17-740 Spring 2025

Algorithmic Foundations of Interactive Learning

Lecture 2: Intro to Online Learning / Weighted Majority

Lecturer : Drew Bagnell Scribe: Gokul Swamy

2.1 The Problem of Induction

Machine learning (and perhaps all of science) can be viewed as attempting to solve the problem of *induction*: making predictions about the future conditioned on the past.

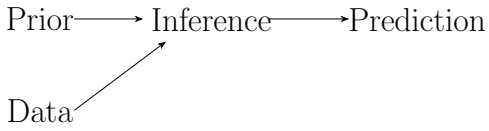

Figure 2.1: We can view induction as the problem of turning what we know into a prediction about something we don't know. What assumptions are required for this to be possible?

Throughout the centuries, there have been a variety of views of the problem of induction:

- **Laplace:** If the sun rises for 1000 straight days, we should conclude that it has a high chance of rising on the 1001st day.
- **Hume (responding to the above):** If a chicken has been fed for 1000 days, should it conclude it won't become dinner tomorrow?

In this course, we will take the *no-regret view* (which grew out of the late 20th century): we will make no assumptions about how well the past predicts the future and attempt to do as well anyone could hope to. Note the contrast between this and the *statistical* view in standard machine learning, where we assume we will be tested on the same distribution our training data was drawn from. As we will explore later in the course, one of the key challenges of interactive learning is being able to predict accurately even under distribution shift, and the no-regret view will give us a rigorous theoretical foundation for doing so.

2.2 Prediction with Expert Advice

For this lecture, we will focus primarily on the setting of *prediction with expert advice.* Specifically, consider predicting a sequence of binary labels for T rounds, i.e. $p_t \in \{0, 1\}$.

We will assume access to a set of experts $\mathcal{E}$ with $|\mathcal{E}| = N$ to help us with this task, e.g.:

- e_0: always predicts 0
- e_1: always predicts 1
- e_2: predict what happened at the last round
- e_3 predicts $0, 1, 0, 1, \ldots$

We will assume there is some ground truth sequence of labels $y = (y_1, \ldots, y_T)$ that is not necessarily chosen in advance (i.e. it can be chosen adversarially in response to what our algorithm predicts). We will judge algorithms based on the sum of per-round losses, where

$$\ell_t(p_t, y_t) = \mathbb{1}[p_t \neq y_t]. \tag{2.1}$$

2.2.1 Regret

Without any assumptions, this problem is quite hard. We will instead attempt to compete with the best *fixed* expert in hindsight. Specifically, we define regret as follows:

Definition 1 ((Static) Regret)

$$\textit{Regret}(T) = \sum_t^T \ell_t(p_t) - \ell_t(e_t^\star) = \min_{e \in \mathcal{E}} \sum_t^T \ell_t(p_t) - \ell_t(e_t). \tag{2.2}$$

We say an algorithm achieves no-regret if it drives the time-averaged regret to zero, i.e. if

$$\lim_{T \to \infty} \frac{\textit{Regret}(T)}{T} \to 0. \tag{2.3}$$

This may still seem like a lot to ask for (we haven't made any assumptions on how the y_t's are picked!). However, as we will spend the next few lectures exploring, there are a plethora of algorithms that satisfy the no-regret property, some of which you are probably already familiar with (e.g. gradient descent).

2.2.2 Follow-The-Leader (FTL)

We will first present what might be the most approach to online learning and then argue why it doesn't quite achieve no-regret. Let us define follow-the leader as the algorithm that picks the best expert in hindsight, i.e.

Definition 2 (Follow-The-Leader (FTL)) *Follow-The-Leader selects*

$$p_t = \arg\min_{e \in \mathcal{E}} \sum_{\tau}^{t-1} \ell_t(e). \tag{2.4}$$

For a moment, let us set $\mathcal{E} = \{e_0, e_1\}$ and tie-break in favor of e_0. Let us run this algorithm against an unforgiving world:

e_0 Correct	e_1 Correct	p_t	y_t
0	0	0	1
0	1	1	0
1	1	0	1

Figure 2.2: If an adversary were to choose y_t in a way to cause us maximal pain, they could cause us to suffer linear regret (i.e. be wrong at every timestep even though no individual expert is).

As we will discuss in future lectures, the core issue with FTL is that it is unstable – it switches its predictions too easily, which makes it possible for an adversary to take advantage of it.

2.2.3 Halving Algorithm

For simplicity, we're now going to assume there is a perfect expert, i.e. $\exists e^\star \in \mathcal{E}$ s.t. $\sum_t^T \ell_t(e_t^\star) = 0$. Let's now try a smarter approach:

Definition 3 (Halving Algorithm) *The Halving Algorithm maintains a set of experts $\mathcal{E}_t$ at each round. $\mathcal{E}_0 = \mathcal{E}$. If an expert ever makes a mistake, it is removed from the set. At each round, p_t is set to the majority vote across $\mathcal{E}_t$.*

Observe that the total number of mistakes (i.e. total regret) the above algorithm can suffer is $\log_2(N)$. This is because if we ever make a mistake, we eliminate half the experts in the last round's $\mathcal{E}_t$ and we can only do this process $\log_2(N)$ times. This is sometimes referred to as a *mistake bound*. Observe that this strategy is effectively weighting an expert by zero if they ever make a mistake.

2.2.4 Deterministic Weighted Majority

If we want to handle the case where no expert may be perfect, we need to adopt a less aggressive weighting scheme:

Definition 4 (Deterministic Weighted Majority (DWM)) *DWM maintains a weight w_i for each $e \in \mathcal{E}$. It predicts*

$$p_t = \mathbb{1}\left[\sum_{i:e_i(t)=1} w_i^t \geq \sum_{i:e_i(t)=0} w_i^t\right] \tag{2.5}$$

(i.e. takes a weighted majority vote). Whenever an expert is wrong, it sets $w_i^{t+1} = 0.5 w_i^t$.

