

## Lecture 5: February 5, 2024

*Lecturer: Yishay Mansour**Scribe: Aryeh Gorun, Erel Barzilay*<sup>1</sup>

## 1 Adversarial Costs: Full Feedback

Up until now we discussed the MAB problem, where we had stochastic models and partial feedback. This lecture, we will discuss adversarial models with full feedback.

**Full feedback:** After each action, we can see the feedback for *all* actions.

**Adversarial cost:**

- We move from attempting to maximize gains to attempting to minimize losses. This is not very significant but it is the convention (for historical reasons).
- There are no distributions! The adversary can select any sequence of costs.

**What to expect?**

- The loss may be high for all actions, thus the loss of any algorithm may be high.
- This is why regret is a useful metric - the online algorithm performance is compared to how well we could have done, had we chosen a single action.

**Online Model**

For each time  $t \in [T]$ :

1. The adversary selects costs  $c_t(a) \in [0, 1]$  for  $a \in A$
2. The algorithm selects  $a_t \in A$ , not knowing  $c_t(\cdot)$
3. The algorithm incurs cost  $c_t(a_t)$
4. The costs of all actions are revealed:  $c_t(a) \forall a \in A$

### 1.1 Sequential Prediction with Expert Advice

We have  $k$  experts (or algorithms).

At each round  $t$  every expert gives a prediction. The learner chooses a prediction based on the expert predictions. The results for time  $t$  are revealed (the loss, which is a function on the predictions).

**The goal:** Achieve a similar performance to the best expert.

---

<sup>1</sup>based on scribe notes of Adi Asher and Bar Moalem from 2021/22.

### What is the connection to machine learning?

We can think of the experts as a class of hypotheses,  $H$  (for example, a class of hyperplanes, decision trees or neural networks). The best expert,  $h^* \in H$  is the one that gives a prediction with the smallest loss.

A good example is the *Perceptron Algorithm*, which is an online algorithm that learns a linear separator. If the positive and negative examples can be separated by a linear hyperplane, the algorithm is guaranteed to converge with a finite number of errors. In our case, however, there may not be such a separator. The best predictor may still have a high loss. Therefore, we want our online algorithm to approach the loss of the best predictor  $h^*$ .

Another difference is that in the Adversarial setting we don't have distributions over observations, so how can we learn?

### Problem protocol (model)

In time  $t \in [T]$ :

1. An observation  $x_t$  arrives.
2. Each expert  $i$  predicts  $z_{i,t}$ , so we have the predictions  $z_{1,t}, \dots, z_{k,t}$ .
3. The algorithm picks an expert  $e \in [k]$  and predicts  $z_{e,t}$ .
4. The correct label  $z_t^*$  is revealed and costs  $C(z_{j,t}, z_t^*) \forall j \in [k]$ .
5. The algorithm incurs cost  $C_t = C(z_{e,t}, z_t^*)$

This model is also called *Online learning with experts*.

### Notes:

- This is a special case of the adversarial cost and full feedback online model.
- The prediction  $z_{j,t}$  can be thought of as a hypothesis  $h_j$ , such that  $h_j(x_t) = z_{j,t}$ .
- The algorithm may select a prediction in any way that it chooses, but it needs to be a prediction that one of the experts gave. For example, it can choose using a probability distribution on the experts.
- The cost function can be any arbitrary function. For example, if the labels are binary  $z \in \{0, 1\}$  the cost could be the zero/one loss (zero for identical label, one for opposite), and if the labels are  $z \in [0, 1]$ , the cost could be the square difference  $C_t = (z_{e,t} - z_t^*)^2$ .

## 1.2 IID cost

As we mentioned earlier, the adversary can choose the costs in any way that they want. First, we will analyze the case in which the adversary chooses a stochastic model with a distribution for each action  $D_a$  over  $[0, 1]$ , like in the previous lectures (but this time with full feedback).

**Greedy Algorithm:** The greedy algorithm chooses at time  $t \in [T]$  the action with the lowest average cost at the time.

We can expect the greedy algorithm to work well here because there is no motivation to 'explore', since we get full feedback anyways. So we start straight in 'exploit' mode.

