Algoritmos e Incerteza **(PUC-Rio INF2978, 2018.2)**

Lecture 04: Online Learning

September 4, 2018

Lecturer: Marco Molinaro *Scribe: Matheus Werner*

Up until now, we have been studying the Online Algorithms field for the online environment. In this approach we receive new information regarding an instance I at each time t and, in response, make a decision trying to minimize (or maximize) a given objective function on the long run.

An example is to maintain k of n pages in a cache for fast page retrieval, as already seen in the Paging problem. In this problem, the instance I requests a page at each time t, and we are penalized if it is not in the cache. Therefore our objective is to select the pages to keep in the cache while minimizing the total penalty.

From this and other problems already studied, we can observe that: First, our only source of information during decision making comes from the instance itself. Second, usually each time t has a state (e.g: pages in the cache), which limits the possible actions for the next time $t + 1$ and, consequentially, directly impact our objective function value.

In this lecture we are going to study a related approach called Online Learning, which deals with stateless problems like prediction, meaning that our decisions will only affect the current time.

As we will see, problems of this nature are more difficult, forcing us to compare our new algorithms against simpler and less dynamic adversaries. As in Online Algorithms, we will not consider any specific instance distribution, focusing on a worst-case analysis.

Finally, for the remainder of the lecture, we will be presenting the most classical problem of Online Learning called Prediction with Experts.

1 Prediction with experts

Let's say we want to study whether an asset value will increase or decrease in a given time and that we have access to experts recommendation for this prediction. Our objective for this problem then becomes minimizing the number of errors made using these recommendations.

The setup for this is summarized as:

- Binary prediction {'-', '+'}
- Access to experts
- The objective is to minimize the predictions error

Moreover, all the algorithms that we are going to develop are going to follow the Algorithm 1 structure for computing and assessing the predictions made.

It is also important to take note that we focus on how to develop the Line 5 and, consequently, analyze its competitiveness. As such, we can consider that we magically received the functions from Lines 4 and 6.

Algorithm 1 Prediction with experts framework

1: **procedure** ALGO
2: **for** *t = 1, 2,. . . , T* **do**
3: **for** *i = 1, 2,. . . , m* **do**
4: $e_i^t \leftarrow ExpertRecommendation(i, t)$ ▷ expert i recommendation $\rightarrow e_i^t \in \{-1, 1\}$
5: $o^t \leftarrow AlgorithmPrediction(e^t)$ ▷ Algorithm prediction $\rightarrow o^t \in \{-1, 1\}$
6: $r^t \leftarrow InstanceRealization(t)$ ▷ Real outcome $\rightarrow r^t \in \{-1, 1\}$
7: $error \leftarrow [r^t \neq o^t]$ ▷ Prediction error $\rightarrow error \in \{0, 1\}$

For example, with $T = 2$, $m = 2$ and our predictor just following Expert 1 recommendation, we have:

Time	1	2
Expert #1	+	+
Expert #2	-	-
Prediction	+	+
Real	+	-
Error	0	1

Q. Against whom to compare for measuring the competitiveness of our algorithms? The first natural idea is to compare ALGO against OPT (Oracle) similarly to Online Algorithms. However, this comparison is unreasonable for two main reasons:

- OPT has always $TotalError = 0$, where ALGO has $TotalError = T$ for any deterministic algorithm in the worst-case instance;
- ALGO heavily depends on the quality of the experts.

As such, it arises as a second idea to compare ALGO against the best expert available.

$$OPT = \min_i [\#Errors_{i^{th}expert}]$$

Also, since OPT has lost its default meaning in this new context, we will begin to refer OPT as the best expert during the remaining of this lecture.

2 Deterministic algorithms

Now that we have finished defining our problem and its framework, let's take a look in some of the deterministic algorithms using experts.

2.1 Halving algorithm

As a first building block, let's assume that between our m experts, the best one makes 0 errors.

Q. What algorithm can we build taking advantage of this property? If we have access to one expert with perfect recommendations, we would like to use it in all our predictions. The problem is that we do not know beforehand which one of our experts has this property.

