CS272 - Theoretical Foundations of Learning, Decisions, and Games

Lecture 7: Online Learning III

September 18, 2025

Lecturer: Nika Haghtalab *Readings: UML Chapter 21*
Scribe: Ryan Campbell

1 Recap

In the last lecture we attempted to get MistakeBound $\leq$ OPT $+ (*)$, where $(*)$ is small some quantity of the order $\approx \log|\mathcal{C}|$. We introduced an algorithm that almost met this requirement, but with a constant multiplicative factor: the Weighted Majority Algorithm with MistakeBound $\leq 2.4(\text{OPT} + \log|\mathcal{C}|)$.

We then introduced the Randomized Weighted Majority algorithm (also known as the Multiplicative Weights Update, MWU), that added randomization to the choice of the expert. While doing so, we also started to work with general-purpose cost functions that go beyond 0-1 losses. We ended the last lecture with the proof of the following lemma on the performance of RWM.

Theorem 1.1. *For cost functions $c^t : [n] \to [0,1]$, the RWM(ε) algorithm gets*

$$\underbrace{\mathbb{E}\left[\sum_{t=1}^{T} c^t(i^t)\right]}_{cost(ALG)} \leq \frac{1}{1-\varepsilon}\underbrace{\mathbb{E}\left[\min_{i^*\in[n]}\sum_{t=1}^{T} c^t(i^*)\right]}_{OPT} + \frac{1}{\varepsilon}\ln(n)$$

In this lecture, we first return to our goal of obtaining bounds on $\text{cost}(ALG) - \text{OPT}$ and then discuss the setting where the number of experts is infinitely large.

2 Regret and No-Regret Algorithms

We define Regret to be the gap between $\text{cost}(ALG)$ and OPT.

Definition 2.1. The *Regret* of an algorithm $\mathcal{A}$ on a sequence of costs $c^1, c^2, \ldots, c^T$ is

$$\text{Regret}\left(\mathcal{A}, \{c\}_{t=1}^{T}\right) = \sum_{t=1}^{T} c^t(i^t) - \min_{i^*\in[n]}\sum_{t=1}^{T} c^t(i^*)$$

where i^t are the choices of $\mathcal{A}$. Note that oftentimes we are concerned with expected regret.

$$\mathbb{E}\left[\text{Regret}\left(\mathcal{A}, \{c\}_{t=1}^{T}\right)\right] = \mathbb{E}\left[\sum_{t=1}^{T} c^t(i^t) - \min_{i^*\in[n]}\sum_{t=1}^{T} c^t(i^*)\right]$$

We often refer to the worst-case regret of algorithm $\mathcal{A}$ on any sequence of costs by just $\text{Regret}(\mathcal{A})$.

Corollary 2.2. *The expected regret of* $\text{MWU}(\varepsilon)$ *for* $\varepsilon^* = \min\left\{\frac{1}{2}, \sqrt{\frac{\ln(n)}{2\cdot}}\right\}$*, is*

$$2\sqrt{2 \cdot \mathit{OPT} \cdot \ln(n)} \leq \mathcal{O}\left(\sqrt{T \cdot \ln(n)}\right).$$

Proof. Note that for $\varepsilon < \frac{1}{2}$, we have that $\frac{1}{1-\varepsilon} < 1 + 2\varepsilon$. Then using Theorem 1.1, we have that

$$\mathbb{E}\left[\text{Regret}\left(MWU\right)\right] \leq 2\varepsilon\text{OPT} + \frac{1}{\varepsilon}\ln(n)$$

When OPT is known, we can choose ε^* according to the following

$$\varepsilon^* = \min\left\{\frac{1}{2}, \sqrt{\frac{\ln(n)}{2 \cdot \text{OPT}}}\right\}$$

to get the desired bound. Note that, $\text{OPT} \leq T$ which gives the final inequality in the stated results. □

