing computer science here, and we will see that this adversarial view will help us apply the same idea to a variety of other settings.

goal, since it is not known until the end of the sequence who the best expert was, whereas the algorithm is required to make predictions all along.

For example, the first algorithm one thinks of is to compute each day's up/down prediction by going with the majority opinion among the experts that day. But this algorithm doesn't work because a majority of experts may be consistently wrong on every single day, while some single expert in this crowd happens to be right every time. In fact, we leave as an exercise to show that every deterministic algorithm incurs $\Omega(T)$ regret for some input sequence. However, randomization will come to our rescue in the next section.

2 The Hedge Algorithm for Full-Feedback

The *hedge algorithm* corrects the trivial algorithm by maintaining a *weighting* of the experts. Initially all have equal weight. As time goes on, some experts are seen as making more wrong predictions than others, and the algorithm decreases their weight proportionately. The algorithm's prediction for each day is computed by by choosing a random expert proportional to its weight for that day.

Formally, let $\mathbf{m}^{(t)} = ({m_1}^{(t)}, {m_2}^{(t)}, \ldots, {m_n}^{(t)}) \in [0,1]^n$ denote the costs of n actions. Starting with ${w_i}^{(1)} := 1$, in round t the algorithm takes exactly one of the n actions with probability proportional to its weight. The algorithm then receives the feedback $\mathbf{m}^{(t)}$ and updates its weights using the exponential function given in (1).

Hedge algorithm

Initialization: Fix an $\eta \leq \frac{1}{2}$. For each decision i, associate the weight ${w_i}^{(t)} := 1$.
For $t = 1, 2, \ldots, T$:

1. Choose decision i with probability proportional to its weight ${w_i}^{(t)}$. I.e., use the distribution over decisions $\mathbf{p}^{(t)} = \{{w_1}^{(t)}/\Phi^{(t)}, \ldots, {w_n}^{(t)}/\Phi^{(t)}\}$ where $\Phi^{(t)} = \sum_i {w_i}^{(t)}$.
2. Observe the costs of the decisions $\mathbf{m}^{(t)}$.
3. Penalize the costly decisions by updating their weights as follows: for every decision i, set
$$ {w_i}^{(t+1)} = {w_i}^{(t)} \cdot \exp(-\eta {m_i}^{(t)}) \qquad \text{(update rule).} \tag{1}$$

Figure 1: The Hedge algorithm.

Theorem 1. *The expected regret of Hedge is bounded by* $\sqrt{2T \ln n}$.

The following lemma will immediately prove the theorem.

Lemma 2. *For any cost vectors* $\mathbf{m}^{(t)} \in \mathbb{R}^n_{\geq 0}$*, the Hedge algorithm satisfies*

$$\sum_t \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} - \sum_t {m_i}^{(t)} \leq \frac{\eta}{2} \sum_t (\mathbf{m}^{(t)})^2 \cdot \mathbf{p}^{(t)} + \frac{\ln n}{\eta},$$

where $(\mathbf{m}^{(t)})^2$ *denotes an* n*-dimensional vector with* i*-th entry* $({m_i}^{(t)})^2$*.*

Remark: The proof of Lemma 2 only requires non-negative costs, and not the fact that the costs are bounded by 1. This additional power will be crucial in the next section on Bandit feedback.

Proof of Theorem 1. Since ${m_i}^{(t)} \leq 1$ and $\mathbf{1} \cdot \mathbf{p} \leq 1$, we get from Lemma 2 that

$$\sum_t \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} - \sum_t {m_i}^{(t)} \leq \frac{\eta}{2} T + \frac{\ln n}{\eta}.$$

Setting $\eta = \sqrt{\frac{2 \ln n}{T}}$ completes the proof of Theorem 1. $\square$

Next, we prove the main lemma.

Proof of Lemma 2. Consider the total weight function $W^{(t)} = \sum_i {w_i}^{(t)}$ which starts with $W^{(1)} = n$. The proof will proceed by obtaining a lower and an upper bound on $W^{(T)}$.