A simple algorithm to find it out is to set our prediction to follow the majority recommendation of our enabled experts at each time t. If the majority recommends wrong, we disable all of them since none of them is the expert we are looking for. We continue this procedure until only our best expert remains. The Algorithm 2 summarizes this approach.

Algorithm 2 Halving Algorithm

procedure HALVING ALGO
  $A \leftarrow \{1, 2, \ldots, m\}$ ▷ Enable all experts
  for *t = 1, 2,..., T* **do**
    for *i = 1, 2,..., m* **do**
      $e_i^t \leftarrow ExpertRecommendation(i, t)$
    $o^t \leftarrow$ If $\sum_{i \in A} e_i^t > 0$ Then 1 Else -1 ▷ Follow majority recommendation
    $r^t \leftarrow InstanceRealization(t)$
    $error \leftarrow [r^t \neq o^t]$
    if $r^t \neq o^t$ **then**
      $A \leftarrow A - \{i | e_i^t = o^t\}$ ▷ Disable all wrong experts

For example, with $T = \infty$ and $m = 5$, we have: ("x" $\rightarrow$ expert got deactivated)

Time	1	2	3	...	∞
Expert #1	+	+	x	...	
Expert #2	+	-	-	...	+
Expert #3	+	+	x	...	
Expert #4	-	x		...	
Expert #5	-	x		...	
Prediction	+	+	-	...	+
Real	+	-	-	...	+
Error	0	1	0	...	0

Theorem 2.1. Halving algorithm makes $ALGO \leq \log m$ errors.

Proof:

(1) If ALGO predicted wrong, the experts' majority recommended wrong.

(2) By construction, we disable at least $\frac{m}{2}$ experts when this happen.

(3) After $\approx \log m$ errors, only the expert with perfect recommendation remains.

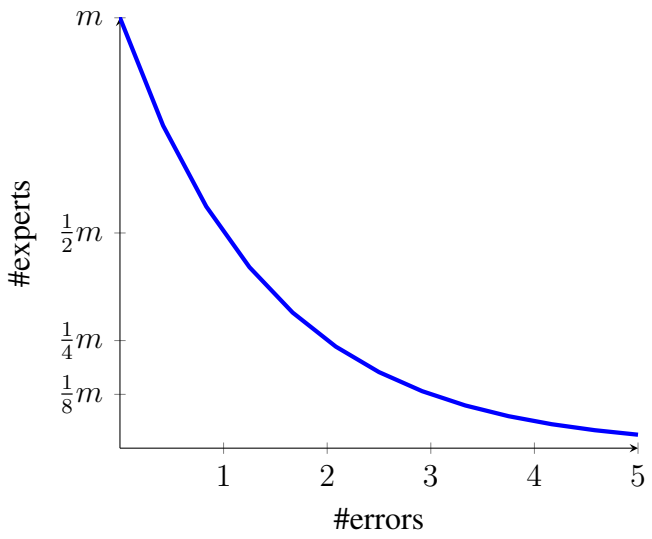


Figure 1: Upper bound on the numbers of experts after each error.

2.2 General Halving algorithm

The main issue with the Halving algorithm is that we assume that we have access to an expert with perfect recommendations, an assumption which does not hold for real-world applications.

Q. How do we remove the assumption of the existence of the expert with the perfect recommendation? If we leverage the Halving algorithm, the straightforward idea is to do the same procedure as Halving, but we enable all of them again when we have disabled all the experts. Algorithm 3 presents the modifications to the original algorithm.