We remark that this recovers the halving algorithm if we change the 0.5 to 0 and a standard majority vote if we instead change to 1.

We now prove a regret bound for this algorithm:

Lemma 1 (DWM Regret Bound) *Let m denote the number of mistakes DWM makes and $m^\star$ denote the number of mistakes the best expert makes. Then, we have*

$$m \leq \log_2\left(\frac{4}{3}\right)(m^\star + \log_2(N)). \tag{2.6}$$

Proof: Observe that for DWM to make a mistake, half of the experts (in terms of weight) need to be wrong and after such a mistake, this half looses half of their weight. This means that every time the algorithm makes a mistake, the full set of experts loose at least $\frac{1}{4}$ of their total weight W. This directly implies that

$$N\left(\frac{3}{4}\right)^m \geq W \geq \left(\frac{1}{2}\right)^{m^\star}. \tag{2.7}$$

Re-arranging terms gives the regret bound. ■

We remark that the above regret bound is tight via a variant of the switching construction from 2.2.2.

Unfortunately, $\log_2\left(\frac{4}{3}\right) \approx 2.41 > 1$, so the above algorithm doesn't quite achieve no-regret.

2.2.5 Randomized Weighted Majority

To actually achieve no-regret, we will now switch to predicting a *distribution* over experts, and outputting what an expert sampled from that distribution said. While it might not be immediately obvious why this is a good idea, in later lectures where we discuss game solving, the reasoning behind this shift will become more transparent.

Definition 5 (Randomized Weighted Majority (RWM)) *RWM initializes all weights to 1, i.e.* $w_i^0 = 1$. *Define distribution* $p_t(i) = w_t(i)/\sum_j^N w_t(j)$. *RWM samples from this distribution and predicts what the sampled expert predicts. If an expert is wrong, it sets* $w_i^{t+1} \leftarrow \beta \cdot w_i^t$.

We won't prove this in this lecture, but we can bound the *expected* regret as

$$\mathbb{E}[m] \leq \frac{m^\star \ln(1/\beta) + \ln N}{\beta}. \tag{2.8}$$

For an appropriate choice of β, this algorithm will achieve no-regret. Observe that because we're now shelling out to an expert, we no longer need to restrict ourselves to the binary prediction setting.

2.2.6 Generalized Weighted Majority

For our final algorithm of this series, we will study Generalized Weighted Majority, for which we will assume $\ell_t \in [0, 1]$:

Definition 6 (Generalized Weighted Majority (GWM)) *GWM is identical to RWM but it sets*

$$w_i^{t+1} \leftarrow w_i^y \cdot \exp(-\varepsilon \ell_t(e_i)), \tag{2.9}$$

where ϵ *can be thought of as a kind of learning rate.*

Again, without proof, we will state that

$$\mathbb{E}[Regret(T)] \leq \varepsilon \sum_t^T \ell_t(e_t^\star) + \frac{\ln(N)}{\varepsilon} \leq T\varepsilon + \frac{\ln(N)}{\varepsilon}, \tag{2.10}$$

where the second inequality comes from the scale of the loss. If we take the gradient of the above w.r.t. ε and set it equal to zero, we can find that the optimal learning rate is $O\left(\frac{1}{\sqrt{T}}\right)$. Plugging in this $\varepsilon^\star$ gives us a final regret bound of

$$\mathbb{E}[Regret(T)] \leq O(\sqrt{T}) + O(\sqrt{T}) \ln(N). \tag{2.11}$$

Because all terms are sub-linear in T, we have proved the above algorithm achieves no-regret. Huzzah! We remark that without further assumptions, the $\sqrt{T}$ and $\ln N$ scaling are unavoidable up to constants.

2.2.7 Cover's Universal Portfolio

We now present an example to perhaps temper the excitement the above example may have evoked in the mind of an avaricious reader. What if, for a moment, we consider each expert a stock in the stock market, and our problem to invest optimally (i.e. compete with the best stock in hindsight). Why does the no-regret property not imply one of the preceding algorithms could do this trivially? Observe that all the no-regret property says is that

$$\frac{1}{T}\sum_{t}^{T} \ell_t(p_t) - \ell_t(e^\star) \to 0, \tag{2.12}$$

i.e. for the losses ℓ_t we *actually observed*, the regret goes to zero. It does not imply that for any counter-factual set of losses we could have observed we would be able to drive regret to zero. The reason this causes issues in stock trading is that if one puts enough money into a particular stock, they change the value of that stock (i.e. the sequence of p_t we choose influences the set of ℓ_t we observe). Thus, while the no-regret property implies we do well *conditioned on our choices*, it does not guarantee we do as well as any investor could have done in the market (as they may induce a different set of ℓ_t with their choices of p_t).

2.3 Convexity

We now define two concepts we will use repeatedly throughout the course.

Definition 7 (Convex Set) *A set $\mathcal{X}$ is convex if, $\forall a, b \in \mathcal{X}$ and $\forall \alpha \in [0,1]$,*

$$\alpha a + (1 - \alpha b) \in \mathcal{X}. \tag{2.13}$$

In words, this definition is saying that there is a straight line to every other point in the set that is contained in the set.

Definition 8 (Convex Function) *A function f is convex if, $\forall x, y \in dom(f)$ and $\forall \alpha \in [0,1]$,*

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha) f(y). \tag{2.14}$$

Equivalently, a function is convex if its epigraph (the set of points above the function) is a convex set.