Notes on the Multiplicative Weights Algorithm

1 General Setup

Imagine that you have n experts that, each day, provide you with advice. Each day, you choose which one to believe, possibly by making a probabilistic choice, that is, on day t you choose to follow the advice of expert i with probability $x_i^{(t)}$. Later, you find out which expert was right and which one was wrong. How many times, on average, you follow the wrong advice? We will see an algorithm that performs approximately as well as knowing in advance the best of the n experts and always following his advice.

Suppose that you are day-trading, and each day you have to decide how to allocate a fixed amount of capital among n stocks (assume you always invest the same amount, so if you lose one day you will dip into your savings to make up for it the next day). Call $x_i^{(t)}$ the fraction of your capital that you allocate to stock i on day t and call $\ell_i^{(t)}$ the loss (which could be negative, and hence be a gain) that you would incur on day t if you invested everything on stock i. Then at the end of day t you have lost $\sum_i x_i^{(t)}\ell_i^{(t)}$, and over T days you have lost $\sum_{t=1}^{T}\sum_{i=1}^{n} x_i^{(t)}\ell_i^{(t)}$. We will see an algorithm that performs approximately as well as knowing in advance the best stock and always trading it.

More abstractly, we consider the following *online optimization* setup: on each time step $t = 1, \ldots$ we can choose among n possible strategies, and we are required to produce an array $x^{(t)} = (x_1^{(t)}, \ldots, x_n^{(t)})$ where $x_i^{(t)} \geq 0$ and $\sum_{i=1}^{n} x_i^{(t)} = 1$. We can interpret $x_i^{(t)}$ as saying with what probability we follow strategy i, or what fraction of our budget we invest in strategy i. After we make our choice of $x^{(t)}$, an array $\ell^{(t)} = (\ell_1^{(t)}, \ldots, \ell_n^{(t)})$ is revealed, where $\ell_i^{(t)}$ specifies the *loss* that we incur when we follow strategy i. Consequently, our loss at time t is

$$\sum_{i=1} x_i^{(t)} \cdot \ell_i^{(t)}$$

After T steps, we define our *regret* R_T at time T as being the difference between the overall loss incurred at the time steps $1, \ldots, T$, minus the loss incurred by the best *fixed* allocation strategy, that is

$$R_T = \sum_{t=1}^{T}\sum_{i=1}^{n} x_i^{(t)} \cdot \ell_i^{(t)} \; - \min_{x \text{ probability distribution}} \quad \sum_{t=1}^{T}\sum_{i=1}^{n} x_i \cdot \ell_i^{(t)}$$

We will use Δ_n to refer to the set of all probability distributions over n elements (that is, the set of all non-negative n-dimensional vectors whose entries sum to one), and we will write Δ without subscript if n is clear from the context. Notice that we have

$$\min_{x\in\Delta} \quad \sum_{t=1}^{T}\sum_{i=1}^{n} x_i \cdot \ell_i^{(t)} = \min_{x\in\Delta} \quad \sum_{i=1}^{n} x_i \cdot \left(\sum_{t=1}^{T} \ell_i^{(t)}\right) = \min_{i=1,\ldots,n} \; \sum_{t=1}^{T} \ell_i^{(t)} \tag{1}$$

because to minimize the average of n numbers over all probability distributions over those n numbers, it is optimal to pick the probability distribution that puts all the probability on the smallest number. We will refer to the quantity in (1) as the *offline optimum*.

In the following, we will assume $0 \leq \ell_i^{(t)} \leq 1$. If the losses are in a bigger range, we will see later that we can scale them and shift them so that they are between 0 and +1, and then apply the algorithm that we will describe below to the scaled and shifted losses.

We will describe and analyze an algorithm, called the *Multiplicative Weight Algorithm* (or the *Hedge Algorithms*) that guarantees

$$R_T \leq 2\sqrt{T \ln n}$$

If we look at the regret-per-time-step achieved by the algorithm, we see that it is $\frac{R_T}{T} = O\left(\sqrt{\frac{\ln n}{T}}\right)$, which goes to zero with T.