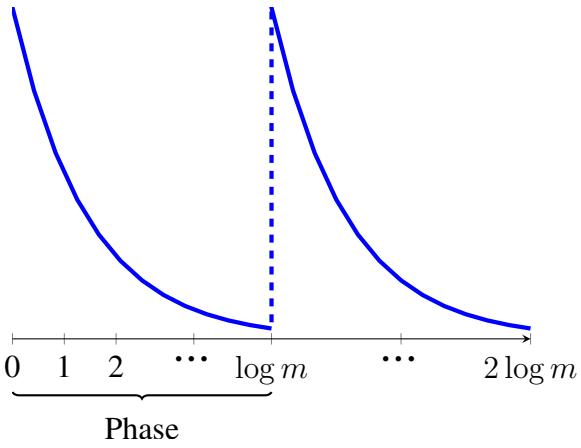
Algorithm 3 General Halving Algorithm

procedure GENERAL HALVING ALGO
  $A \leftarrow \{1, 2, \ldots, m\}$ ▷ Enable all experts
  for *t = 1, 2,..., T* **do**
    for *i = 1, 2,..., m* **do**
      $e_i^t \leftarrow ExpertRecommendation(i, t)$
    $o^t \leftarrow$ If $\sum_{i \in A} e_i^t > 0$ Then 1 Else -1 ▷ Follow majority recommendation
    $r^t \leftarrow InstanceRealization(t)$
    $error \leftarrow [r^t \neq o^t]$
    if $r^t \neq o^t$ **then**
      $A \leftarrow A - \{i | e_i^t = o^t\}$ ▷ Disable all wrong experts
      if $|A| = 0$ **then**
        $A \leftarrow \{1, 2, \ldots, m\}$ ▷ Enable all experts again

Theorem 2.2. General Halving algorithm makes $ALGO \leq (OPT + 1) \log m$ errors.

Proof:

(1) **Definition:** Let's consider **Phase** as the interval to each expert recommends wrong at least once;



(2) ALGO has $\leq \log m$ errors per Phase
(The proof follows the steps shown for Theorem 2.1, although no expert remains at the end);

(3) Each expert individually makes ≥ 1 errors at each Phase, even OPT;

(4) At the worst-case, OPT makes 1 error while ALGO makes $\log m$ errors per Phase. Thus, ALGO makes $OPT \log(m) + \log(m)$ errors in total, due to $OPT \equiv \#Phases$ and the last Phase can be left unfinished.

2.3 Weighted Majority algorithm

Another issue raised from Halving algorithm is that it has a radical behavior if an expert makes an error, removing it without a second thought. A better judgment would be just reducing the expert importance instead since we can be removing good experts unintentionally, which can undermine the prediction for the remainder predictions in turn.

Q. How can we place an importance over each expert instead of just removing it? Simply assign a weight w_i^t for each expert. Consequently this addition will require adaptation on the remainder algorithm.

The first modification consists of the algorithm prediction follow the weighted majority expert recommendation instead of just the majority:

$$o^t = \begin{cases} +, & \text{if } \sum_{\{i|e_i^t=+\}} w_i^t \geq \sum_{\{i|e_i^t=-\}} w_i^t \\ -, & \text{otherwise} \end{cases}$$

For example,

Expert	Weight	Recommendation
Expert #1	2	+
Expert #2	3	+
Expert #3	1	-
Expert #4	1	-
Expert #5	2	-
Prediction		+

The next modification is related to updating the experts' weight at each time t, how should it be done? Naturally, if an expert predicts correctly, its weight does not need to be updated. However, if an expert predicts wrongly, its weight should be reduced as symbolizing the decrease of its importance.

The question is, by how much? Initially, we could reduce it by half, but this would be a problem for longer predictions. So, a more generic approach is to update it by an ϵ factor which can vary depending on the application. Then,

$$w_i^{t+1} = \begin{cases} (1-\epsilon)w_i^t, & \text{if } e_i^t \neq r^t \\ w_i^t, & \text{otherwise} \end{cases}$$

We also have to set the initial value of all weights. As we do not have any a priori knowledge about the experts, we simply set all w_i^t equally as 1.

Finally, Algorithm 4 describes the new procedure.

