Machine Learning Theory (CSC 482A/581A) - Lecture 25

Nishant Mehta

1 Decision-theoretic online learning

Consider a game in which, every morning, you go to a horse racetrack with \$1000 to bet on horses. Not knowing much about horse races, but having access to K experts who can place bets for you, you decide to divide up your \$1000 among the experts each day. After the races for the day are over and the outcomes have been determined, each expert's bet based on your initial investment with them has grown or shrunk into a new quantity, and this new quantity is returned to you. If there is an expert that does particularly well over all the days, in hindsight you naturally wish that you had always placed all of your money with the expert on each day. The amount of regret that you feel is the difference between the amount of money you end up with and the amount of money you would have ended up with had you always gone with that best expert.

In *decision-theoretic online learning*, Learner has a fixed set of K actions. In each round, Learner selects a probability distribution $\boldsymbol{p}$ over the K actions, each action $j \in [K]$ incurs some loss $\ell_{j,t}$ in a range $[0, 1]$, and Learner suffers a loss equal to $\boldsymbol{p}_t \cdot \boldsymbol{\ell}_t = \sum_{j=1}^{K} p_{j,t} \ell_{j,t}$, i.e. the expected loss of the experts according to $j \sim \boldsymbol{p}$. The goal of Learner is to ensure that its regret over T rounds is small, where the regret is defined as the amount by which Learner's cumulative loss exceeds the cumulative loss of the best action. Formally, the protocol is as follows:

Protocol:

For round $t = 1, 2, \ldots$

1. Learner plays probability distribution $\boldsymbol{p}_t$ over $[K]$.
2. Nature plays loss vector $\boldsymbol{\ell}_t = (\ell_{1,t}, \ldots, \ell_{K,t})$ and reveals $\boldsymbol{\ell}_t$ to Learner.
3. Each expert $j \in [K]$ suffers loss $\ell_{j,t}$ and Learner suffers loss $\boldsymbol{p}_t \cdot \boldsymbol{\ell}_t$.

In sequential prediction problems, the strength of the adversary (Nature) can vary; the adversary can be either:

- *oblivious* - Nature knows which algorithm Learner is using, but the adversary must commit to its entire sequence of loss vectors before Learner takes its first action.
- *non-oblivious* or *adaptive* - In round t, Nature can select the loss vector based on all of Learner's previous actions $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{t-1}$.

We will make no assumptions about Nature: Nature will be a non-oblivious adversary. Moreover, we will even assume that Nature gets to observe $\boldsymbol{p}_t$ before it plays $\boldsymbol{\ell}_t$.

It will be convenient to introduce some notation. For a given online learning algorithm that plays action $\boldsymbol{p}_t$ in round t, let $\hat{\ell}_t = \boldsymbol{p}_t \cdot \boldsymbol{\ell}_t$. We also define cumulative loss variables. For each t and for each $j \in [K]$, define $\hat{L}_t$ and $L_{j,t}$ as

$$\hat{L}_t := \sum_{s=1}^{t} \hat{\ell}_s \qquad\qquad L_{j,t} := \sum_{s=1}^{t} \ell_{j,s} \qquad\qquad .$$

Using this notation, the regret of an online learning algorithm that plays $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_T$ against the sequence of loss vectors $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T$ is

$$\hat{L}_T - \min_{j \in [K]} L_{j,T}.$$

Our interest will be in obtaining an upper bound on the regret that holds for any sequence of loss vectors. That is, we wish to upper bound the worst-case regret of a given learning strategy:

$$\max_{\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T} \left\{ \hat{L}_T - \min_{j \in [K]} L_{j,T} \right\}.$$

Since all the losses are in the interval $[0, 1]$, it is trivial to obtain regret that grows linearly in T. Therefore, our goal will be to obtain a regret bound that is sublinear. An algorithm which obtains sublinear regret often is called a *no-regret* algorithm, the idea being that sublinear regret, when averaged over rounds, vanishes as $T \to \infty$.

2 Hedge

There is a no-regret learning algorithm for decision-theoretic online learning. This algorithm, Hedge, is due to Freund and Schapire. Hedge is also often called *exponential weights*, because it maintains weights over each action, and the weight of an action decays exponentially in the cumulative loss incurred by that action over the previous rounds. This algorithm is related to an earlier algorithm of Vovk (1990) for the game of prediction with expert advice (which we will study next week), and it traces the idea of using exponential weights also to the weighted majority algorithm of Littlestone and Warmuth (1994).

