Algorithms and Uncertainty, Winter 2023/24 Lecture 18 (5 pages)

No-Regret Learning: Experts

Thomas Kesselheim Last Update: December 9, 2024

Today and also in the following lectures, we will consider online learning against an adversary. Generally the spirit is similar to the problem that we considered last time. There are a couple of actions available out of which we have to determine the best one. The difference is that the rewards or costs are not determined by (unknown) probability distributions but rather by an adversary.

As a motivating example, consider the following question of *binary classification*. You observe a sequence of samples from a data set and you have to classify them as "positive" or "negative". You make the choices one after the other, only after each choice you will get to know the true label for this sample. You can rely on a set of n classifiers that will each tell you their classification.

1 Majority Algorithm

Let us first consider the setting that one of the n classifiers is perfect and never makes any mistakes. The difficulty is: You do not know which one it is.

A very simple and natural approach is the following *Majority* algorithm: Let S be the set of classifiers that have never been wrong so far. Follow the advice of the majority in S (with arbitrary tie breaking).

Observation 18.1. *If there is a perfect classifier, then the Majority algorithm makes at most* $\log_2 n$ *mistakes.*

Proof idea. Every time, the algorithm makes a mistake, at least $|S|/2$ of the classifiers in S are wrong. Therefore, in the following step, S will be at most half the size. As $1 \leq |S| \leq n$ in every step, the claim follows. □

2 Weighted Majority Algorithm

Let us now come to the general setting, in which each classifier makes a mistake every once in awhile. We would like to not make a lot more mistakes than the best among the n classifiers.

Of course, we cannot follow the above Majority rule because the set S will sooner or later be empty. Instead, we maintain for each classifier a weight w_i. Let $w_i^{(1)} = 1$. If classifier i is correct in step t, then $w_i^{(t+1)} = w_i^{(t)}$, otherwise, if it is wrong, reduce it by setting $w_i^{(t+1)} = (1-\eta)w_i^{(t)}$, where $0 < \eta \leq 1/2$ is a parameter of the algorithm. Our decision in step t is to follow the *weighted majority* of classifiers.

Theorem 18.2. *Weighted Majority makes at most* $(2+2\eta)\min_i m_i + 2\ln n/\eta$ *mistakes, where* m_i *is the number of mistakes that classifier* i *makes.*

Proof. Let $W^{(t)} = \sum_{i=1}^n w_i^{(t)}$ be the sum of weights in step t. Note that $W^{(t)}$ never increases as weights are only reduced. By the change of $W^{(t)}$, we can estimate how many mistakes Weighted Majority makes.

Consider a fixed step t, in which Weighted Majority makes a mistake. Let $U \subseteq [n]$ be the set of classifiers that are incorrect. Then, by definition $\sum_{i\in U} w_i^{(t)} \geq \sum_{i\notin U} w_i^{(t)}$, or equivalently $\sum_{i\in U} w_i^{(t)} \geq \frac{1}{2}\sum_i w_i^{(t)} = \frac{1}{2}W^{(t)}$.

For all $i \in U$, the algorithm updates the weight $w_i^{(t+1)} = (1-\eta)w_i^{(t)}$. For $i \notin U$, we have $w_i^{(t+1)} = w_i^{(t)}$. Therefore,

$$W^{(t+1)} = \sum_{i\in U} w_i^{(t+1)} + \sum_{i\notin U} w_i^{(t+1)} = \sum_{i\in U}(1-\eta)w_i^{(t)} + \sum_{i\notin U} w_i^{(t)} = W^{(t)} - \eta\sum_{i\in U} w_i^{(t)} \leq \left(1-\frac{\eta}{2}\right) W^{(t)} \ .$$

Let M be the number of mistakes that the algorithm makes within the first T steps. By this observation, we have $W^{(T+1)} \leq \left(1-\frac{\eta}{2}\right)^M W^{(1)} = \left(1-\frac{\eta}{2}\right)^M n$.

Let m_i be the number of mistakes that classifier i makes within the first T steps. The algorithm is defined to set $w_i^{(T+1)} = (1-\eta)^{m_i} w_i^{(1)} = (1-\eta)^{m_i}$. Also $W^{(T+1)} \geq w_i^{(T+1)}$.

