Online Learning and the Multiplicative Weights Algorithm

Online Decision-Making

At each time step $t = 1, 2, \ldots, T$:

a decision-maker picks a probability distribution $\mathbf{p}^t$ over her experts or actions $i = 1, \ldots, N$

an adversary picks a **loss** vector $\ell^t : A \to [-1, 1]$

an action i^t is chosen according to the distribution $\mathbf{p}^t$, and the decision-maker receives loss ℓ_i^t

the decision-maker learns ℓ^t, the entire **loss** vector

The input arrives "one piece at a time."

What should we compare to?

Thus far, we've been trying to achieve optimal solutions, or comparing to optimal solutions assuming we know full information about the future and what is optimal. Does that still make sense?

Example 1 (Comparing to the Best Action Sequence). Suppose your set of experts (or actions) is $A = \{1, 2\}$. Each day t, the adversary chooses the loss vector ℓ^t as follows: if the algorithm chooses a distribution $\mathbf{p}^t$ for which the probability on action 1 is at least $\frac{1}{2}$, then ℓ^t is set to the vector $(1, 1)$. Otherwise, the adversary sets ℓ^t equal to $(1, -1)$.

This adversary forces the expected loss of the algorithm to be nonpositive, while ensuring that the loss of the best action sequence in hindsight is T. Thus, the algorithm's approximation is x/T where $x \leq 0$—no approximation at all.

Example 1 tells us that we should not be trying to compare to the Best Action Sequence—this is too strong of a goal. Instead, we compare it to the loss incurred by the best *fixed action* in hindsight. In words, we change our benchmark from

$$\sum_{t=1}^{T} \min_{i=1}^{N} \ell_i^t \quad \text{to} \quad \min_{i=1}^{N} \sum_{i=1}^{T} \ell_i^t.$$

Definition 1 (Regret). Fix loss[1] vectors $\ell^1, \ldots, \ell^T$. The regret of the action sequence $a^1, \ldots, a^T$ is

$$\underbrace{\sum_{t=1}^{T} \ell^t(a^t)}_{\text{our algorithm}} - \underbrace{\min_{i=1}^{N} \sum_{t=1}^{T} \ell_i^t}_{\text{best fixed action}} . \tag{1}$$

- We'd like an online decision-making algorithm that achieves low regret, as close to 0 as possible. Notice that the worst-possible regret in T (since losses lie in $[0,1]$).

- We think of regret $\Omega(T)$ as an epic fail for an algorithm.

Example 2 (Randomization Is Necessary for No Regret)**.**

Example 3 ($\sqrt{(\ln n)/T}$ Regret Lower Bound)**.**

The Multiplicative Weights Algorithm

Formally, we want an algorithm that works in the following framework:

1. In rounds $1, \ldots, T$, the algorithm chooses some expert i^t.

2. Each expert i experiences a loss $\ell_i^t \in [0,1]$. The algorithm experiences the loss of the expert it chooses: $\ell_A^t = \ell_{i^t}^t$.

3. The total loss of expert i is $L_i^T = \sum_{t=1}^{T} \ell_i^t$, and the total loss of the algorithm is $\mathbb{E}_{\mathbf{p}}[L_{MWU}] = \mathbb{E}_{\mathbf{p}}\left[\sum_{t=1}^{T} \ell_{i^t}^t\right]$. The goal of the algorithm is to obtain loss not much worse than that of the best expert: $\min_i L_i^T$.

---

[1]Note that for the rewards setting, the definition of regret would instead be $\max_{i=1}^{N} \sum_{t=1}^{T} r_i^t - \sum_{t=1}^{T} r^t(a^t)$, still minimizing the difference between the algorithm and the best fixed action, but now the maximum reward for the best fixed action will be larger than the algorithm instead of the minimum loss being smaller.