**Theorem 1** *The greedy algorithm achieves  $\mathbb{E}(\text{regret}) = O(\frac{1}{\Delta})$ , where  $\Delta$  is the difference between the expected cost of the worst action and the expected cost of the best action.*

**Note:** *This is not dependant on  $\log T$ !*

**Proof:** Let's call the actions  $a_1, a_2$  and their expected costs  $c_1, c_2$ . Assume w.l.o.g  $c_2 > c_1$ . Denote  $\Delta = c_2 - c_1$ . This implies  $\Delta > 0$ . Let us define:

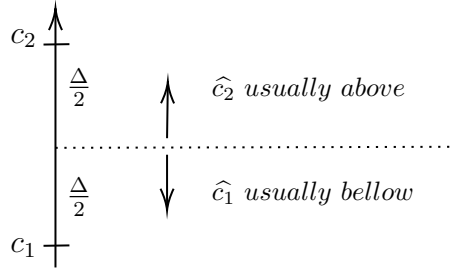


Figure 1: Graphic demonstration of cases  $n_1$  and  $n_2$ .

$$n_1 := \sum_{t=1}^T \mathbb{1} \left\{ \hat{c}_{1,t} > c_1 + \frac{\Delta}{2} \right\} \text{ (the number of times } \hat{c}_{1,t} > c_1 + \frac{\Delta}{2} \text{).}$$

$$n_2 := \sum_{t=1}^T \mathbb{1} \left\{ \hat{c}_{2,t} < c_1 + \frac{\Delta}{2} \right\} \text{ (the number of times } \hat{c}_{2,t} < c_1 + \frac{\Delta}{2} \text{).}$$

If the Greedy algorithm made a mistake, that means that the empirical cost of  $a_1$  is greater than the empirical cost of  $a_2$ . This implies that either  $n_1$  or  $n_2$  occurred (or possibly both). For the values of  $\hat{c}_{1,t}$  and  $\hat{c}_{2,t}$  to 'flip', at least one of them must be more than halfway of the distance between the expect values. See figure ?? for a visual explanation.

Now:

$$\mathbb{E}[\text{regret}] = \sum_{t=1}^T \mathbb{1} \{a_t = a_2\} \cdot \Delta \leq \mathbb{E}[n_1 + n_2] \cdot \Delta$$

$$\mathbb{E}[n_1] \leq \sum_{t=1}^T Pr[\hat{c}_{1,t} > c_1 + \frac{\Delta}{2}] \stackrel{Hoeffding}{\leq} \sum_{t=1}^{\infty} e^{-\frac{1}{2}(\frac{\Delta}{2})^2 t} \approx \int_1^{\infty} e^{-\frac{1}{2}(\frac{\Delta}{2})^2 t} dt = -\frac{e^{-\frac{1}{2}(\frac{\Delta}{2})^2 t}}{\frac{1}{2}(\frac{\Delta}{2})^2} \Big|_1^{\infty} = O\left(\frac{1}{\Delta^2}\right)$$

$$\mathbb{E}[n_2] \leq \sum_{t=1}^T Pr[\hat{c}_{2,t} < c_1 + \frac{\Delta}{2}] \stackrel{c_1 + \frac{\Delta}{2} = c_2 - \frac{\Delta}{2}}{\leq} \sum_{t=1}^{\infty} e^{-\frac{1}{2}(\frac{\Delta}{2})^2 t} = O\left(\frac{1}{\Delta^2}\right)$$

So in total the expected regret is:

$$\mathbb{E}[\text{regret}] \leq O\left(\frac{1}{\Delta}\right)$$

□

**Some intuition:**

With bandit feedback, we always got a factor of  $O(\log T)$ , but not with full information. What is the difference?

In the past, we always had to “fight” rare but possible bad events in which our estimates were wrong.

In order to limit the probability of these events, we had to “waste” some time on exploration.

For example, if we had  $a_1 \sim Br(\frac{1}{4}), a_2 \sim Br(\frac{3}{4})$ , we expect the averages to be  $1/4$  and  $3/4$  accordingly.

Now let’s look at the bad event, where for the first  $0.1 \log T$  timesteps, all  $a_1$  results were 1, and all  $a_2$  results were 1. The probability of this event is  $\frac{1}{4}^k \frac{1}{4}^k = 2^{-4k} = \frac{1}{T^{0.4}}$ , and if we stopped exploring after this, we would get a regret of  $T^{0.6}$ . This is why we had to explore for about  $\log T$  timesteps, to make sure the probability of a bad event is low.

With full feedback, we observe on every timestep, no matter what we choose. This means that we don’t have to spend time actively exploring in order to limit the probability of a bad event.

Is this bound tight?

Yes, it is. As we saw in the second lecture, when we have two Bernoulli coins, for the first  $\frac{1}{\Delta^2}$  coin tosses we don’t know (with good probability) which one is better.

## 2 Prediction with Expert Advice

### 2.1 Adversaries and Regret

We define two types of adversaries:

**Oblivious Adversary:** This adversary decides the costs of all observations in advance. It knows the algorithm, but not the results of any randomization that the algorithm uses.