Remark 2.3. When OPT is not known, one can use $\varepsilon^* = \min\left\{\frac{1}{2}, \sqrt{\frac{\ln(n)}{2 \cdot T}}\right\}$ to get the same asymptotic regret bound. Sometime T might not even be known. In such cases, a "guess-and-double" trick can be used: First guess a small horizon T_i, run the algorithm with for this number of steps with the associated step size. If the actual horizon exceed T_i then double the estimate $T_{i+1} = 2T_i$ and continue with the associated ϵ. Continue until the guessed horizon exceeds the actual number of time steps played. Note that the regret of this trick is

$$\sum_{i=0}^{\log(T)} O\left(\sqrt{2^i \ln(n)}\right) \in O\sqrt{T\ln(n)},$$

since for any k

$$\sum_{i=0}^{k} (\sqrt{2})^{\,i} = \frac{(\sqrt{2})^{\,k+1} - 1}{\sqrt{2} - 1} = \Theta\Big((\sqrt{2})^{\,k}\Big) = \Theta\Big(\sqrt{2^{\,k}}\Big).$$

There are schedules for changing ϵ that solely depend on ϵ_t that lead to smaller constants hidden in $O(\cdot)$ notation. We won't discuss these and from here onward we assume that we always know T in advance (since the guess-and-double trick can be used to overcome lack of knowledge otherwise).

Definition 2.4. A *No-Regret* algorithm is an algorithm where the (expected) regret on any sequence of costs is $\leq o(T)$. Equivalently, an algorithm $\mathcal{A}$ is no-regret is

$$\lim_{T\to\infty} \frac{\text{Regret}(\mathcal{A})}{T} = 0$$

3 No-regret Algorithms with Infinite Hypothesis Classes

A natural question to ask is what happens if $|\mathcal{C}|$ is unbounded? In this case, without further assumptions, nothing can be done to learn with no regret. To begin answering this question, we consider what we have shown so far in Table 1.

Table 1: Summary of Online vs. Offline

	Sample Complexity	**error / Regret**	
		finite $\mathcal{C}$	**infinite $\mathcal{C}$**
Offline	$m \in \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(d + \ln\frac{1}{\delta}\right)\right)$	$\varepsilon \leq \mathcal{O}\left(\sqrt{m \ln|\mathcal{C}|}\right)$	$\varepsilon \leq \mathcal{O}\left(\sqrt{m\,\mathrm{VCD}(\mathcal{C})}\right)$
Online		$\mathcal{O}\left(\sqrt{T \ln|\mathcal{C}|}\right)$	?

We wish to fill in the "?" part of the table, but it is not so clear whether or not VCD is the correct notion for Online learning with infinite $\mathcal{C}$.

This question cannot be separated from the question of what algorithms are no-regret! While, we have seen that RWM is no-regret for finite hypothesis classes, its dependence on $\ln(|\mathcal{C}|)$ is problematic. On the other hand, ERM is also not a good algorithm even for finite $\mathcal{C}$, as shown below:

Theorem 3.1. *For any deterministic Algorithm, there is an adversarial sequence where*

$$\begin{aligned} \textit{Regret} &\geq \Omega(T) \\ &\geq \left(1 - \frac{1}{n}\right) T \end{aligned}$$

Proof. Any deterministic algorithm chooses i^t as a function of the history so far. For example, in ERM,

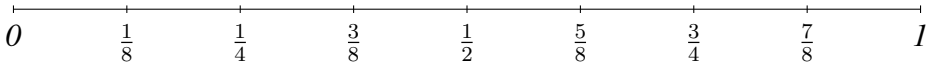
$$i^t \leftarrow \operatorname*{argmin}_{i \in [n]} \sum_{\tau=1}^{t-1} c^\tau(i).$$

Because of the determinism, i^t can always be known to an adversary. Thus, the adversary can choose $c^t(i^t) = 1$ and $c^t(i') = 0$ for $i' \neq i^t$. Then the cost of ALG will be T. Since $\exists i \in [n]$ such that i is used $\leq T/n$ times, OPT $\leq T/n$. Thus, Regret $\geq T - T/n = \left(1 - \frac{1}{n}\right) T$. □