Algorithm 1. HEDGE

```
Given: $\eta > 0$
Set $w_{j,0} = 1$ for $j = 1, \ldots, K$
For $t = 1, \ldots, T$:
    1. Set $p_{j,t} = \frac{w_{j,t-1}}{\sum_{i=1}^{K} w_{i,t-1}}$ for $j = 1, \ldots, K$
    2. Observe loss vector $\boldsymbol{\ell}_t$ from Nature
    3. Suffer loss $\boldsymbol{p}_t \cdot \boldsymbol{\ell}_t$
    4. Set $w_{j,t} = w_{j,t-1} e^{-\eta \ell_{j,t}}$ for $j = 1, \ldots, K$
```

2.1 A first regret bound

Hedge satisfies the following guarantee:

Theorem 1. *Let Hedge be run with learning rate* $\eta = \sqrt{\frac{8 \log K}{T}}$*. Then, for all sequences of loss vectors* $\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_T$*,*

$$\hat{L}_T \leq \min_{j \in [K]} L_{j,T} + \sqrt{\frac{T \log K}{2}}.$$

In order to prove this result, we will use a result called Hoeffding's Lemma, the key supporting lemma for proving Hoeffding's inequality.

Lemma 1 (Hoeffding's Lemma)**.** *Let* X *be a random variable satisfying* $\mathsf{E}[X] = 0$ *and* $a \leq X \leq b$*. Then for any* $\lambda \in \mathbb{R}$*,*

$$\log \mathsf{E}[e^{\lambda X}] \leq \frac{\lambda^2 (b-a)^2}{8}.$$

Proof of Theorem 1. Let $\eta > 0$; we will set it to the value in the theorem statement near the end of the proof.

For $t \in [T]$, define

$$W_t := \sum_{j=1}^{K} w_{j,t}.$$

The proof is in 3 steps.

Step 1: We first show that

$$\log \frac{W_T}{W_0} \geq -\eta \min_{j \in [K]} L_{j,T} - \log K. \tag{1}$$

To see this, observe that for any $j \in [K]$, we have $w_{j,T} = e^{-\eta L_{j,T}}$, and so

$$\log \frac{W_T}{W_0} = \log \left(\sum_{j=1}^{K} e^{-\eta L_{j,T}} \right) - \log K.$$

Now, since a sum of nonnegative terms is lower bounded by their maximum element, we have

$$\begin{aligned} \log \left(\sum_{j=1}^{K} e^{-\eta L_{j,T}} \right) - \log K &\geq \log \left(\max_{j \in [K]} e^{-\eta L_{j,T}} \right) - \log K \\ &= -\eta \min_{j \in [K]} L_{j,T} - \log K. \end{aligned}$$

Step 2: Next, we show that for any $t \in [T]$,

$$\log \frac{W_t}{W_{t-1}} \leq -\eta \, \mathsf{E}_{j \sim p_t}[\ell_{j,t}] + \frac{\eta^2}{8}. \tag{2}$$

Observe that

$$\begin{aligned}\log \frac{W_t}{W_{t-1}} &= \log \frac{\sum_{j=1}^K e^{-\eta L_{j,t}}}{\sum_{j=1}^K w_{j,t-1}} \\ &= \log \frac{\sum_{j=1}^K e^{-\eta L_{j,t-1}} e^{-\eta \ell_{j,t}}}{\sum_{j=1}^K w_{j,t-1}} \\ &= \log \frac{\sum_{j=1}^K w_{j,t-1} e^{-\eta \ell_{j,t}}}{\sum_{j=1}^K w_{j,t-1}} \\ &= \log \mathsf{E}_{j \sim \boldsymbol{p}_t} \left[e^{-\eta \ell_{j,t}} \right].\end{aligned}$$

Rewriting by centering $\ell_{j,t}$ around its expectation (for $j \sim \boldsymbol{p}_t$), the above is equal to

$$-\eta \, \mathsf{E}_{j \sim \boldsymbol{p}_t}[\ell_{j,t}] + \log \mathsf{E}_{j \sim \boldsymbol{p}_t} \left[e^{-\eta(\ell_{j,t} - \mathsf{E}_{j \sim \boldsymbol{p}_t}[\ell_{j,t}])} \right].$$