**Adaptive Adversary:** Decides the costs for every timestep  $t$ , given the actions of the algorithm before that time.

Why do we use adversarial models? Why assume the worst?

In the IID model, we made a *lot* of assumptions (for example, that the distributions don't change, and that they are independent of each other). We can start to unravel some of the assumptions by looking at changing distributions or dependant variables. The adversarial model already includes all of them by assuming any possible sequence of cost. No matter how much we "weaken" our assumptions, the adversarial model still includes them. We will show that even in this model we can achieve good regret bounds.

The type of adversary depends on the use case. We can use an oblivious adversary for cases in which our decisions don't affect future costs, like a weather forecast. The weather tomorrow is not dependant on the prediction we make today. We can use an adaptive adversary for more interactive use cases, in which our actions affect the future costs. We don't know how our actions affect the distributions, so we assume the worst case.

**Definitions:**

$$\begin{aligned} \text{cost}(a) &= \sum_{t=1}^T c_t(a) \\ \text{cost}(ALG) &= \sum_{t=1}^T c_t(a_t) \\ a^* &= \operatorname{argmin}_{a \in A} \text{cost}(a) \quad \text{Cost}^* = \text{cost}(a^*) \end{aligned}$$

Why is  $\text{Cost}^*$  our benchmark? In the stochastic case it makes more sense, because the best action is the one we would always choose if we had full information. But here we could choose, for example, to compare to the best *sequence* of actions (we can choose a different action every time). We are choosing to still compare to the best action here, since it is a very simple and intuitive benchmark. In the context of machine learning, we are comparing to the best hypothesis in our class of hypotheses.

We define two types of oblivious adversaries:

**Deterministic oblivious adversary:** This adversary builds a table of costs in advance:

$$\{c_t(a) : a \in A, t \in [T]\}$$

And so:

$$\text{Regret} = \text{cost}(ALG) - \min_{a \in A} (\text{cost}(a)) = \text{cost}(ALG) - \text{Cost}^*$$

**Randomized oblivious adversary:** This adversary builds multiple tables of costs in advance, and gives a distribution of them for every  $t$ . So:

$$\text{PseudoRegret} = \text{cost}(ALG) - \min_{a \in A} (\mathbb{E}[\text{cost}(a)])$$

### 2.2 Binary Prediction with Expert Advice

This algorithm is also called the *halving algorithm*. We use a binary system, with binary labels and 0/1 loss. We assume that there exists a perfect expert, for whom all costs are zero, i.e an expert  $e$  such that:

$$\forall t : c_t(e) = 0$$

For example, for the Perception Algorithm this would be a hyperplane that separates the data perfectly.

### The Halving Algorithm:

1. Start with  $S \leftarrow AllExperts$ .  $S$  will include at time  $t$  all of the experts who were never wrong until now.
2. At time  $t$ :
  - (a)  $S_1 = \{e \in S : z_{e,t} = 1\}$
  - (b)  $S_0 = \{e \in S : z_{e,t} = 0\}$
  - (c) If  $|S_1| > |S_0|$ , select label 1, otherwise select 0
  - (d) The result  $b_t$  is revealed
  - (e) Set  $S \leftarrow S_{b_t}$  - this removes from  $S$  the experts which were wrong in this round.

**Lemma 2** *The number of errors is at most  $\log k$  (for  $k$  the number of experts).*

**Proof:**

Define:

- $S_t = S$  at time  $t$
- $W_t = |S_t|$  the size of the set of remaining experts

Initially  $W_1 = k$ , because  $S$  includes all experts at the beginning.

In addition, we know that one expert is never wrong, so for all  $t \in [T]$ :  $W_t \geq 1$

If the algorithm makes a mistake at time  $t$ , more than half of the experts recommended the wrong label.

They are removed from  $S$ , so  $W_{t+1} \leq W_t/2$ .

Since every mistake reduces the group size by at least half, and it cannot be smaller than 1:

$$\text{num of errors} \leq \log_2 k$$

□

### Proof methodology

We will use a similar proof concept multiple times during this lecture. Each time, we will define  $W_t$  so that:

1.  $W_t$  measures the “weight” of experts left
2.  $W_1$  has an upper bound
3.  $W_t$  does not increase
4.  $W_T$  has a lower bound because of the best expert (the benchmark)

Then, we will find a connection between the cost of the algorithm and the decrease in  $W_t$  to achieve the bound.

## 2.3 Weighted Majority Algorithm

For the halving algorithm, we assumed that there is a perfect expert. This is of course a very generous assumption. We would like to use an algorithm that does not make this assumption.