The algorithm proceeds as follows. It works with a parameter $0 \leq \epsilon \leq 1/2$ that is hardcoded in the algorithm, and it maintains, at each time step t a set of *weights* $w^{(t)} = (w_1^{(t)}, \ldots, w_n^{(t)})$ associated to the strategies, defined as

$$w_i^{(0)} = 1$$

$$w_i^{(t+1)} = w_i^{(t)} \cdot (1-\epsilon)^{\ell_i^{(t)}}$$

The effect of this definition is that strategies that produced high losses in the past have small weight, and strategies that produces low losses have higher weight. At every step, the algorithm chooses allocations proportionally to the weights

$$x_i^{(t)} = \frac{w_i^{(t)}}{\sum_j w_j^{(t)}}$$

This completes the description of the algorithm. We will provide the following analysis

THEOREM 1
Assuming that all losses are in the range $[0,1]$*, the regret of the Multiplicative Weight Algorithm run for* T *steps with a parameter* $0 < \epsilon \leq 1/2$ *is*

$$R_T \leq \epsilon T + \frac{\ln n}{\epsilon}$$

In particular, if $T > 4 \ln n$ *and we choose* $\epsilon = \sqrt{\frac{\ln n}{T}}$*, we have*

$$R_T \leq 2\sqrt{T \ln n}$$

2 Analysis

The outline of the proof of Theorem 1 is as follows. Define

$$W_t := \sum_{i=1}^{n} w_i^{(t)}$$

to be the sum of the weights at time t. We will show that, at the end of the execution of the algorithm:

- If W_{T+1} is small, then the offline optimum is large;
- If the loss suffered by the algorithm is large, then W_{T+1} is small.

Together, the above two facts implies that if the algorithm suffers a large loss, then the offline optimum is also large, so that the regret is small. The proof of the first fact (see Lemma 2) will rely on the fact that if the offline optimum is small, then there is an i^* such that $\sum_{t=1}^{T} \ell_{i^*}^{(t)}$ is small, and the corresponding weight $w_i^{(T+1)}$ is large. The proof of second fact (see Lemma 3) is intuitively based on the fact that if there is a large loss for the algorithm at time t, it is because some strategy i has large allocation $x_i^{(t)}$ and large loss $\ell_i^{(t)}$; but then at the next step a previously large weight $w_i^{(t)}$ becomes a smaller weight $w_i^{(t)} \cdot (1-\epsilon)^{\ell_i^{(t)}}$. Thus, a sequence of large losses bring down the value of W_t a lot.

Of course the above discussion is informal and qualitative, but we will turn this intuition into formal and quantitative proofs. Although the proofs will look just like a sequence of manipulations of inequalities, the above intuition will shed some light on the logic underlying those manipulations.

It will be helpful to give short names to quantities that we will encounter frequently. We will call the offline optimum L^*:

$$L^* := \min_{x \in \Delta} \quad \sum_{t=1}^{T} \sum_{i=1}^{n} x_i \cdot \ell_i^{(t)} = \min_{i=1,\ldots,n} \quad \sum_{t=1}^{T} \ell_i^{(t)}$$

and we will use L_t to denote the loss incurred by the algorithm at time t

$$L_t := \sum_{i=1}^{n} x_i^{(t)} \ell_i^{(t)}$$

Lemma 2 (If W_{T+1} is small then the offline optimum is large)

$$W_{T+1} \geq (1-\epsilon)^{L^*}$$

Proof: Let j be any strategy, then

$$W_{T+1} = \sum_{i=1}^{n} w_i^{(T+1)} \geq w_j^{(T+1)} = \prod_{t=1}^{T} (1-\epsilon)^{\ell_j^{(t)}} = (1-\epsilon)^{\sum_{t=1}^{T} \ell_j^{(t)}}$$

Now let i^* be an offline optimal strategy, that is, let i^* be a strategy such that

$$L^* = \sum_{t=1}^{T} \ell_{i^*}^{(t)}$$

and observe that

$$W_{T+1} \geq (1-\epsilon)^{\sum_{t=1}^{T} \ell_{i^*}^{(t)}} = (1-\epsilon)^{L^*}$$