4 Consistency Model in Infinite Hypothesis classes

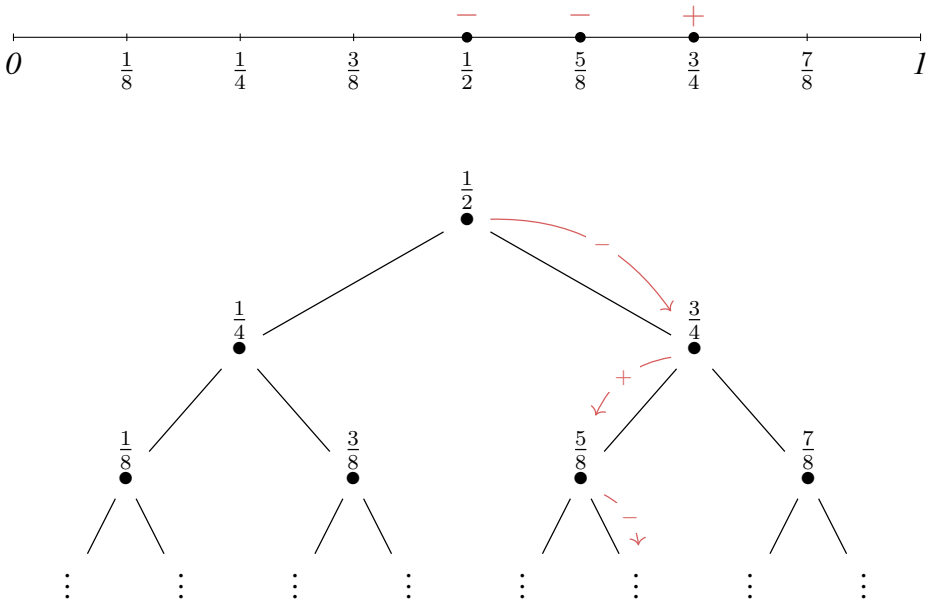
What is analogous to VCDim then? In this section we explore this question. Recall the following example.

Example 4.1. *Consider the case where $\mathcal{C}$ are 1-dim thresholds. That is, $\mathcal{C} = \{h_a \mid a \in \mathbb{R}\}$ with each $h_a = \mathbb{1}(x \geq a)$. This can be visualized as splitting a number line.*



In the finite case, each prediction and label 'reveals' some information about where a is, though it is not hard to see that in the infinite case, the adversary can easily obfuscate a for as long as they want, while ensuring that the learner makes infinite mistakes along the way. Consider the following possible sequence:

- *Adversary gives us $\frac{1}{2}$, we predict 1 but the true label is 0*
- *Adversary gives us $\frac{3}{4}$, we predict 0 but the true label is 1*
- *Adversary gives us $\frac{5}{8}$, we predict 1 but the true label is 0*
- *So on and so forth . . .*



The adversary can always see our prediction, provide the opposite label, and pose the next question based on what information has been revealed in order to cause infinite mistakes. Even simpler, an adversary can choose each label with equal chance, causing every algorithm to have cost $T/2$ in expectation while OPT remains perfect. All of this is true even though VCD$(\mathcal{C}) = 1$. We still have MistakeBound $= \infty$.

This example tells us that finiteness of VCDim is not enough for online learnability. Thus, we proceed by introducing a different notion of combinatorial dimension, the hope being that we can identify concept classes that are learnable in the online and infinite size setting.

Definition 4.2 (Root-to-leaf path)**.** Consider a full binary tree whose nodes are elements of $\mathcal{X}$. Given such a tree of depth d and sequence of binary numbers $(y_1, y_2, \ldots, y_d) \in \{0,1\}^d$, the corresponding root-to-leaf path is a sequence $(x_1, \ldots, x_d)$ that is generated by traversing this tree. That is, x_t is the root of the current tree, we then recursively take the right subtree of if $y_t = -1$, and the left child if $y_t = +1$, and continue to $t+1$.

Definition 4.3 (Shattering a tree)**.** Consider a full binary tree of depth d, where each node is $x \in \mathcal{X}$. This tree is shattered by concept class $\mathcal{C}$ if for all $(y_1, \ldots, y_d) \in \{0,1\}^d$, the root-to-leaf path $(x_1, \ldots, x_d)$ defined by $(y_1, \ldots, y_d)$ has the following property:

$$\exists h \in \mathcal{C} \text{ such that } \forall i \in [d] \text{ it is the case that } h(x_i) = y_i.$$

Definition 4.4 (Littlestone Dimension of a class $\mathcal{C}$)**.** The maximal depth of a tree that can be shattered by $\mathcal{C}$ is $\mathrm{LDim}(\mathcal{C})$, the Littlestone dimension.

Based on the definition of Littlestone dimension, we have the following easily-provable facts.