The algorithm will use:

- $w_t(i)$  - the weight of expert  $i$ . Can be thought of as the credibility of that expert.
- $\eta$  - the weight updating parameter. Can be thought of as a “learning rate”.

After an expert  $i$  makes a mistake, the algorithm adjusts:  $w_{t+1}(i) = (1 - \eta) w_t(i)$   
 Our prediction will be according to a weighted average of  $w_t(i)$   
 Formally:

- $\eta \in [0, 1]$
- Initialize  $w_1(i) = 1$  for  $i \in [k]$
- At time  $t$ :
  1.  $S_{0,t} = \{i : z_{i,t} = 0\}, S_{1,t} = \{i : z_{i,t} = 1\}$
  2. If  $\sum_{i \in S_{1,t}} w_t(i) \geq \sum_{i \in S_{0,t}} w_t(i)$ , predict 1. Else, predict 0.
  3. Observe the true label  $z^*$
  4. For each expert  $i$ :
    - If  $z^* \neq z_{i,t}$ , set  $w_{t+1}(i) = (1 - \eta)w_t(i)$
    - Else  $w_{t+1}(i) = w_t(i)$

**Note:** unlike the halving algorithm, WMA adjusts the weights even if it did not make a mistake.

**Theorem 3** *The number of errors WMA makes is smaller than  $\frac{2}{1-\eta}Cost^* + \frac{2}{\eta} \ln k$*

**Note:** There is a tradeoff in the selection of  $\eta$ . If  $\eta$  is small, the penalty that we take from the number of experts is larger, but the multiplier of  $Cost^*$  is smaller (close to 2). If  $\eta$  is large, the penalty from  $\ln k$  is lower, but the multiplier of  $Cost^*$  can be very large.

**Proof:** We will use the same methodology as before.

Define:  $W_t = \sum_i w_t(i)$

So:  $W_1 = \sum_{i=1}^k 1 = k$

Each time  $a^*$  makes a mistake, its weight is reduced. Since  $Cost^*$  is the number of mistakes expert  $a^*$  makes after  $T$  rounds:

$$W_{T+1} > w_T(a^*) = (1 - \eta)^{Cost^*}$$

$$\frac{W_{T+1}}{W_1} > \frac{(1 - \eta)^{Cost^*}}{k}$$

We define  $S_{e,t}$  the set of experts that made a mistake in timestep  $t$ .

The individual weights never increase, so  $W_{t+1} \leq W_t$

At each timestep:

$$(1)W_{t+1} = \sum_i w_{t+1}(i) = \sum_{i \in S_{e,t}} (1 - \eta)w_t(i) + \sum_{i \notin S_{e,t}} w_t(i) = W_t - \eta \sum_{i \in S_{e,t}} w_t(i)$$

When the algorithm makes a mistake, the sum of the weights of the wrong label is larger. So in that case:  $\sum_{i \in S_{e,t}} w_t(i) \geq \frac{1}{2}W_t$ . Now from equation (1):

$$W_{t+1} \leq (1 - \frac{\eta}{2}) W_t$$

If we assume that the algorithm made  $M$  mistakes:

$$\frac{(1 - \eta)^{Cost^*}}{k} < \frac{W_{T+1}}{W_1} = \prod_{t=1}^T \frac{W_{t+1}}{W_t} \leq (1 - \frac{\eta}{2})^M$$

We take the logarithm of both sides:

$$Cost^* \ln(1 - \eta) - \ln k < \ln(1 - \frac{\eta}{2}) \cdot M \stackrel{\ln(1-x) \leq -x \text{ for } x > 0}{\leq} -\frac{\eta}{2} M$$

And we get the bound:

$$M < Cost^* \frac{2}{\eta} \ln\left(\frac{1}{1-\eta}\right) + \frac{2}{\eta} \ln k$$

Since  $\ln(x+1) \leq x$  for all  $x$ ,  $\ln(\frac{1}{1-\eta}) = \ln(\frac{\eta}{1-\eta} + 1) \leq \frac{\eta}{1-\eta}$ , and so:

$$M < Cost^* \frac{2}{1-\eta} + \frac{2}{\eta} \ln k$$

□

**Note:** if  $Cost^* = \Theta(T)$ , the regret is linear in  $T$ . The factor 2 is especially worrying. It means that from 20% error for the benchmark, our algorithm will reach 40% error, which is a significant difference. We want to reach a factor of 1, plus maybe something sub-linear. In the next section we will start by developing algorithms that are not as good, in order to advance towards the bound we want.

### 3 Online Learning with Adversarial Costs

There are  $k$  actions, action  $a$  has a cost  $c_t(a)$  at time  $t$ .