For the lower bound, we know $W^{(T+1)} \geq {w_i}^{(T)}$. A simple induction shows that ${w_i}^{(t)} = \exp(-\eta \sum_{s<t} {m_i}^{(s)})$, so we get

$$\exp\Big(-\eta \sum_t {m_i}^{(t)} \Big) \leq W^{(T+1)}. \tag{2}$$

The upper bound on $W^{(T+1)}$ will require the following two inequalities, which we leave as an exercise:

$$\begin{aligned} \exp(-x) &\leq 1 - x + x^2/2 \qquad \forall x \geq 0 \quad \text{and} \\ \exp(x) &\geq 1 + x \qquad \forall x. \end{aligned}$$

Let us now write $W^{(t+1)}$ in terms of $W^{(t)}$. Recall that ${p_i}^{(t)} = \frac{{w_i}^{(t)}}{W^{(t)}}$, so

$$\begin{aligned} W^{(t+1)} &= \sum_i {w_i}^{(t)} \exp(-\eta {m_i}^{(t)}) \\ &\leq \sum_i {w_i}^{(t)} \cdot \Big(1 - \eta {m_i}^{(t)} + \frac{\eta^2}{2} ({m_i}^{(t)})^2 \Big) \\ &= W^{(t)} - \eta W^{(t)} \sum_i {p_i}^{(t)} {m_i}^{(t)} + \frac{\eta^2}{2} W^{(t)} \sum_i {p_i}^{(t)} ({m_i}^{(t)})^2 \\ &\leq W^{(t)} \cdot \exp\Big(-\eta \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} + \frac{\eta^2}{2} (\mathbf{m}^{(t)})^2 \cdot \mathbf{p}^{(t)} \Big). \end{aligned}$$

Applying this inequality recursively, we get

$$W^{(T+1)} \leq W^{(1)} \cdot \exp\Big(-\eta \sum_t \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} + \frac{\eta^2}{2} \sum_t (\mathbf{m}^{(t)})^2 \cdot \mathbf{p}^{(t)} \Big).$$

Using $W^{(1)} = n$ and combining this with the lower bound in (2) gives

$$\exp(-\eta \sum_t {m_i}^{(t)}) \leq n \cdot \exp\Big(-\eta \sum_t \mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} + \frac{\eta^2}{2} \sum_t (\mathbf{m}^{(t)})^2 \cdot \mathbf{p}^{(t)} \Big).$$

Taking logs finishes the proof of the lemma. □

There are several known extensions of the setup in Theorem 1. For example, in the next lecture we will consider a setting with an infinite number of actions but with more structured (convex) cost functions. See the survey [1] for more applications.

3 The Exp3 Algorithm for Partial/Bandit Feedback

Till now we have worked with in the full-feedback model whereby in round t the algorithm receives the full cost vector $\mathbf{m}^{(t)}$ as its feedback. For several applications, however, it's not possible to receive the entire cost vector, but only ${m_i}^{(t)}$ for the action i that is played in round t. This model is known as the Bandit feedback model. The name comes from slot machines in a casino where the gambler has to decide which of the n slot machines to play in round t. Here the gambler only receives feedback for the played machine and no feedback from the other machines.

Exp3 algorithm

Initialization: Fix an $\eta \leq \frac{1}{2}$. For each decision i, associate the weight ${w_i}^{(t)} := 1$.
For $t = 1, 2, \ldots, T$:

1. Choose decision i with probability proportional to its weight ${w_i}^{(t)}$. I.e., use the distribution over decisions $\mathbf{p}^{(t)} = \{{w_1}^{(t)}/\Phi^{(t)}, \ldots, {w_n}^{(t)}/\Phi^{(t)}\}$ where $\Phi^{(t)} = \sum_i {w_i}^{(t)}$.
2. Observe the cost ${m_i}^{(t)}$ of decision i. Define a surrogate cost vector of the decisions $\tilde{\mathbf{m}}^{(t)}$ where ${\tilde{m}_i}^{(t)} = \frac{{m_i}^{(t)}}{{p_i}^{(t)}}$ and 0 otherwise. I.e., only the i-th entry of $\tilde{\mathbf{m}}^{(t)}$ is non-zero.
3. Penalize the costly decisions by updating their weights as follows: for every decision i, set

$$ {w_i}^{(t+1)} = {w_i}^{(t)} \cdot \exp(-\eta {\tilde{m}_i}^{(t)}) \qquad \text{(update rule)}. \tag{3}$$

Figure 2: The Exp3 algorithm.

Theorem 3. *The expected regret of Exp3 is bounded by* $\sqrt{2Tn \ln n}$.

