CSCI699: Topics in Learning & Game Theory

Lecture 5

Lecturer: Shaddin Dughmi *Scribes: Umang Gupta & Anastasia Voloshinov*

In this lecture, we will give a brief introduction to online learning and then go through some online learning algorithms. Our discussion today will be in a non-game theoretic setting but we will show implications for games in the next lecture.

1 Online Learning

In online learning we have a single agent versus an adversarial world. We consider T time steps, where at each step $t = 1 \dots T$, the agent chooses 1 of n actions. For example, we might consider a scenario where the time steps are days and each day you choose one of n routes that you will take to work.

The cost of an action at a time t is determined by an adversary. We will denote the cost at time t of action a as $c_t(1) = [-1, 1]$. When $c_t(a)$ is negative, we can think of this as utility or reward and when it is positive it is dis-utility or penalty.

The adversary has access to the agent's algorithm, the history of the agents actions up to time $t-1$, and the distribution p_t on the actions. So, the adversary is quite strong since it can use all of this information to tailor its $c_t(a)$. The only leverage that the agent has is that the agent gets to choose what action they will take at time t.

1.1 Learning Setup (Perspective of Universe)

In this section, we describe the learning setup mathematically. This is the procedure that universe runs.

At each time step $t = 1 \dots T$ following occurs:

1. The agent picks a distribution p_t over $A = \{a_1 \dots a_n\}$.
2. The adversary picks the cost vector $c_t : A \to [-1, 1]$.
3. An action $a_t \sim p_t$ is chosen and agent incurs loss $c_t(a_t)$.
4. The agent learns c_t for use in later time steps.

In this procedure, an agent gets to pick its distribution over the actions. Then, the adversary chooses the cost seeing this distribution. After playing an action, the agent learns the cost function and then can reflect on the outcome to use the knowledge in future time steps.

1.2 General Online Learning Algorithm

1.2.1 Perspective of Agent

In this section, we present the structure of a general online learning algorithm.

Algorithm 1 General Online Algorithm for agent

Input: History up to time $t-1$. This includes the following information:
$c_1 \dots c_{t-1} : A \to [-1, 1]$
$p_1 \dots p_{t-1} \in \Delta(A)$
$a_1 \dots a_{t-1} \in A$
Output: Distribution over actions that you are going to take, $p_t \in \Delta(A)$

In reality, we only really need $c_1 \dots c_{t-1}$ to make a decision about our new distribution, it turns out that the other information is not really helpful.

Note that after each round, we learn the costs of all the actions including those that we did not choose. This is a full-information online learning setup.

1.2.2 Perspective of the Adversary

In this section, we look at the online learning algorithm from the perspective of the adversary. We assume that the adversary has no computational limitations.

Algorithm 2 General Online Algorithm for adversary

Input: Everything except the randomness used to draw $a_t \sim p_t$. More specifically, this includes the following:
History up to $t-1$
The distribution p_t, but not the draw from the distribution
The algorithm used by the agent
Output: $c_t : A \to [-1, 1]$

2 Benchmarks

The Objective of Online Learning: The objective is to minimize the expected cost per unit time incurred by the agent as compared to a suitable benchmark.

Naturally, this leads to the question of what benchmark is suitable. We shall explore one failed benchmark in this section and then the benchmark that we will end up using. First however, we will formalize our notion of cost to be able to define the objective.

2.1 Formalizing Cost

We will define the cost of the algorithm at time step t as

$$cost_{alg}(t) = c_t(a_t).$$

The total cost of the algorithm will be defined as the cumulative cost over all T rounds as

$$cost_{alg} = \sum_{t=1}^{T} c_t(a_t).$$

Given that we are randomizing, we care about the expected cost. So, we will define the expected cost at time t as the summation over all actions of the product of the probability that they choose action a, and the cost of choosing action a. We denote this by -

$$E[cost_{alg}(t)] = \sum_{a=1}^{n} p_t(a)c_t(a).$$

Note that by expressing the expectation in this manner, we are assuming that the cost and the draw from the distribution are independent.

For the expected total cost, we sum up the expected cost at time t over all values of t to get the following:

$$E[cost_{alg}] = \sum_{t=1}^{T}\sum_{a=1}^{n} p_t(a)c_t(a).$$

Our Goal: To make $E[cost_{alg}]$ small, no matter how clever the adversary is, as compared to a benchmark. Formally, we want

$$lim_{T\to\infty}\frac{1}{T}(E[cost_{alg}] - E[benchmark]) = 0$$

If this holds, we say that the algorithm has **no regret** or **vanishing regret** with respect to the benchmark.

Now that we have defined cost, we will first look at a unrealistic example of a benchmark and then the actual benchmark we will be using.