We define:

$$L_t^a = \sum_{\tau=1}^t c_\tau(a), \quad L_t^* = \min_a L_t^a$$

$L_t^a$  is the sum of the costs of an action until time  $t$ .

**Our goal:** find an algorithm that will achieve  $L_T^*$ , plus maybe some sub-linear additional cost.

#### 3.1 Deterministic Algorithms

First let's start with deterministic algorithms, and look at the greedy algorithm. After all, it worked well in the stochastic case.

**Greedy Algorithm:** The greedy algorithm  $G$  selects at time  $t$ :  $a_t = \arg \min_a L_{t-1}^a$

In the case of actions with equal sum of costs, it chooses the smaller index (this is important to mention because we want it to be completely deterministic).

Note that  $G$  can calculate  $L_{t-1}^*$  for all  $t$  because it has full feedback.

**Remark:** Here and for the rest of the proofs in this lecture we will assume  $c_t(a) \in \{0, 1\}$

**Theorem 4**

$$cost(G) \leq kL_T^* + k - 1$$

**Proof:** We look at a specific time  $t$  and define  $L_t^* = n$ .

We define  $B_n$  to be the set of actions  $a$  where  $L_t^a = L_t^* = n$ .

Every time that  $G$  is wrong, the size of  $B_n$  is reduced by at least 1 (because  $G$  selected one of the best actions so far, which is of course in  $B_n$ , and that action was not optimal). After at most  $k$  mistakes,  $L_t^*$  must increase by 1 (because all individual actions were wrong at least once).

If we define  $L_{B_n}^G$  to be the number of mistakes  $G$  made while  $L_t^* = n$ , we get:

$$cost(G) \leq \sum_{n=0}^{L_T^*} L_{B_n}^G \stackrel{(1)}{\leq} k \cdot L_T^* + k - 1$$

(1) is true because  $L_{B_n}^G \leq k$  (since  $B_n$  decreases with every mistake and is at most  $k$  at first). The last  $k - 1$  is the “remainder” for the last value of  $L_t^*$ , where the algorithm made at most  $k - 1$  mistakes without the value changing. □

**Notes:**

- It is easy to build an adversarial cost sequence that is very bad for the greedy algorithm: each time, we will make all but one of the actions succeed. The mistaken action will start at action 1, and go in order, wrapping back to 1 when it reaches  $k$ . The algorithm will choose at each round the only wrong action.
- This means that the bound from the theorem is tight.

We will now show that this is not only true for the greedy algorithm but for every deterministic algorithm.

**Theorem 5** *For every deterministic algorithm  $D$ :*

$$L_T^D \geq kL_T^* + T \bmod k$$

**Proof:** Since the algorithm is deterministic, we know which action  $a_t$  it will take at time  $t$ . So we set the costs at time  $t$  to be:

$$c_t(a_t) = 1, \quad c_t(a) = 0 \text{ for } a \neq a_t$$

There is an action  $a$  such that  $L_T^a \leq \lfloor \frac{T}{k} \rfloor$  (pigeonhole principle). That means that  $L_T^* \leq \lfloor \frac{T}{k} \rfloor$ . The algorithm is always wrong, so:

$$L_T^D = T = k \left\lfloor \frac{T}{k} \right\rfloor + T \bmod k \geq kL_T^* + T \bmod k$$

□

It seems like the fact that the deterministic algorithm is too predictable can be exploited by the adversary. So to improve our cost, we need to use randomized algorithms.

### 3.2 Randomized Greedy Algorithm

We define:

$$S_t = \{a \in A | L_t^a = L_t^*\}$$

So  $S_t$  is the set of all actions that performed the best up to time  $t$ .

Then at time  $t$  the *RG* algorithm will select an action  $a$  with probability  $P_t^a$ , such that:

$$P_t^a = \begin{cases} \frac{1}{|S_{t-1}|} & \text{if } a \in S_{t-1} \\ 0 & \text{else} \end{cases}$$

**Theorem 6**

$$L_T^{RG} \leq (\ln(k) + 1)L_T^* + \ln(k)$$

**Proof:** Let us look at  $S_t$  when  $L_t^* = n$  (This is the same as  $B_n$  from before). The adversary knows we will select an action from  $S_t$ . So for every timestep  $t$  they will also select an action  $a \in S_t$  and set:

$$c_t(a) = 1, \quad c_t(a') = 0 \text{ for } a' \neq a$$

Of course, the adversary will not want to set a loss for any action not in  $S_t$ , because they will not be chosen, and the goal is to keep  $L_T^*$  low. But why not set a loss for more than one action in  $S_t$ ?