The second term can be bounded via Hoeffding's Lemma (Lemma 1) since the losses are in $[0, 1]$ (and since shifting the losses does not change the difference $(b-a)$ in Hoeffding's lemma); the above is thus at most

$$-\eta \, \mathsf{E}_{j \sim \boldsymbol{p}_t}[\ell_{j,t}] + \frac{\eta^2}{8}.$$

Step 3: Finally, observe that

$$\log \frac{W_T}{W_0} = \sum_{t=1}^T \log \frac{W_t}{W_{t-1}}.$$

Thus, applying the lower bound from (1) and the upper bound from (2), we have

$$-\eta \min_{j \in [K]} L_{j,T} - \log K \le \log \frac{W_T}{W_0} \le -\eta \sum_{t=1}^T \mathsf{E}_{j \sim p_t}[\ell_{j,t}] + \frac{T\eta^2}{8}.$$

Rewriting and dividing by η, this is equivalent to

$$\hat{L}_T \le \min_{j \in [K]} L_{j,T} + \frac{\log K}{\eta} + \frac{T\eta}{8}.$$

Tuning η by setting it to $\sqrt{\frac{8 \log K}{T}}$ yields the result.

□

2.2 Extension for unknown T

The guarantee in Theorem 1 relies upon tuning the learning rate η with knowledge of the time horizon T. When we do not know T, we can still get a good regret bound by using the "doubling trick". The idea is to run the algorithm in epochs of lengths $2^0, 2^1, \ldots$ until we have hit the time horizon. At the start of an epoch, the algorithm is reset, and the learning rate η is tuned according to the epoch size. Proceeding in this way, we will run the algorithm for epochs $r = 0, 1, \ldots, N$, where $N = \lceil \log_2(T + 1) \rceil - 1$. Letting $\hat{L}_T$ be the cumulative loss of this strategy, we have the following regret guarantee:

Corollary 1.

$$\hat{L}_T \leq \min_{j\in[K]} L_{j,T} + \frac{\sqrt{2}}{\sqrt{2}-1}\sqrt{\frac{T\log K}{2}} + \sqrt{\frac{\log K}{2}}.$$

Proof. Observe from Theorem 1 that within the r^{th} epoch, we have

$$\sum_{t=2^r}^{2^{r+1}-1} \hat{\ell}_t \leq \min_{j\in[K]} \sum_{j=2^r}^{2^{r+1}-1} \ell_{j,t} + \sqrt{\frac{2^r \log K}{2}}.$$

Thus, over all the epochs, we have

$$\hat{L}_T = \sum_{r=0}^{N} \sum_{t=2^r}^{2^{r+1}-1} \hat{\ell}_t \leq \left(\sum_{r=0}^{N} \min_{j\in[K]} \sum_{j=2^r}^{2^{r+1}-1} \ell_{j,t}\right) + \left(\sum_{r=0}^{N} \sqrt{\frac{2^r \log K}{2}}\right).$$

We bound each of the two terms on the right-hand side in sequence. Clearly,

$$\sum_{r=0}^{N} \min_{j\in[K]} \sum_{j=2^r}^{2^{r+1}-1} \ell_{j,t} \leq \min_{j\in[K]} \sum_{r=0}^{N} \sum_{j=2^r}^{2^{r+1}-1} \ell_{j,t} = \min_{j\in K} L_{j,T}.$$

It remains to bound $\sum_{r=0}^{N} 2^{r/2}$. For this, observe that

$$(2^{1/2} - 2^0) \sum_{r=0}^{N} 2^{r/2} = 2^{\frac{N+1}{2}} - 1;$$

this is easy to see by expanding the summation, yielding a telescoping series. Therefore,

$$\sum_{r=0}^{N} 2^{r/2} = \frac{2^{\frac{N+1}{2}} - 1}{\sqrt{2}-1} = \frac{2^{\frac{\lceil \log_2(T+1)\rceil}{2}} - 1}{\sqrt{2}-1} \leq \frac{\sqrt{2}\sqrt{T+1}-1}{\sqrt{2}-1} \leq \frac{\sqrt{2}\sqrt{T}+\sqrt{2}-1}{\sqrt{2}-1} = \frac{\sqrt{2}\sqrt{T}}{\sqrt{2}-1} + 1.$$

□

References

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

Vladimir Vovk. Aggregating strategies. In *Proceedings of the third annual workshop on Computational learning theory*, pages 371–383. Morgan Kaufmann Publishers Inc., 1990.