Lecture 10: May 30, 2006

Lecturer: Yishay Mansour Scribe: Anat Halpern, Galina Ryvchin, Guy Tannenbaum †

10.1 External Regret - Reminder

Let us recall *external regret*. We have a single player, playing against an adversarial environment.

The model (single player)**:**

- Actions $A = \{a_1 \dots a_m\}$
- Time dependent loss function, (can also be defined as a gain function):
  l_i^t - loss of action a_i at time t

The game:

a. For each step t, the player chooses a distribution $p^t \in \Delta(A)$

b. The adversary decides on the losses $l^t = (l_1^t \dots l_m^t)$, where $l_i^t \in [0, 1]$

c. The player's loss at time t is $\sum_{i=1}^m l_i^t p_i^t$, and
the loss up to time T is

$$L_{ON}^T = \sum_{t=1}^{T} \sum_{i=1}^{m} l_i^t p_i^t$$

The goal: Minimize the total loss, L_{ON}^T.

We define: $L_i^T = \sum_{t=1}^T l_i^t$ as the loss of playing always the action a_i.
External regret was defined as a measure of optimality (comparing our performance to the performance of the single best action):

$$ER = L_{ON}^T - \min L_i^T$$

We showed that,

$$ONLINE \leq \min_i \{L_i + 2\sqrt{Q_i \ln m}\},$$

†based in part on the scribe notes of Eitan Yaffe, and Noa Bar-Yosef from 2003/4.

$$\text{where } Q_i = \sum_{t=1}^{m} (l_i^t)^2 \leq L_i.$$

In this lecture we show algorithms that have "stronger" guarantees.

10.2 Correlated Equilibrium

The model:

- N players - $\{1 \dots n\}$
- Actions of player i - A_i
- Joint action - $A = A_i \times \dots \times A_n$
- Utility function of player i - $u_i : A \to \mathbf{R}$
  We will later look at $u_i : A \to [0, 1]$

10.2.1 Internal Regret

In this section we define *pure* and *correlated equilibria* in a different manner than previously. We use a new measure, called *Internal Regret*:

$$IR_i(a, x, y) = \begin{cases} u_i(a^{-i}, y) - u_i(a), & a_i = x \\ 0, & a_i \neq x \end{cases}$$

The meaning is measuring the loss of player i caused by playing action x instead of action y.

Definition $a \in A$ is a ***pure equilibrium*** if:

$$\forall i, \forall x, y \in A_i : IR_i(a, x, y) \leq 0$$

Definition Let $Q \in \Delta(A)$ be a distribution over the joint actions. We adapt the *regret* definition in the following manner:

$$IR_i(Q, x, y) = E_{a \sim Q}[IR_i(a, x, y)]$$

Using this definition, Q is a ***correlated equilibrium*** if:

$$\forall i, \forall x, y \in A_i : IR_i(Q, x, y) \leq 0$$

The existence of a correlated equilibrium is guaranteed - we gave a direct proof using zero sum games.

We can formalize this also in a different manner:
Let us define the set of functions $F_i = \{f : A_i \to A_i\}$. These are static (time independent) mappings - we are still talking about the one-shot game. Each function maps A_i, the actions of player i, in some way to A_i. Note that when all actions are mapped to the same action, we get the same setting in which external regret is defined.
Then Q is a *Correlated Equilibrium* if:

$$\forall i \in N, \forall f \in F_i : E_{a\sim Q}[u_i(a)] \geq E_{a\sim Q}[u_i(a^{-i}, f(a_i))]$$

10.3 ϵ-Correlated Equilibrium

The definition of a correlated equilibrium can be relaxed, by requiring only that a player cannot gain more than ϵ, as a result of the action change.