If the adversary chooses to give  $r > 1$  losses at time  $t$ , the probability of a loss for the algorithm is  $\frac{r}{n}$ . On the other hand, if the actions in  $S_t$  are chosen one by one (with the size of the set decreasing by 1 every time), we get:

$$\frac{1}{n} + \frac{1}{n-1} + \dots + \frac{1}{n-r+1} > \frac{r}{n}$$

So by distributing  $r$  costs over time, the adversary increases the probability of the algorithm choosing an action with a cost, and thus the algorithm's loss increases. The expected loss of the algorithm:

$$\mathbb{E}[L_{B_n}^{RG}] \leq \sum_{i=1}^k \frac{1}{i} \leq \ln(k) + 1$$

$$\mathbb{E}[L^{RG}] = \mathbb{E}\left[\sum_{n=0}^{L_T^*}\right] \leq (\ln(k) + 1)L_T^* + \ln k$$

The last  $\ln k$  is again added because of the remainder. □

**Note:** This is a very significant improvement over the deterministic algorithm, but still not as good as we want. We are still linear in  $L_T^*$ , and with a factor dependent on  $k$ .

### 3.3 Randomized Weighted Majority

We now introduce a randomized version of the **Weighted Majority Voting** algorithm. This time, in each time step  $t$ , the algorithm chooses an action according to a distribution - which is proportional to the weights of each action. For simplicity, we assume that  $c_t(a) \in \{0, 1\}$ . Formally:

- $\eta \in [0, 1]$
- Initialize  $w_i(a) = 1 \forall i \in K$ , the initial weight of each action.
- For  $t \in [T]$ :
  1. Select action  $a_t = a$  with probability  $p_t(a)$ , where  $p_t(a) = \frac{w_t(a)}{\sum_{a'} w_t(a')}$
  2. Update:  $w_{t+1}(a) = \begin{cases} (1 - \eta) \cdot w_t(a) & c_t(a) = 1 \\ w_t(a) & c_t(a) = 0 \end{cases}$

This can be summarized to  $w_{t+1}(a) = (1 - \eta)^{c_t(a)} w_t(a)$ .

In words, if the cost is 0, the weight doesn't change, and if it is 1, it is multiplied by  $1 - \eta$ .

**Note:** The update will also work for  $c_t(a) \in [0, 1]$ . The proof is quite similar but was not shown in class.

We now define  $L_T^{RWM} = \sum_{t=1}^T \sum_a p_t(a) c_t(a)$ , which is the total sum of the expected cost in each round  $t$  (called fractional model). This will be the loss of the algorithm.

**Corollary 7** Let  $\eta \in (0, \frac{1}{2})$ , then the upper bound of the algorithm is:

$$L_T^{RWM} \leq (1 + \eta)L_T^* + \frac{\ln k}{\eta}$$

**Proof:** We define the probability mass of the actions which receive a cost of 1 at time  $t$  by:

$$F_t = \frac{\sum_{a:c_t(a)=1} w_t(a)}{W_t} = \sum_{a:c_t(a)=1} p_t(a)$$

Basically, this is the total loss at round  $t$ . Note that  $L_T^{RWM} = \sum_t F_t$ .

We will now analyze  $W_{t+1}$ :

$$\begin{aligned}
W_{t+1} &= \sum_a w_{t+1}(a) = \sum_{a:c_t(a)=0} w_{t+1}(a) + \sum_{a:c_t(a)=1} w_{t+1}(a) \\
&= \sum_{a:c_t(a)=0} w_t(a) + \sum_{a:c_t(a)=1} (1-\eta)w_{t+1}(a) \\
&= \sum_a w_t(a) - \eta \sum_{a:c_t(a)=1} w_t(a) \\
&= W_t - \eta \cdot F_t W_t \\
&= W_t(1 - \eta F_t) \\
&= W_1 \prod_{\tau=1}^t (1 - \eta F_\tau)
\end{aligned} \tag{1}$$

In the first line we split the sum to actions that received cost 1 and actions that received cost 0 in the last round. The second line follows the way we defined  $w_{t+1}$ . The third line is from rearranging the summands. The fourth equation is from the definition of  $F_t$ . Finally, we recursively open  $W_t$ . Since  $W_1 = K$  we get:

$$W_{T+1} = K \prod_{t=1}^T (1 - \eta F_t).$$

Additionally, since  $W_{T+1}$  is at least  $w_{T+1}(a^*)$  we get:

$$W_{T+1} \geq w_{T+1}(a^*) = (1 - \eta)^{L_T^*}.$$

Where  $L_T^*$  is the cost of the best action.