2.2 Best Action Sequence in Hindsight Benchmark (Unrealistic)

For our unrealistic benchmark example, we will define the benchmark as the cost of the best action sequence in hindsight. Thus, you will look at the expected cost of your algorithm compared to an omniscient algorithm that can always choose the best action tailored to the adversary.

Formally, this value will be $\sum_{t=1}^{T} \min_{a \in A} c_t(a_t)$. We can think of this value as how well you could do if you hacked your adversary and saw their cost assumptions. We can already see that this is not attainable because you do not have access to c_t before having to choose a_t.

Claim 1. *There is no online learning algorithm achieving vanishing regret with respect to the best action sequence in hindsight.*

Proof. The clever adversary can set $c_t(a) = 0$ for the action a that minimizes $p_t(a)$ and $c_t(a) = 1$ otherwise. This will give your lowest probability action (which has probability at most $\frac{1}{n}$) a cost of 0, and the actions that you have at least a $\frac{n-1}{n}$ probability of choosing a cost of 1.

In this case, the benchmark that we defined would be 0, since for each action it would make a choice that gets 0 in cost.

However, the expected value of the algorithm would be

$$E[cost_{alg}] = \sum_{t} \sum_{A} p_t(a) c_t(a) \geq (1 - \frac{1}{n})T$$

Since the inner sum has 0 for the lowest probability action, and 1 for everything else. Note that probability of least possible action will be at most $\frac{1}{n}$

Thus, compared to the benchmark, we see that

$$\frac{E[cost_{alg}] - benchmark}{T} = 1 - \frac{1}{n}$$

So, with at least two actions (which is the simplest non-trivial case), we see that the regret does not shrink with each time step.

This benchmark was very unrealistic, so we cannot even hope to get close to it. Next, we are going to define a better benchmark that we will use. □

2.3 Best Fixed Action in Hindsight Benchmark

In this section, we will define the benchmark that we will be using. It was noted that this benchmark has connections to equilibria, which we will discuss next lecture.

The benchmark that we will be using is the best fixed action in hindsight. Intuitively, our algorithm should learn over time which fixed action is better. Formally, we define this benchmark as $\min_{a \in A} \sum_{t=1}^{T} c_t(a)$.

Using our new benchmark, we will now define external regret.

Definition 2. *The external regret of an online learning algorithm is defined as*

$$Regret_{alg}^{T} = \frac{1}{T}(\sum_{t=1}^{T} E[cost_{alg}(t)] - \min_{a \in A} \sum_{t=1}^{T} c_t(a))$$

Thus, we say that an algorithm has vanishing external regret (or no external regret) if

$$Regret_{alg}^{T} \xrightarrow{T \to \infty} 0 \quad \forall \text{adversaries}, \forall c_t(a)$$

Thus, no matter how clever the adversary, the average cost that you incur with time is only vanishingly bigger than this benchmark.

3 Follow the Leader Algorithm

In this section, we make our first attempt toward an algorithm with vanishing external regret. However, this algorithm will not be successful.

The algorithm called *Follow the Leader (FTL)* works as follows:

Algorithm 3 Follow the Leader

Input: $c_{t'}(a) \quad \forall a \in A, t' = 1 \ldots t-1$
Output: $a_t \in \text{argmin}_{a \in A} \sum_{t'=1}^{t-1} c_{t'}(a)$.

Intuitively, this algorithm chooses an action that minimizes the historical cost up to time $t-1$, so an action with the minimum total cost so far.

However, this algorithm does not have vanishing external regret. In fact, we can state a stronger theorem that will include this algorithm.

Theorem 3. *No deterministic algorithm has vanishing external regret.*

Proof. Recall that the adversary has access to the same history as the algorithm. Thus, the adversary knows your deterministic algorithm, so the adversary can simulate your algorithm, determine a_t, and use this information to set the cost. The adversary will thus set $c_t(a_t) = 1$ and $c_t(a) = 0$ for $a \neq a_t$. The cost of every action you choose will be 1, and the cost of every other action will be 0. Thus, $cost_{alg}^T = T$.

Now, we consider how well the benchmark would do in this case. There must be at least one action, a^*, that you choose with the least frequency, at frequency at most $\frac{T}{N}$. Thus, the minimum cost of the action in hindsight would be

$$\min_{a \in A} \sum_t c_t(a) \leq \sum_t c_t(a^*) \leq \frac{T}{N}.$$

Thus, the regret of the algorithm is greater than or equal to $1 - \frac{1}{n}$.