Combining these two bounds, we get

$$(1-\eta)^{m_i} \leq W^{(T+1)} \leq \left(1-\frac{\eta}{2}\right)^M n \ .$$

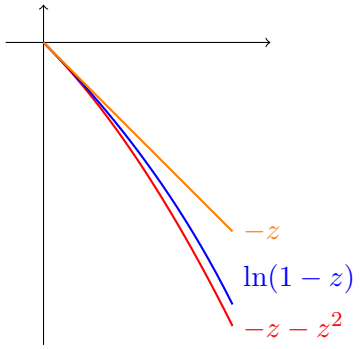
Let us take the logarithm on both sides

$$m_i \ln(1-\eta) \leq M \ln\left(1-\frac{\eta}{2}\right) + \ln n \ .$$

In order to simplify this bound, we will use the following approximation of the logarithm:

$$-z - z^2 \leq \ln(1-z) \leq -z \ , \tag{1}$$

which holds for every $z \in [0, \frac{1}{2}]$.



Therefore

$$m_i(-\eta - \eta^2) \leq M\left(-\frac{\eta}{2}\right) + \ln n \ ,$$

or equivalently

$$M \leq (2+2\eta)m_i + 2\frac{\ln n}{\eta} \ . \qquad \square$$

So, we can hope to make about twice as many errors as the best classifier. Now, we will see that we can actually do much better by using randomization. We will replace every 2 in this formula by 1, meaning that asymptotically we only make as many errors as the best classifier.

3 Randomized Weighted Majority and the Experts Setting

The *Randomized Weighted Majority* algorithm (Littlestone and Warmuth, 1994) maintains weights $w_i^{(t)}$ exactly as Weighted Majority, meaning that $w_i^{(t+1)} = w_i^{(t)}$ if the classifier was correct and $w_i^{(t+1)} = (1-\eta)w_i^{(t)}$ if it was wrong. Instead of using a majority vote, we now interpret these weights as a probability distributions and choose classifier i with probability $p_i^{(t)} = w_i^{(t)} / \sum_{i'=1}^n w_{i'}^{(t)}$.

As a matter of fact, this approach immediately works in a more general setting.

The Experts Setting

Instead of having binary classifiers, we now have arbitrary *experts*, who give us a piece of advice for every round. We choose one of these experts and follow her advice. In particular, the advice could simply be the positive or negative label. Afterwards, we get to know how good each of these experts performed in this round.

More formally, we will consider a sequence of cost vectors $(\ell_i^{(t)})_{i\in[n],t\in[T]}$, $\ell_i^{(t)} \in [0,1]$ for all i and t. In step t, we choose an expert I_t at random, then we get to know $\ell_1^{(t)}, \dots, \ell_n^{(t)}$ and incur costs $\ell_{I_t}^{(t)}$. The classification setting is recovered by setting $\ell_i^{(t)} = 0$ if classifier i is correct in step t and $\ell_i^{(t)} = 1$ if it is wrong.

The algorithm *Multiplicative Weights* or *Hedge* generalizes Randomized Weighted Majority as follows. Again, $\eta \in (0, \frac{1}{2}]$ is a parameter of the algorithm to be chosen later.

- Initially, set $w_i^{(1)} = 1$, for every $i \in [n]$.
- At every time t
  - Let $W^{(t)} = \sum_{i=1}^n w_i^{(t)}$
  - Choose expert i with probability $p_i^{(t)} = w_i^{(t)}/W^{(t)}$
  - Set $w_i^{(t+1)} = w_i^{(t)} \cdot (1-\eta)^{\ell_i^{(t)}}$ for all experts i

So, in the special case of $\ell_i^{(t)} \in \{0,1\}$, note that $w_i^{(t+1)} = w_i^{(t)}$ if $\ell_i^{(t)} = 0$ and $w_i^{(t+1)} = w_i^{(t)} \cdot (1-\eta)$ if $\ell_i^{(t)} = 1$ just as in (Randomized) Weighted Majority.

Theorem 18.3. *Multiplicative Weights, for any sequence of cost vectors from* $[0,1]$*, guarantees that for all experts* i

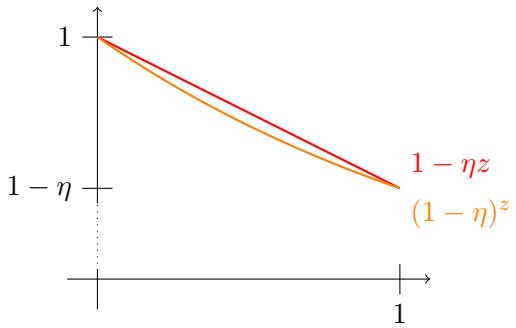
$$L_{Alg}^{(T)} \le (1+\eta)L_i^{(T)} + \frac{\ln n}{\eta} \ ,$$

where $L_i^{(T)} = \sum_{t=1}^T \ell_i^{(t)}$ *is the sum of costs of expert* i *and* $L_{Alg}^{(T)} = \sum_{t=1}^T \sum_{i=1}^n p_i^{(t)} \ell_i^{(t)}$ *is the expected sum of costs of the multiplicative-weights algorithm.*