Algorithm 4 Weighted Majority Algorithm

procedure WEIGHTED MAJORITY ALGO
  $w_i^1 \leftarrow 1, \quad \forall i = 1, 2, \ldots, m$ ▷ Weight initialization
  for *t = 1, 2,..., T* **do**
    for *i = 1, 2,..., m* **do**
      $e_i^t \leftarrow ExpertRecommendation(i, t)$
    $o^t \leftarrow$ If $\sum_{\{i|e_i^t=+\}} w_i^t \geq \sum_{\{i|e_i^t=-\}} w_i^t$ Then $+1$ Else -1 ▷ Weighted Majority
    $r^t \leftarrow InstanceRealization(t)$
    $error \leftarrow [e_i^t \neq r^t]$
    for *i = 1, 2,..., m* **do**
      $w_i^{t+1} \leftarrow$ If $e_i^t \neq r^t$ Then $(1-\epsilon)w_i^t$ Else w_i^t ▷ Weight update

Theorem. 2.3 Weighted Majority algorithm makes $ALGO \leq 2(1+\epsilon)OPT + \frac{2\log m}{\epsilon}$ errors, where $\epsilon \in [0, \frac{1}{2}]$.

Proof: Consider $W^t = \sum_i w_i^t$ as the total weight of the experts at time t.

(1) **How does it behaves over the errors?** When ALGO is wrong, this means that the weighted majority is wrong. Thus, at least $\frac{W^t}{2}$ must be updated.

$$\begin{aligned} W^{t+1} &\leq \frac{W^t}{2} + (1-\epsilon)\frac{W^t}{2} \\ &= W^t\Big(1 - \frac{\epsilon}{2}\Big) \\ &\leq W^1\Big(1 - \frac{\epsilon}{2}\Big)^{ALGO} \end{aligned}$$

(2) **How does it relates to OPT?** By our definition,

$$\begin{aligned} W^t &= \sum_i w_i^t \\ &\geq w_{OPT}^t \\ &= w_{OPT}^1 (1-\epsilon)^{OPT} \end{aligned}$$

(3) If we connect (2) and (3), we have

$$W^1\Big(1 - \frac{\epsilon}{2}\Big)^{ALGO} \geq W^t \geq w_{OPT}^1 (1-\epsilon)^{OPT}$$

Establishing a direct relation between $ALGO$ and OPT.

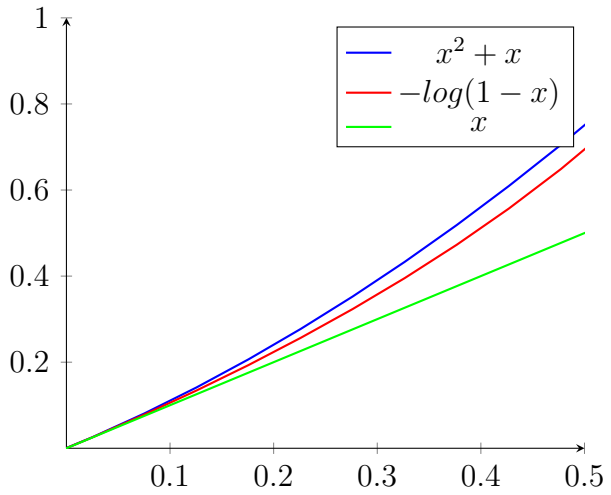
So continuing,

$$\begin{aligned}
&\Rightarrow m\left(1-\frac{\epsilon}{2}\right)^{ALGO} && \geq (1-\epsilon)^{OPT} \\
&\overset{\log}{\equiv} \log m + ALGO \log\left(1-\frac{\epsilon}{2}\right) && \geq OPT \log(1-\epsilon) \\
&\overset{-1}{\equiv} \log m + OPT \log\left(\frac{1}{1-\epsilon}\right) && \geq ALGO \log\left(\frac{1}{1-\frac{\epsilon}{2}}\right)
\end{aligned}$$

Using the following property:

$$x + x^2 \geq \log\left(\frac{1}{1-x}\right) \geq x, \quad x \in \left[0, \frac{1}{2}\right]$$

And visualization for intuition:



We have

$$\begin{aligned}
&\Rightarrow \log m + \epsilon(1+\epsilon)OPT && \geq \frac{\epsilon}{2} ALGO \\
&\Rightarrow \frac{2\log m}{\epsilon} + 2(1+\epsilon)OPT && \geq ALGO
\end{aligned}$$

Getting our desired bound.