Definition Q is ϵ***-Correlated Equilibrium*** if:

$$\forall i \in N, \forall f \in F_i : E_{a\sim Q}[u_i(a)] \geq E_{a\sim Q}[u_i(a^{-i}, f(a_i))] - \epsilon$$

10.3.1 Swap Regret

We define the *Swap Regret* as follows:

$$SR(Q, i, f) = \sum_{a_i \in A_i} Pr[a_i \sim Q] \cdot IR_i(Q, a_i, f(a_i)) \leq \epsilon$$

and in general:

$$SR(Q) = \max_{i\in N} \max_{f\in F_i} SR(Q, i, f)$$

Thus, Q is an ϵ***-correlated equilibrium*** if:

$$SR(Q) \leq \epsilon$$

We can extend this definition for a **series of games**: For a series of joint actions $\vec{a} = a^1 \cdots a^T$, we define in a similar manner:

$$SR(\vec{a}) = \max_{i\in N} \max_{f\in F_i} \sum_t IR_i(a^t, a_i^t, f(a_i^t))$$

Claim 10.1 *If $SR(\vec{a}) \leq \epsilon \cdot T$, then the distribution Q is an ϵ-correlated equilibrium, where:*

$$Q(z) = \begin{cases} \frac{1}{T}, & z = a^t \\ 0, & otherwise \end{cases}$$

We would now like to see how players can be driven to create such a series of actions. That is, we'd like to learn more about the dynamics of reaching an equilibrium.

10.3.2 Reduction of External Regret to Swap Regret

In order to achieve equilibrium, we need an algorithm which minimizes swap regret. In this section, we will use our knowledge of algorithms that use the external regret measure, as a way to create an algorithm for swap regret. For that, we will construct a reduction: external regret $\mapsto$ swap regret.
Let's look at a single player who is only aware of his own losses (note that we switched to loss terminology instead of utility). Assume the number of actions of the single player is $|A_i| = m$. We will use m external regret algorithms $B_1 \dots B_m$ as shown in Figure 10.1.

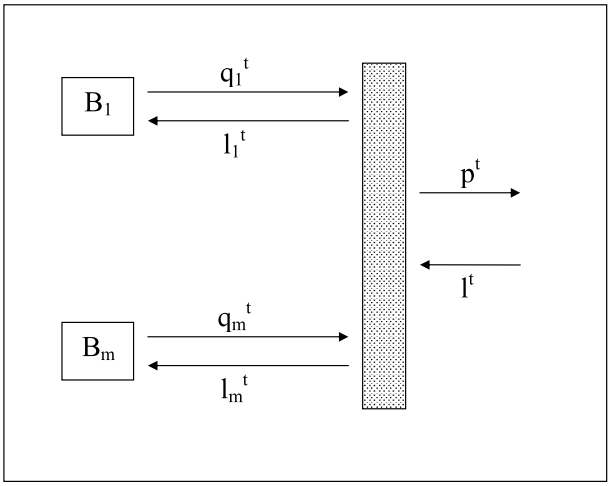

Figure 10.1: Reduction of External Regret to Swap Regret algorithm

Recall the *ER assumption*:
For any series of t losses $\{l_j^t\}$,

$$L_{ON}^T = \sum_{t=1}^{T} l_{ON}^t \leq \sum_{t=1}^{T} l_{a_j}^t + ER = L_j^T + ER$$

This implies that

$$\forall j \in A_i, L_{ON} \leq L_j + ER$$

We will construct our algorithm using m external regret algorithms, each guaranteeing an external regret of at most ER_i. Intuitively, each external regret algorithm will be responsible of a single action (there are m algorithms - one for each possible action of the player). Each algorithm outputs a vector q_i^t of what it would like to play, and we need to return to each separate external regret algorithm its loss, e_i^t. We need to wrap these algorithms in some sort of interface which will calculate the distribution p^t and receive the loss. Thus we have two important actions to do:

a. Calculate p^t from $\vec{q_1^t}, \cdots, \vec{q_m^t}$

b. "Distribute" $\vec{l^t}$ - return to B_i its loss vector $\vec{l_i^t}$.

Let's start with distributing the loss: we simply return to B_i its loss vector $\vec{l_i^t} = p_i^t \cdot \vec{l^t}$.
We need to define now how we combine the separate "recommendations" q_i^t to get the distribution p. We construct a matrix

$$Q = \begin{pmatrix} q_{11} & \cdots & q_{1m} \\ \vdots & & \vdots \\ q_{m1} & \cdots & q_{mm} \end{pmatrix} \begin{matrix} \leftarrow \vec{q_1^t} \\ \\ \leftarrow \vec{q_m^t} \end{matrix}$$

We choose p such that $pQ = p$.

Intuition: p is the output of our algorithm - its meaning is the distribution over actions.We can choose an action in 2 ways:

- choose an action directly from among all possible actions.
  This is the output of the algorithm.