Now, comparing the lower and upper bounds we get:

$$(1 - \eta)^{L_T^*} \leq K \prod_{t=1}^T (1 - \eta F_t).$$

We apply logarithm on both sides:

$$L_T^* \ln(1 - \eta) \leq \ln K + \sum_{t=1}^T \ln(1 - \eta F_t).$$

We will use the fact that  $\forall z \in [0, \frac{1}{2}] -z - z^2 \leq \ln(1 - z) \leq -z$  and get:

$$L_T^*(-\eta - \eta^2) \leq \ln K + \sum_{t=1}^T (-\eta F_t)(\cdot)$$

Finally, we get:

$$\underbrace{\sum_{t=1}^T F_t}_{L_T^{RWM}} \leq (1 + \eta)L_T^* + \frac{\ln K}{\eta}$$

□

Which completes the proof.

**Corollary 8** Let  $\eta = \min\{\frac{1}{2}, \sqrt{\frac{\ln k}{T}}\}$ , then the upper bound of the algorithm is:

$$L_T^{RWM} \leq L_T^* + 2\sqrt{T \ln k}$$

**Proof:** The value of  $\eta$  is received by minimizing the upper bound of the algorithm loss, that we got in the last theorem.

In case  $\eta = \frac{1}{2}$  it follows that:

$$\frac{1}{2} \leq \sqrt{\ln(K) \cdot T}$$

$$\sqrt{T} \leq 2\sqrt{\ln K}$$

Multiplying by  $\sqrt{T}$ :

$$\text{regret} \leq T \leq 2\sqrt{T \ln K}$$

Now let's look at the case where  $\eta = \sqrt{\frac{\ln K}{T}}$ . From the last theorem we have:

$$L_T^{RWM} \leq L_T^* + \eta L_T^* + \frac{\ln K}{\eta}$$

Using the value of  $\eta$  and the fact that  $L_T^{RWM} \leq T$  and  $L_T^* \leq T$  we get:

$$L_T^{RWM} \leq L_T^* + \sqrt{\frac{\ln k}{T}} T + \sqrt{\frac{T}{\ln K}}$$

$$L_T^{RWM} \leq L_T^* + 2\sqrt{T \ln K}$$

□

**Note:** This is a loose bound. If we had a tighter upper bound on  $L_T^*$  we could get a better regret bound.

## 4 Time Selection Functions

So far, in the previous sections, we looked at all the time steps together and tried to find an upper bound that is relevant to the regret in each and every time step. But what happens if we want to address only a subset of the time steps?

Let's define functions  $I : [T] \rightarrow \{0, 1\}$ .

The regret of an algorithm  $H$  in respect to function  $I$  and action  $a$  is defined as:

$$R_T^H(I, a) = \sum_{t=1}^T I(t)(c_t(H) - c_t(a))$$

Basically, for each time step in  $I$ , we compare the cost of the algorithm to the cost of action  $a$ .  $a_t$  is the action that the algorithm chooses at time step  $t$ .

For a set of  $M$  functions  $I(\cdot)$  we require that:

$$\forall a \forall I R_T^H(I, a) = o(T)$$

In other words, we would like to get a sub-linear regret for all the  $M$  functions (which can be seen as subsets of the time steps).

### Algorithm H

- $\beta \in (0, 1)$
- Define the regret  $\tilde{R}_T^H(I, a) = \sum_{t=1}^T I(t)(\beta c_t(H) - c_t(a))$ . The parameter  $\beta$  decreases the online loss in each time step.
- Define for each time selection function  $I$  and for each action  $a$  a weight  $w_t(I, a)$ . Initialize  $w_0(I, a) = 1$ . The update rule will be:

$$w_{t+1}(I, a) = w_t(I, a) \beta^{-I(t)(\beta c_t(H) - c_t(a))} = w_t(I, a) \beta^{-\tilde{R}_t(I, a)}$$

- For each  $t \in [T]$  we will choose an action  $a_t$ :

1.  $w_t(a) = \sum_I I(t)w_t(I, a)$

This is the weight of action  $a$  in time  $t$  - We sum the weights  $w_t(I, a)$  for all the functions that give 1 for this time step.

2.  $W_t = \sum_a w_t(a)$

3.  $p_t(a) = \frac{w_t(a)}{W_t}$

4. Choose action  $a_t$  according to the distribution  $p_t(\cdot)$

**Lemma 9**  $\forall t \quad 0 \leq \sum_{a,I} w_t(I, a) \leq \sum_{a,i} w_{t-1}(I, a) \leq MK$   
*This means that the total weight does not increase in time.*

**Proof:** We prove by induction over  $t$  that the above holds.

For  $t = 0$  this is trivial.  $w_0(I, a) = 1$  for each  $a$  and  $I$ , hence the total weight is exactly  $MK$ .