This result may seem surprising that even with such limited feedback an algorithm exists with $o(T)$ regret (for small n). The idea of the algorithm is to still play the Hedge algorithm but on surrogate $\tilde{\mathbf{m}}^{(t)}$ costs (since $\mathbf{m}^{(t)}$ is unknown). The crucial property of $\tilde{\mathbf{m}}^{(t)}$ is that in expectation it equals $\mathbf{m}^{(t)}$.

Remark: The algorithm in Figure 2 is a simpler variant of the original Exp3 algorithm in [2] due to Stoltz [3].

Proof. The proof of this theorem relies on the analysis of Hedge in Section 2. In particular, since $\tilde{m}_i{}^{(t)} \geq 0$ the analysis of Hedge in Lemma 2 (we don't need $\tilde{m}_i{}^{(t)} \leq 1$ as remarked there) can be repeated to give

$$\mathbb{E}\Big[\sum_t \tilde{\mathbf{m}}^{(t)} \cdot \mathbf{p}^{(t)} - \sum_t \tilde{m}_i{}^{(t)}\Big] \leq \mathbb{E}\Big[\frac{\eta}{2}\sum_t (\tilde{\mathbf{m}}^{(t)})^2 \cdot \mathbf{p}^{(t)}\Big] + \frac{\ln n}{\eta}. \tag{4}$$

The LHS of (4) can be simplified to equal the expected regret of the algorithm, i.e.,

$$\sum_t \mathbb{E}_{\leq t-1}\Big[\mathbb{E}_t\big[\tilde{\mathbf{m}}^{(t)} \cdot \mathbf{p}^{(t)}\big] - \sum_t \mathbb{E}_t\big[\tilde{m}_i{}^{(t)}\big]\Big] = \sum_t \mathbb{E}_{\leq t-1}\Big[\mathbf{m}^{(t)} \cdot \mathbf{p}^{(t)} - \sum_t m_i{}^{(t)}\Big].$$

This is because $\mathbb{E}_t[\tilde{\mathbf{m}}^{(t)} \cdot \mathbf{p}^{(t)}] = \mathbb{E}_t[\tilde{\mathbf{m}}^{(t)}] \cdot \mathbf{p}^{(t)} = \mathbf{m}^{(t)}$ as after fixing the algorithm's distribution $\mathbf{p}^{(t)}$ for round t, we draw $\tilde{\mathbf{m}}^{(t)}$ independently. (In the literature, such a property is often written as $\tilde{\mathbf{m}}^{(t)}$ gives an *unbiased estimator* of $\mathbf{m}^{(t)}$.)

Now we simplify the RHS of (4). Observe that

$$\mathbb{E}[\tilde{m}_i{}^{(t)})^2] = p_i{}^{(t)}\Big(\frac{m_i{}^{(t)}}{p_i{}^{(t)}}\Big)^2 = \frac{(m_i{}^{(t)})^2}{p_i{}^{(t)}},$$

which gives

$$\mathbb{E}[(\tilde{\mathbf{m}}^{(t)})^2 \cdot \mathbf{p}^{(t)}] = \sum_i (m_i{}^{(t)})^2 \leq n.$$

Substituting this in (4), the total expected regret of the algorithm can be bounded as

$$\mathbb{E}[\sum_t \tilde{\mathbf{m}}^{(t)} \cdot \mathbf{p}^{(t)}] - \sum_t m_i{}^{(t)} \leq \frac{\eta}{2}\sum_t n + \frac{\ln n}{\eta}.$$

Choosing $\eta = \sqrt{\frac{2\ln n}{T}}$ completes the proof of the theorem. □

The dependence of Theorem 3 on n is polynomial, unlike the Hedge algorithm for full-feedback where the dependence was logarithmic. This makes a big difference in how broadly this result can be applied. Indeed, it's still an active area of research (see the book [4]) to understand when can we obtain $o(T)$ regret even with an exponentially large n (we will see one such example of Bandit Convex Optimization in a later lecture).

Notes

The lecture is partly based on COS 521 notes from Princeton.

References

[1] S. Arora, E. Hazan, S. Kale. *The multiplicative weights update method: A meta algorithm and its applications.* Theory of Computing, Volume 8 (2012), pp. 121–164.

[2] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002). The nonstochastic multiarmed bandit problem. SIAM journal on computing, 32(1), 48-77.

[3] Stoltz, Gilles. Incomplete information and internal regret in prediction of individual sequences. PhD diss., Université Paris Sud-Paris XI, 2005.

[4] Lattimore, Tor, and Csaba Szepesvári. Bandit algorithms. Cambridge University Press, 2020.