Multiplicative Weights (MW) Algorithm

initialize weights $w_i^1 = 1$ for every expert $i = 1 \ldots, N$
for each time step $t = 1, 2, \ldots, T$ **do**

    let $W^t = \sum_{i=1}^{N} w_k^t$ be the sum of the weights

    choose expert k with probability $p_k^t = w_k^t / W^t$

    for each expert k, update weights

$$w_k^{t+1} = w_k^t \cdot (1 - \varepsilon\, \ell_k^t)$$

Theorem 1. *For any sequence of losses, over the randomness of our algorithmic choices* $\mathbf{p}$,

$$\mathbb{E}_{\mathbf{p}}[\text{REGRET}_{\text{MWU}}] \le 2\sqrt{\ln(N)T} + \varepsilon T.$$

That is, for any expert k

$$\frac{1}{T}\mathbb{E}_{\mathbf{p}}\left[\sum_{t=1}^{T} \ell_{it}^t\right] \le \frac{1}{T}\left[\sum_{t=1}^{T} \ell_k^t\right] + \varepsilon + \frac{\ln(N)}{\varepsilon \cdot T}$$

In particular, by setting $\varepsilon = \sqrt{\frac{\ln(N)}{T}}$ *we get:*

$$\frac{1}{T}\mathbb{E}_{\mathbf{p}}\left[\sum_{t=1}^{T} \ell_{it}^t\right] - \frac{1}{T}\left[\min_{i=1}\sum_{t=1}^{T} \ell_i^t\right] \le \varepsilon + 2\sqrt{\frac{\ln(N)}{T}}.$$

In other words, the average loss of the algorithm quickly approaches the average loss of the best expert exactly, at a rate of $1/\sqrt{T}$. Note that this works against an arbitrary sequence of losses, which might be chosen adaptively by an adversary. This is pretty incredible. And it will be the source of the power of this framework in applications: we (the algorithm designer) can play the role of the adversary to get the results that we want.

Corollary 2. *There is an online decision-making algorithm that, for every adversary and* $\varepsilon > 0$, *has expected time-averaged regret*[2] *at most* ε *after at most* $(4\ln n)/\varepsilon^2$ *time steps.*

Recap of notation:

- N: the number of experts (actions)

- i, k: index of a specific expert (action)

---

[2]Time-averaged regret just means the regret, divided by T.

- w: weights assigned to experts, a vector for each expert, indexed for each time step t and expert i

- W^t: the sum over all experts of weights at time t—$W^t = \sum_{i=1}^{N} w_i^t$.

- p: a probability distribution over experts, indexed for each time step t and expert i, equal to weights w normalized by the sum W—$p_i^t = w_i^t / W^t$.

- ε: update parameter

- ℓ: adversary's loss assignments for each time step and expert, $\ell_i^t \in [0, 1]$.

- F: *expected* loss. $F^t = \sum_{i=1}^{N} p_i^t \ell_i^t$.

Proof. Let F^t denote the expected loss of the MWU algorithm at time t. By linearity of expectation, we have $\mathbb{E}[L_{MWU}^T] = \sum_{t=1}^{T} F^t$. We also know that:

$$F^t = \tag{2}$$

Thus we want to lower bound the sum of the F^t's.

How does W^t change between rounds? We know that $W^1 = N$, and looking at the algorithm, we derive W^{t+1} as a function of W^t and the expected loss (2)

$$W^{t+1} =$$

So by induction, we can write:

$$W^{T+1} =$$

Taking the log, and using the fact that $\ln(1 - x) \leq -x$, we can write:

$$\ln(W^{T+1}) =$$

Similarly, we can unroll the update rule for our weights

$$w_k^{t+1} =$$
(3)

(using the fact that $\ln(1 - x) \geq -x - x^2$ for $0 < x < \frac{1}{2}$), we know that for every expert k:

$$\ln(W^{T+1})$$

Combining these two bounds, we get that for all k:

for all k. Dividing by ε and rearranging, we get:

□