At time  $t$ :

$$\begin{aligned}
W_t c_t(H) &= W_t \sum_a p_t(a) c_t(a) \\
&\stackrel{(1)}{=} \sum_a w_t(a) c_t(a) \\
&\stackrel{(2)}{=} \sum_a \sum_I I(t) w_t(I, a) c_t(a)
\end{aligned} \tag{2}$$

(1) is because  $p_t(a) = \frac{w_t(a)}{W_t}$ .

(2) is from the definition of  $w_t(a)$ .

Now let's analyze  $W_{t+1}$ :

$$\begin{aligned}
W_{t+1} &= \sum_a \sum_I w_{t+1}(I, a) \\
&\stackrel{(1)}{=} \sum_a \sum_I w_t(I, a) \beta^{-I(t)(\beta c_t(H) - c_t(a))} \\
&= \sum_a \sum_I w_t(I, a) \beta^{I(t)c_t(a)} \beta^{-\beta I(t)c_t(H)} \\
&\stackrel{(2)}{\leq} \sum_{a,I} w_t(I, a) (1 - (1 - \beta)I(t)c_t(a)) (1 + (1 - \beta)I(t)c_t(H)) \\
&\stackrel{(3)}{\leq} \underbrace{\sum_{a,I} w_t(I, a)}_{W_t} - \underbrace{(1 - \beta) \sum_{a,I} I(t) w_t(I, a) c_t(a)}_{W_t c_t(H)} + \underbrace{(1 - \beta) \sum_{a,I} I(t) w_t(I, a) c_t(H)}_{W_t c_t(H)} \\
&= W_t
\end{aligned} \tag{3}$$

(1) According to update rule.

(2) is due to the following inequalities:

$$\begin{aligned}
&\forall x, \beta \in [0, 1] \\
&\beta^x \leq 1 - (1 - \beta)x \\
&\beta^{-x} \leq 1 + \frac{(1 - \beta)x}{\beta}
\end{aligned}$$

(3) is from applying equation (2) from above. We showed that  $W_{t+1} \leq W_t$  for every  $t$ , thus the lemma holds true.  $\square$

Now we will use this lemma to find a regret bound:

**Corollary 10**

$$\forall a, I \quad w_t(I, a) = \beta^{L_T^{a,I}} - \beta^{L_T^{H,I}} \leq MK$$

Where:

$$L_T^{a,I} = \sum_t I(t)c_t(a) \quad L_T^{H,I} = \sum_t I(t)c_t(H)$$

Applying logarithm on both sides:

$$(-L_T^{a,I} + \beta L_T^{H,I}) \ln \frac{1}{\beta} \leq \ln MK$$

$$\beta L_T^{H,I} \leq L_T^{a,I} + \frac{\ln MK}{\ln \frac{1}{\beta}}$$

And we get:

**Theorem 11**

$$\forall a, I \quad L_T^{H,I} \leq \frac{1}{\beta} (L_T^{a,I} + \frac{\ln MK}{\ln \frac{1}{\beta}})$$

Setting  $\beta = 1 - \eta$ :

$$L_T^{H,I} \leq (1 + \frac{\eta}{1 - \eta}) L_T^{a,I} + \frac{\ln(kM)}{(1 - \eta) \ln(\frac{1}{1 - \eta})} \leq L_T^{a,I} + 2\eta L_T^{a,I} + \frac{2 \ln(kM)}{\eta}$$

After optimization over  $\beta = 1 - \eta$ , and using the identity  $\frac{x}{1+x} \leq \ln(1+x) \leq x$  for  $x \geq 0$  we get the final regret bound:

**Corollary 12**

$$\forall a, I : \quad L_T^{H,I} \leq L_T^{a,I} + O(\sqrt{T \log(Mk)})$$

This result can be improved if there is an upper bound on  $L_{min} = \max_I \min_a L_T^{a,I}$

## 5 Lecture Summary

In this lecture we discussed the following problems:

- Adversarial models
- The experts problem
- Online learning with full feedback
- Added time selection

For all of the problems we showed algorithms with sub-linear regret, even with no assumptions on the cost function.

## 6 Sources

1. Introduction to Multi-Armed Bandits by Aleksandrs Slivkins - Chapter 5
2. Learning, Regret, Minimization and Equilibria by A. Blum and Y. Mansour - Chapter 4.3
3. From External to Internal Regret by A. Blum and Y. Mansour - Chapter 7