□

Lemma 3 (If the loss suffered by the algorithm is large, then W_{T+1} is small)

$$W_{T+1} \leq n \cdot \prod_{t=1}^{T} (1 - \epsilon L_t)$$

Proof: We know that $W_1 = n$, so to prove the lemma it will be enough to prove that

$$W_{t+1} \leq W_t \cdot (1 - \epsilon L_t) \tag{2}$$

for every $t = 1, \ldots, T$ because then the statement of the lemma will follow by induction. To prove (2), observe that

$$\begin{aligned}
W_{t+1} &= \sum_{i=1}^{n} w_i^{(t+1)} && (3)\\
&= \sum_{i=1}^{n} w_i^{(t)} \cdot (1-\epsilon)^{\ell_i^{(t)}} && (4)\\
&\leq \sum_{i=1}^{n} w_i^{(t)} \cdot (1 - \epsilon \cdot \ell_i^{(t)}) && (5)\\
&= W_t \sum_{i=1}^{n} x_i^{(t)} \cdot (1 - \epsilon \cdot \ell_i^{(t)}) && (6)\\
&= W_t \cdot (\sum_{i=1}^{n} x_i^{(t)} - \sum_{i=1}^{n} \epsilon \cdot x_i^{(t)} \cdot \ell_i^{(t)}) && (7)\\
&= W_t \cdot (1 - \epsilon L_t) && (8)
\end{aligned}$$

In step (5) we used the inequality

$$(1-\epsilon)^z \leq 1 - \epsilon z$$

which is true for all $0 \leq z \leq 1$ and $0 \leq \epsilon \leq 1$, and in step (6) we used the fact that $x_i^{(t)} = w_i^{(t)} / W_t$. □

Now we put the two lemmas together and we take logarithms. We have:

$$(1-\epsilon)^{L^*} \leq n \cdot \prod_{t=1}^{T} (1 - \epsilon L_t)$$

and so:

$$L^* \ln(1-\epsilon) \leq \ln n + \sum_{t=1}^{T} \ln(1-\epsilon L_t)$$

Finally, we apply the inequalities

$$-z - z^2 \leq \ln 1 - z \leq -z$$

which are true[1] for every $0 \leq z \leq 1/2$, and we have

$$L^* \cdot (-\epsilon - \epsilon^2) \leq \ln n - \epsilon \cdot \sum_{i=1}^{T} L_t$$

which, rearranging and dividing by ϵ gives us

$$\sum_{i=1}^{T} L_t - L^* \leq \epsilon L^* + \frac{\ln n}{\epsilon} \leq \epsilon T + \frac{\ln}{\epsilon}$$

which proves our theorem. At this point we have to choose ϵ. Our regret bound has a term that is directly proportional to ϵ and a term that is inversely proportional to ϵ. In such cases, the best (smallest) expression is obtain by choosing an ϵ that makes the two terms equal, that is $\epsilon = \sqrt{\frac{\ln n}{T}}$. For that choice of ϵ, we get the regret bound $2\sqrt{T \ln n}$.

3 More General Losses

Suppose that we are in a context in which the losses are not in $[0, 1]$ but are, say, in $[-2, 5]$ or, in general, in a known interval $[a, b]$. Then we can define a new set of losses

$$\ell_i'^{(t)} := \frac{\ell_i^{(t)} - a}{b - a}$$

which are always between a and b, and we can run the algorithm as described in the previous section on these modified losses. This will give us the guarantee

$$\sum_{t=1}^{T} \sum_{i=1}^{n} \ell_i'^{(t)} x_i^{(t)} - \min_{i=1,\ldots,n} \sum_{t=1}^{T} \ell_i'^{(t)} \leq \epsilon T + \frac{\ln n}{\epsilon}$$

which is equivalent to

$$\frac{1}{b-a} \cdot \sum_{t=1}^{T} \sum_{i=1}^{n} \ell_i^{(t)} x_i^{(t)} - \frac{1}{b-a} \min_{i=1,\ldots,n} \sum_{t=1}^{T} \ell_i^{(t)} \leq \epsilon T + \frac{\ln n}{\epsilon}$$

[1] This is because we have the Taylor series

$$\ln 1 - z = \sum_{n=1}^{\infty} -\frac{z^n}{n}$$

that is,

$$\sum_{t=1}^{T}\sum_{i=1}^{n}\ell_i^{(t)}x_i^{(t)} - \min_{i=1,\ldots,n}\sum_{t=1}^{T}\ell_i^{(t)} \leq (b-a)\cdot\left(\epsilon T + \frac{\ln n}{\epsilon}\right)$$

This means that we can achieve a regret, with respect to the original losses $\ell_i^{(t)}$, which is at most $2\cdot(b-a)\cdot\sqrt{T\ln n}$.