An interesting fact about this Theorem's bound - and all others - is that we can see two error types: One related to errors made due to the quality of the experts at hand; Another related to the errors made while searching the better experts configuration.

$\frac{2\log m}{\epsilon} \Rightarrow$ Errors made until we find a "good" experts weights configuration

$2(1+\epsilon)OPT \Rightarrow$ Errors made due to the quality of the experts at hand

Thus over time the number of errors made by ALGO should be proportional to $2(1+\epsilon)OPT$.

2.4 Lower bound for Deterministic algorithms

As we saw now, our best algorithm is proportional to $2(1+\epsilon)OPT$, so a next natural follow up is knowing if this is the best we can do using deterministic algorithms or if it is possible to improve even more this bound. It turns out, we can't.

Lemma 2.4 $\forall$ deterministic algorithm $ALGO$, $\exists$ instance I, such that $ALGO \geq 2OPT$.

Proof:

(1) Fixed a deterministic algorithm $ALGO$ using experts
(Not necessarily one of those presented in this lecture)

(2) Between the experts used: One always predicts '+' and other always predicts '-'

(3) In the worst-case scenario, instance I always output the opposite of $ALGO$

Time	1	2	3	...	∞
Expert '+'	+	+	+	...	+
Expert '-'	-	-	-	...	-
$ALGO$	+	-	-	...	+
Real	-	+	+	...	-

(4) $ALGO$ has T errors by construction

(5) Because Expert '+' and Expert '-' are complement to each other, the sum of their errors must be T. As such, independently of the instance I, the best one of them will have $\leq \frac{T}{2}$ errors

(6) Connecting (4) and (5), we have $ALGO \geq 2OPT$

3 Randomized / Fractional algorithms

Because we proved the limitation of the deterministic approach, we are now going to shift our attention to the randomized approach, as it usually further improves the algorithmic bounds as already seen in previous lectures. This, as you can imagine, will also be true for the Prediction with Experts problem.

More interestingly, the algorithm that we are going to develop can be used to model many Online Learning problems, in special the Prediction with Experts. As such, we have to make a digression first to accommodate all these different problems in the same setup, called the Abstract game.

3.1 Abstract Game

We can think of the abstract game as the game that many Online Learning algorithms are playing behind closed doors, and so, also a generalization of the Prediction with Experts game.

First, similarly to the deterministic algorithms setup above:

- There are m actions or objects (e.g: experts)
- There will be $t = 1, 2, \ldots, T$ rounds (or time for conformity)

At each time t, the game plays out like the following :

(1) The ALGO generates a probability distribution $p^t \in \Delta^m$, where p_i^t is the probability of selecting object i at time t

$$p^t = (p_1^t, p_2^t, \ldots, p_m^t) \qquad p_i^t \in [0,1] \qquad \sum_{i=1}^{m} p_i^t = 1$$

(2) Next, the instance I reveals the loss $l_i^t \in [0,1]$ of each object i at time t.

(3) Finally, ALGO computes the expected loss cost of the generated distribution, which is given by:

$$\begin{aligned} \langle p^t, l^t \rangle &= \sum_{i=1}^{m} p_i^t l_i^t \\ &\equiv E[\text{selected object cost}] \end{aligned}$$

The game objective is to minimize the total expected loss cost. And after that compare it against the best object for measuring its competitiveness in the worst-case scenario.

$$\min \sum_{t=1}^{T} \langle p^t, l^t \rangle \qquad vs \qquad \min_i \sum_{t=1}^{T} l_i^t \equiv OPT$$

For example, with $T = 2$ and $m = 4$, we have:

Time	1		2	
	p^1	l^1	p^2	l^2
Object #1	1/4	1	1/8	1
Object #2	1/4	0	3/8	0.2
Object #3	1/4	0.1	1/4	0.5
Object #4	1/4	1	1/4	1

$$\begin{aligned} ALGO\ Cost &= (1/4 \times 1 + 1/4 \times 0 + 1/4 \times 0.1 + 1/4 \times 1) \\ &+ (1/8 \times 1 + 3/8 \times 0.2 + 1/4 \times 0.5 + 1/4 \times 1) \\ &\approx 1.1 \end{aligned}$$

$$OPT = 0.2$$

Ideally, a "good"$ALGO$ must detect the best object and, over time, put all distribution mass into this object.

3.1.1 Prediction with Experts via Abstract Game

So, how does the Prediction with Experts fits into this game?

- $Objects \equiv Experts$
- Between (1) and (2), $ALGO$ selects one expert at random following the distribution p^t (selects expert i with probability p_i^t and follows its recommendation)
- The loss of each expert i at time t follows:

$$l_i^t = \begin{cases} 1, & \text{if expert } i \text{ recommended wrong at time } t \\ 0, & \text{otherwise} \end{cases}$$

- The expect loss cost at time t:

$$\begin{aligned} \langle p^t, l^t \rangle &= \sum_{i=1}^{m} p_i^t l_i^t \\ &= \sum_{(i|l_i^t=1)} p_i^t \\ &= \text{Pr}[ALGO \text{ makes an error}] \end{aligned}$$

- The total expect loss cost:

$$\sum_{t=1}^{T} \langle p^t, l^t \rangle = E[\text{Total errors made by } ALGO]$$

- And finally the best expert:

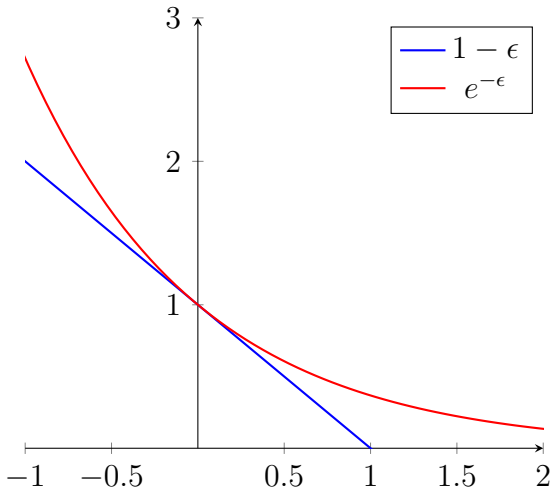
$$OPT = \min_i \sum_{t=1}^{T} l_i^t = \text{Number of errors made by expert } i$$

3.2 Multiplicative Weight Update (MWU) method

And so, how do we develop an algorithm that can model so many problems? It turns out we just need to adapt the Weighted Majority for the Abstract Game. To do this, we will need to apply the following major modifications:

(1) Normalize the weights of the Weighted Majority algorithm
Justification: The weights cannot be directly used, since they do not respect the properties of probabilistic distributions

(2) Generalization of the weights update to $w_i^{t+1} \leftarrow w_i^t e^{-\epsilon l_t^i}$
Justification: Ideally, the natural extension for $(1-\epsilon)$ would be $(1-\epsilon l_i^t)$, but the exponential function is a smoother updating factor. In any case, we can also see $e^{-\epsilon l_t^i}$ as a "generalization" of $(1-\epsilon l_i^t)$, because the former represents the first two terms of the latter approximation via Taylor series.



Algorithm 5 describes the Multiplicative Weights Update algorithm with these and other smaller modifications for following the abstract game described before.