□

3.1 Ideas for improving FTL

We want to tweak FTL so that we balance historically good actions (exploitation) with being unpredictable (exploration) and giving poor performing actions another chance. FTL is an algorithm that is an example of exploitation because it solely picks historically good actions. On the other hand, an algorithm that would be just exploration would be choosing actions uniformly at random every time, ignoring history.

The intuition for the algorithm that we will propose in the next section is to choose an action randomly, where historically better actions are exponentially more likely than historically poor performing actions to be chosen. This algorithm will maintain a weight for each action and multiply this weight by $1 - \epsilon c_t(a)$ for each time step t. The higher the cost, the more the weight of the action decreases. If the cost is small, the weight will not change much, and if the cost is negative, then the weight will go up. We assume that $\epsilon \in (0, 1/2)$, and it is referred to as the "learning rate", which will be optimized later. Intuitively, the larger the value ϵ, the more sensitive you are to what is happening, so the closest you are to FTL. On the other hand, if $\epsilon = 0$, then you are not learning at all and are just uniformly randomizing.

4 Multiplicative Weights Algorithm

Recall, the main ideas for improving FTL were:

- Maintain weight for each action a and multiply this weight by $w_a = (1 - \epsilon c_t(a))$ at each time stamp

- Choose action a with $p_t \propto w_a$
- $\epsilon \in (0, \frac{1}{2})$ is the learning rate

Based on these ideas, we present Algorithm 4, the Multiplicative Weights Algorithm.

Algorithm 4 Multiplicative Weights Algorithm

let $w_i(a)$ be weight of action a at time i
let $A = \{a_1 \dots a_n\}$ be the set of n actions
Initialize: $w_1(a) \leftarrow 1, \forall a \in A$
for $t = 1$ to $t = T$ **do**
  $W_t \leftarrow \sum_{\forall a \in A} w_t(a)$
  $p_t(a) \leftarrow \frac{w_t(a)}{W_t}, \forall a \in A$
  (After learning c_t)
  $w_{t+1}(a) = w_t(a)(1 - \epsilon c_t(a))$ (weight update)
end for

Note that multiplication factor $1 - \epsilon c_t(a)$ leads to exponential update in weights as $1 - \epsilon c_t(a)$ can be approximated with $e^{-\epsilon c_t(a)}$ (for small ϵ). In the multiplicative weights algorithm note that if c_t is more, w_{t+1} is less and hence good actions (i.e. actions with low cost) will have more weight. Also note that if adversary decides to make one action better than the other, it cannot do so without increasing the probability of that action.

Next we try to prove the regret bounds for Multiplicative weights algorithm. Our motive is to develop an algorithm for online learning with sub-linear regret bounds (see definition 2).

Let,

$$W_t = \sum_{\forall a \in A} w_t(a) \tag{1}$$

be total weight at time t

$c_1 \dots c_T$ are adversary's choice of the cost function. Cost function can be anything but $c_t(a)$ is independent of p_t

$$p_t(a) = \frac{w_t(a)}{W_t} \tag{2}$$

Define,

$$\bar{C_t} = E[cost^t_{MW}] = \sum_{\forall a \in A} p_t(a) * c_t(a) \tag{3}$$

$$\bar{C} = E[cost_{MW}] = \sum_{t=1}^{T} \bar{C}_t = \sum_{t=1}^{T} \sum_{\forall a \in A} p_t(a) * c_t(a) \tag{4}$$

Next, we will present 3 lemmas (Lemmas 4-6) that will help us prove that Multiplicative weights is a no-external regret algorithm.

Lemma 4. $W_{t+1} = W_t(1 - \epsilon \bar{C}_t)$

Intuitively, we can say that if the algorithm does well ($p_t(a)$ is large for a where $c_t(a)$ is small), then the weights will stay constant, but if the algorithm performs poorly, then the total weight of the actions is going to drop a lot. Also, W_t is normalizing denominator in $p_t(a)$.

Proof.

$$\begin{aligned}
W_{t+1} &= \sum_{a \in A} w_{t+1}(a) && \text{(By eq 1)} \\
&= \sum_{a \in A} w_t(a)(1 - \epsilon c_t(a)) && \text{(By algorithm 4)} \\
&= \sum_{a \in A} w_t(a) - \epsilon \sum_{a \in A} w_t(a) c_t(a) \\
&= W_t - \epsilon W_t \sum_{a \in A} \frac{w_t(a)}{W_t} c_t(a) \\
&= W_t - \epsilon W_t \sum_{a \in A} p_t(a) c_t(a) && \text{(By eq 2)} \\
&= W_t - \epsilon W_t \sum_{a \in A} \bar{C}_t && \text{(By eq 3)} \\
&= W_t(1 - \epsilon \bar{C}_t)
\end{aligned}$$

□

Lemma 5. $W_{t+1} \leq n e^{(-\epsilon \bar{C})}$

Lemma 5 says that the total weight can not drop too drastically. Note that it says weights are going to drop at a rate less than the exponential of average cost.