Fact 4.5. *For all $\mathcal{C}$, LDim$(\mathcal{C}) \leq \log(|\mathcal{C}|)$.*

Proof. Shattering a depth d tree implies all 2^d root-to-leaf paths can be labeled, so $|\mathcal{C}| \geq 2^d$. □

Fact 4.6. *LDim$(\mathcal{C}) \geq$ VCDim$(\mathcal{C})$*

Proof. Let the set (not the tree!) $S = \{x_1, \ldots, x_n\}$ be shattered by $\mathcal{C}$. Then construct a tree where at each depth j, all nodes are x_j. This tree is also shattered by $\mathcal{C}$, since every root-to-path leaf refers to $x_1, \ldots, x_n$. □

5 Standard Optimal Algorithm

Using our newly-defined Littlestone dimension we can state and later prove the following theorem. We can also fill out Table 1, as shown in Table 2.

Theorem 5.1. *In the consistency model, there is an algorithm with MistakeBound $\leq \mathcal{O}($LDim$(\mathcal{C}))$.*

Table 2: Summary of Online vs. Offline (filled in)

	Sample Complexity	**error / Regret**	
		finite $\mathcal{C}$	**infinite $\mathcal{C}$**
Offline	$m \in \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(d + \ln\frac{1}{\delta}\right)\right)$	$\varepsilon \leq \mathcal{O}\left(\sqrt{m \ln \lvert\mathcal{C}\rvert}\right)$	$\varepsilon \leq \mathcal{O}\left(\sqrt{m\,\mathrm{VCD}(\mathcal{C})}\right)$
Online		$\mathcal{O}\left(\sqrt{T \ln \lvert\mathcal{C}\rvert}\right)$	$\widetilde{\mathcal{O}}\left(\sqrt{T \cdot \mathrm{LDim}(\mathcal{C})}\right)$

The idea behind the algorithm in Theorem 5.1 is similar to MAJ. We define some set S^t containing the hypotheses that have been perfect so far, and split it into S^t_{+1} and S^t_{-1} based on the prediction of the hypothesis on x_t. We predict based on whichever set between S^t_{+1} or S^t_{-1} has larger Littlestone dimension. The goal will be to show that whenever the algorithm makes a mistake, then $\text{LDim}(S^{t+1}) \leq \text{LDim}(S^t) - 1$. More formally, we write this out as Algorithm 1.

Algorithm 1 The Standard Optimal Algorithm (SOA)

Initialize $S^1 \leftarrow \mathcal{C}$ ▷ all hypotheses are valid to start
for $t = 1 \to T$ **do**
  $S^t \leftarrow \{h \in \mathcal{C} \mid \forall \tau < t, h(x^\tau) = y^\tau\}$
  $S^t_{+1} \leftarrow \{h \in S^t \mid h(x^t) = +1\}$
  $S^t_{-1} \leftarrow \{h \in S^t \mid h(x^t) = -1\}$
  predict $\hat{y} = \text{argmax}_{r \in \{\pm 1\}} \text{LDim}(S^t_r)$
  observe y^t
  $S^{t+1} \leftarrow S^t_{y_t}$ ▷ update the set of valid hypothesis with newly observed label
end for

Proof. We want to show that if the algorithm makes a mistake on t, then $\text{LDim}(S^{t+1}) < \text{LDim}(S^t)$. Thus, assume that $\hat{y}^t \neq y^t$. Then,

$$\begin{aligned} \text{LDim}\left(S^t\right) &\geq \text{LDim}\left(S^t_{\hat{y}^t}\right) && \text{by monotonicity of LDim} \\ &\geq \text{LDim}\left(S^t_{y^t}\right) && \text{by the definition of SOA} \\ &= \text{LDim}\left(S^{t+1}\right) && \text{by the definition of SOA} \end{aligned}$$

Assume for the sake of contradiction that the inequalities above are in fact all equalities.

$$\text{LDim}\left(S^t\right) = \text{LDim}\left(S^{t+1}\right) = \underbrace{\text{LDim}\left(S^t_{+1}\right) = \text{LDim}\left(S^t_{-1}\right)}_{:=d}$$

However, if $\text{LDim}\left(S^t_{+1}\right) = \text{LDim}\left(S^t_{-1}\right) = d$, then $\text{LDim}\left(S^t\right) = d + 1$, since S^t is simply the concatenation of left subtree S^t_{+1} and right subtree S^t_{-1} with common root x^t. This implies that one of the inequalities must be strict, so

$$\text{LDim}\left(S^t\right) > \text{LDim}\left(S^{t+1}\right).$$

□