Algorithm 5 Multiplicative Weights Update Algorithm

procedure MULTIPLICATIVE WEIGHTS UPDATE ALGO
  $w_i^1 \leftarrow 1, \quad \forall i = 1, 2, \ldots, m$ ▷ Weight initialization
  $p_i^1 \leftarrow 1/m, \quad \forall i = 1, 2, \ldots, m$ ▷ Probability initialization
  for *t = 1, 2,..., T* **do**
    $UseObjectDistribution(p^t)$ ▷ $\equiv$ Follows object i with probability p_i^t
    $l^t \leftarrow InstanceRealization(t)$ ▷ Instance I reveals loss of each object i
    $loss \leftarrow \langle p^t, l^t \rangle$
    for *i = 1, 2,..., m* **do**
      $w_i^{t+1} \leftarrow w_i^t e^{-\epsilon l_i^t}$ ▷ Weight update
    for *i = 1, 2,..., m* **do**
      $p_i^{t+1} = w_i^{t+1} / \sum_j w_j^{t+1}$ ▷ Probability update

Theorem. 3.2 Multiplicative Weights Update algorithm as the following guarantee:

$$\sum_{t=1}^{T} \langle p^t, l^t \rangle \lessapprox (1+\epsilon) OPT + \frac{\log m}{\epsilon}, \epsilon \in \left(0, \frac{1}{2}\right]$$

Which unfortunately will not be proved.

Q. How should we choose ϵ? Although earlier we said that we could choose the ϵ arbitrarily, it turns out that there is an optimal value for ϵ. As the guarantee found is a convex function, we can derived it, find its global minimum and use it.

- **If we know OPT beforehand:** $\epsilon = \sqrt{\frac{\log m}{OPT}}$

$$\begin{aligned}
\sum_{t=1}^{T} \langle p^t, l^t \rangle &\lessapprox (1+\epsilon) OPT + \frac{\log m}{\epsilon} \\
&\Rightarrow OPT + \epsilon OPT + \frac{\log m}{\epsilon} \\
&\Rightarrow OPT + \sqrt{OPT \log m} + \sqrt{OPT \log m} \\
&\Rightarrow OPT + 2\sqrt{OPT \log m}
\end{aligned}$$

- **If we do not know OPT beforehand:** $\epsilon = \sqrt{\frac{\log m}{T}}$ (Consider $OPT \equiv T$)

$$\begin{aligned}\sum_{t=1}^{T}\langle p^t, l^t\rangle &\lessapprox (1+\epsilon)OPT + \frac{\log m}{\epsilon}\\ &\Rightarrow OPT + \epsilon T + \frac{\log m}{\epsilon}\\ &\Rightarrow OPT + \sqrt{T\log m} + \sqrt{T\log m}\\ &\Rightarrow OPT + 2\sqrt{T\log m}\end{aligned}$$

We leave the first term as OPT - although still proportional to T - so we can analyze the contribution of each error type to the error rate through time.

$$\begin{aligned}\frac{\#Errors_{ALGO}}{T} &\leq \frac{OPT}{T} + \frac{2\sqrt{T\log m}}{T}\\ &\leq \frac{OPT}{T} + 2\sqrt{\frac{\log m}{T}}\end{aligned}$$

Which shows that over time ALGO only make errors proportional to the quality of the experts at hand.

3.3 Max version for Abstract Game / MWU

The description above is for the minimum formulation, what about the maximum formulation? For it, we just need some minor adjustments:

(1) loss $l_i^t \Rightarrow$ reward r_i^t
Justification: For the maximum formulation, we are interested in give rewards for better experts

(2) $w_i^{t+1} \leftarrow w_i^t e^{-\epsilon l_t^i} \Rightarrow w_i^{t+1} \leftarrow w_i^t e^{\epsilon r_t^i}$
Justification: Instead of decrease the importance of worse experts, we now start to increase the importance of better experts through rewards ($\equiv -$loss)

Theorem. 3.3 Multiplicative Weights Update algorithm (Max version) as the following guarantee:

$$\sum_{t=1}^{T}\langle p^t, r^t\rangle \gtrapprox (1-\epsilon)OPT - \frac{\log m}{\epsilon}, \quad \epsilon \in \left(0, \frac{1}{2}\right], \quad OPT = \max_i \sum_{t=1}^{T} r_i^t$$

Which unfortunately will also not be proved.