Proof. Let us analyze how the sum of weights $W^{(t)}$ decreases over time. It holds

$$W^{(t+1)} = \sum_{i=1}^n w_i^{(t+1)} = \sum_{i=1}^n w_i^{(t)} (1-\eta)^{\ell_i^{(t)}} \ .$$

Observe that $(1-\eta)^z = (1-z\eta)$, for both $z = 0$ and $z = 1$. Furthermore, $z \mapsto (1-\eta)^z$ is a convex function in z. For $z \in [0,1]$ this implies $(1-\eta)^z \le 1 - z\eta$.



This gives us

$$W^{(t+1)} \leq \sum_{i=1}^{n} w_i^{(t)}(1-\ell_i^{(t)}\eta) = W^{(t)} - \eta \sum_{i=1}^{n} w_i^{(t)}\ell_i^{(t)} \ .$$

Let $\ell_{\text{Alg}}^{(t)}$ denote the expected cost of the algorithm in step t. The expected cost of the algorithm $\ell_{\text{Alg}}^{(t)}$ is given by $\ell_{\text{Alg}}^{(t)} = \sum_{i=1}^{n} \ell_i^{(t)} w_i^{(t)}/W^{(t)}$. Substituting this into the bound for $W^{(t+1)}$ gives

$$W^{(t+1)} \leq W^{(t)} - \eta\ell_{\text{Alg}}^{(t)} W^{(t)} = W^{(t)}(1-\eta\ell_{\text{Alg}}^{(t)}) \ .$$

As a consequence,

$$W^{(T+1)} \leq W^{(1)} \prod_{t=1}^{T}(1-\eta\ell_{\text{Alg}}^{(t)}) = n\prod_{t=1}^{T}(1-\eta\ell_{\text{Alg}}^{(t)}) \ .$$

The sum of weights after step T can be upper bounded in terms of the expected costs of the algorithm. On the other hand, the sum of weights after step T can be lower bounded in terms of the costs of the best expert as follows:

$$W^{(T+1)} \geq w_i^{(T+1)} = \left(w_i^{(1)} \prod_{t=1}^{T}(1-\eta)^{\ell_i^{(t)}} \right) = \left((1-\eta)^{\sum_{t=1}^{T}\ell_i^{(t)}} \right) = (1-\eta)^{L_i^{(T)}} \ .$$

Combining the bounds and taking the logarithm on both sides gives us

$$L_i^{(T)} \ln(1-\eta) \leq (\ln n) + \sum_{t=1}^{T} \ln(1-\eta\ell^{(t)}) \ .$$

Applying Equation (1), we get

$$\begin{aligned} L_i^{(T)}(-\eta-\eta^2) &\leq (\ln n) + \sum_{t=1}^{T}(-\eta\ell^{(t)}) \\ &= (\ln n) - \eta L_{\text{Alg}}^{(T)} \ . \end{aligned}$$

Finally, solving for $L_{\text{Alg}}^{(T)}$ gives

$$L_{\text{Alg}}^{(T)} \leq (1+\eta)L_i^{(T)} + \frac{\ln n}{\eta} \ . \quad \square$$

Note that setting $\eta = \sqrt{\frac{\ln n}{T}}$ yields

$$L_{\text{Alg}}^{(T)} \leq \min_i L_i^{(T)} + 2\sqrt{T \ln n} \ .$$

We call $\text{Regret}^{(T)} = L_{\text{Alg}}^{(T)} - \min_i L_i^{(T)}$ the (external) regret of the algorithm on the sequence. An algorithm that guarantees that for any sequence $\text{Regret}^{(T)} = o(T)$ is called a no-external-regret algorithm.

Corollary 18.4. *The multiplicative weights algorithm with $\eta = \sqrt{\frac{\ln n}{T}}$ has external regret at most $2\sqrt{T \ln n} = o(T)$ and hence is a no-external-regret algorithm.*

4 Extensions

Instead of setting the update to $w_i^{(t+1)} = w_i^{(t)} \cdot (1-\eta)^{\ell_i^{(t)}}$, it is also common to update the weights by $w_i^{(t+1)} = w_i^{(t)} \cdot \mathrm{e}^{-\eta' \ell_i^{(t)}}$. Indeed, this is the same algorithm using only a different parameterization, namely setting $\eta = 1 - \mathrm{e}^{-\eta'}$; there is a one-to-one correspondence between η and η'. Sometimes, the former parameterization is easier to work with, sometimes the latter.