4 Zero-Sum Games

In this Section we will prove the Minimax Theorem for two-player zero-sum games and given an algorithm for finding near-optimal strategies.

Suppose that A is the payoff matrix for the first player in a two-player zero-sum game. Now consider what happens if let the two players play repeated games against each other, while both of them are running the multiplicative weight algorithm, using the win/losses of the game as their respective loss vector.

More precisely, at each time step t, player 1 comes up with a probability vector $x^{(t)}$ and player 2 comes up with a probability vector $y^{(t)}$. Then player 1 is notified that his loss vector is $-Ay^{(t)}$, and player 2 is notified that his loss vector is $A^Tx^{(t)}$. Note that is consistent with player 1 having a loss of $-(x^{(t)})^TAy^{(t)}$ and player 2 having a loss of $(x^{(t)})^TAy^{(t)}$. Then the two players will come up with new probability vectors $x^{(t+1)}$ and $y^{(t+1)}$ and so on.

Using big-Oh notation to hide dependencies on the largest and smallest entry of A, and choosing the optimal ϵ, after playing T games Theorem 1 gives us the following regret bounds for the first player

$$\sum_{t=1}^{T}(x^{(t)})^T(-A)y^{(t)} - \min_{x\in\Delta}\sum_{t=1}^{T}x^T(-A)y^{(t)} = O\left(\sqrt{\ln n\cdot T}\right)$$

and the following regret bound for the second player

$$\sum_{t=1}^{T}(x^{(t)})^TAy^{(t)} - \min_{y\in\Delta}\sum_{t=1}^{T}(x^{(t)})^TAy = O\left(\sqrt{\ln n\cdot T}\right)$$

To simplify the expressions above, define

$$\bar{x} := \frac{1}{T}\sum_{t=1}^{T}x^{(t)}$$

and

$$\bar{y} := \frac{1}{T}\sum_{t=1}^{T}y^{(t)}$$

Then we can rewrite the regret bound of the first player as

$$\max_{x\in\Delta}\ x^TA\bar{y} - \frac{1}{T}\sum_{t=1}(x^{(t)})^TAy^{(t)} = O\left(\sqrt{\frac{\ln n}{T}}\right)$$

and the regret bound of the second player as

$$\frac{1}{T}\sum_{t=1}(x^{(t)})^T A y^{(t)} - \min_{y\in\Delta} \; \bar{x}^T A y = O\left(\sqrt{\frac{\ln n}{T}}\right)$$

so, adding up the two bounds, we have

$$\max_{x\in\Delta} \; x^T A\bar{y} \; - \; \min_{y\in\Delta} \; \bar{x}^T A y = O\left(\sqrt{\frac{\ln n}{T}}\right)$$

which also implies

$$\min_{y\in\Delta} \; \max_{x\in\Delta} \; x^T A y \; - \max_{x\in\Delta} \; \min_{y\in\Delta} \; x^T A y = O\left(\sqrt{\frac{\ln n}{T}}\right)$$

We know from weak duality that

$$\min_{y\in\Delta} \; \max_{x\in\Delta} \; x^T A y \; - \max_{x\in\Delta} \; \min_{y\in\Delta} \; x^T A y \geq 0$$

and, noting that the difference does not depend[2] on T, we must have

$$\min_{y\in\Delta} \; \max_{x\in\Delta} \; x^T A y \; - \max_{x\in\Delta} \; \min_{y\in\Delta} \; x^T A y = 0$$

proving strong duality for two-player zero-sum games. This is also called the "Minimax Theorem" of two-player zero-sum games, or "Von Neumann's Minimax Theorem" of two-player zero-sum games.

[2]Do not confuse the symbol for "transpose of a vector" with the number of steps T.