Proof.

$$\begin{aligned}
W_{t+1} &= W_t(1 - \epsilon \bar{C}_t) && \text{By lemma 4} \\
&\leq W_t * e^{(-\epsilon C_t)} && (1 - x \leq e^{-x}) \\
&\leq W_1 * e^{\sum_{t=1}^{T} (-\epsilon C_t)} \\
&\leq n * e^{-\epsilon \bar{C}} && \text{(By eq 4, } w_1(a) = 1)
\end{aligned}$$

□

Lemma 6. *Let C^* be the lowest regret with fixed action i.e. $C^* = min_{a\in A}\sum_{t=1}^{T} c_t(a)$ then, $W_{t+1} \geq e^{(-\epsilon C^* - \epsilon^2 t)}$.*

The intuition behind Lemma 6 is that the total weight is going to drop at least exponentially as the cost of best fixed action in hindsight as this will contribute something to the weight.

Proof.

$$W_{t+1} = W_t(1 - \epsilon \bar{C}_t) \qquad (\text{By lemma 4})$$

Let a^* be the best action in hindsight then,

$$C^* = \sum_{i=1}^{t} c_i(a*) = min_{a\in A}\sum_{i=1}^{t} c_i(a) \qquad (\text{By definitions})$$

Now,

$$W_{t+1} = \sum_{a\in A} w_{t+1}(a) \geq w_{t+1}(a^*) \qquad (\text{Since weights are positive})$$

Consider $w_{t+1}(a^*)$

$$\begin{aligned}
w_{t+1}(a^*) &= w_t(a^*)(1 - \epsilon c_t(a^*)) && (\text{By weight update rule}) \\
&= 1 * \prod_{i=1}^{t}(1 - \epsilon c_i(a^*)) \\
&\geq \prod_{i=1}^{t} e^{-\epsilon c_i(a^*) - \epsilon^2 c_i^2(a^*)} && \text{Since, } 1 - x \geq e^{-x-x^2} \\
&\geq e^{-\epsilon \sum_{i=1}^{t} c_i(a^*) - \epsilon^2 \sum_{i=1}^{t} c_i^2(a^*)}
\end{aligned}$$

$$\sum_{i=1}^{t} c_i^2(a^*) \leq t \qquad \text{Since } c_i(a) \in [-1, 1]$$

$$\begin{aligned}
e^{-\sum_{i=1}^{t} c_i^2(a^*)} &\geq e^{-t} \\
W_{t+1} &\geq w_{t+1}(a*) \geq e^{-\epsilon \sum_{i=1}^{t} c_i(a^*) - \epsilon^2 \sum_{i=1}^{t} c_i^2(a^*)} \\
&\geq e^{-\epsilon C^* - \epsilon^2 T}
\end{aligned}$$

□

Theorem 7. *Multiplicative Weights algorithm is a no-external regret algorithm. In particular, for suitable choice of ϵ, we get*

$$Regret_{MW}^{T} \leq 2\sqrt{\frac{ln(n)}{T}}$$

Note that, $lim_{T\to\infty} Regret_{MW}^{T} \to 0$.

Proof.

$$\begin{aligned} e^{-\epsilon C^* - \epsilon^2 T} \leq W_{T+1} &\leq n e^{-\epsilon \bar{C}} && \text{(By lemma 5, 6)} \\ -\epsilon C^* - \epsilon^2 T &\leq ln(n) - \epsilon \bar{C} \\ \epsilon(\bar{C} - C^*) &\leq ln(n) + \epsilon^2 T \end{aligned}$$

Recall,

$$Regret_{MW}^{T} = \frac{\bar{C} - C^*}{T}$$

so,

$$\begin{aligned} Regret_{MW}^{T} &\leq \frac{ln(n) + \epsilon^2 T}{\epsilon T} \\ &\leq \frac{ln(n)}{\epsilon T} + \epsilon \\ &\leq 2\sqrt{\frac{ln(n)}{T}} && \text{since, } \frac{ln(n)}{\epsilon T} + \epsilon \text{ at } \epsilon = \sqrt{\frac{ln(n)}{T}} \end{aligned}$$

□

So, regret for multiplicative weights algorithm is at max $2\sqrt{\frac{ln(n)}{T}}$ and occurs when $\epsilon = \sqrt{\frac{ln(n)}{T}}$, $n = |A|$. So if there are more actions, the algorithm needs to run for longer time steps to ensure the regret is bounded.