- choose an algorithm B_i first, according to p, and then select an action according to $\vec{q_i^t}$.

In defining p as above, we ensure that both ways are indeed equivalent. It's possible to prove in several different ways that a solution exists.

Analysis:
The loss that B_i "sees" is:

$$(p_i^t \cdot \vec{l^t})\vec{q_i^t} = p_i^t(\vec{q_i^t} \cdot \vec{l^t})$$

B_i is an external regret algorithm with ER. For each B_i and for each action j we have a bound on the external regret (assuming all algorithms guarantee the same regret - ER):

$$\sum_{t=1}^{T} p_i^t(\vec{q_i^t} \cdot \vec{l^t}) \leq \sum_{t=1}^{T} p_i^t \cdot l_{f(j)}^t + ER.$$

When we sum up the losses over the different B_i, we get that for any point in time:

$$\sum_{i=1}^{m}(p_i^t \cdot \vec{q_i^t}) \cdot \vec{l^t} = \sum_{i=1}^{m} p_i^t(\vec{q_i^t} \cdot \vec{l^t}) = \vec{p^t} \cdot Q \cdot \vec{l^t} = \vec{p^t} \cdot \vec{l^t} = l_{ON}^t$$

Therefore, if we want to find the loss of $ONLINE$, we need to sum all the B_i's losses over time. We get in total:

$$L_{ON}^T = \sum_{i=1}^{m} L_{B_i}^T \leq \sum_{i=1}^{m} L_{B_i,f(i)}^T + ER = L_{ON,f} + ER$$

Where

$$L_{ON,f}^T = \sum_{i=1}^{m}\sum_{t=1}^{T} p_i^t \cdot l_{f(i)}^t$$

$$L_{B_i,f(i)} = \sum_{t=1}^{T} p_i^t \cdot l_{f(i)}^t$$

Recall that we previously proved that: $ER \sim \sqrt{T \log m}$ so by summing over all ER_i we have that:

$$SR \leq m\sqrt{T \log N}$$

This bound can be easily improved to $SR \leq \sqrt{mT \log N}$ by noticing that the sum of the losses of all the B_i's is bounded by T.

10.4 Dynamics of Reaching Equilibrium

In this section, we will study dynamics of play in multi-player game that guarantee convergence to Nash Equilibrium. We will consider dynamic processes that are subject to the natural restriction of uncoupleness: we require that the dynamics of play be "uncoupled" among the players, that is, the strategy of every player does not depend on the utility functions of the other player.
Hart and Mas-Collel (2003) showed that there are games for which no deterministic uncoupled dynamics lead to Nash equilibrium.
We will allow random actions and will work in discrete time framework. Thus, we consider repeated play of a given game, under assumption that each player observes the actions of all players; as for payoffs, he knows only his own utility function.

10.4.1 Model

- $N = \{1, ..., n\}$ - players
- A_i - actions of player i
- A - joint actions of players $A = A_1 \times ... \times A_n$
- u_i - the utility function of player i: $u_i : A \to [0, 1]$
- $a_i^t \in A_i$ - action of player i at time t

10.5 Algorithm for Pure Equilibrium

We will see an algorithm that guarantees convergence of play to a pure Nash equilibrium in every game where a pure equilibrium exists.
The idea behind this algorithm is to use randomization until Nash equilibrium is hit by pure chance and then stop there. In order to recognize a Nash equilibrium, players need to keep history of the past actions. If a player does not benefit from changing his action and he sees that no other player changed his action since last step, he assumes that a Nash equilibrium is reached. Otherwise, he tries to move to another state by choosing an action uniformly at random. If only players who are not playing one of their best response actions would change it at each step, the algorithm could perform infinitely many steps without reaching an equilibrium state. To solve this problem players randomize their actions not only when they can benefit themselves from the change but also when they suppose that other players are not in Nash equilibrium, i.e., when others keep changing their actions. Thus, a pure equilibrium needs to be hit twice in a row until all players recognize it.

10.5.1 Algorithm

- At the first step, each player i plays a random $a_i^1 \in A_i$
- at the step $t + 1$, player i sees $a^t = a_1^t, ..., a_n^t$
  if $a_i^t \in BR_i(a_{-i}^t)$ and $a^t = a^{t-1}$, then he does not change his action, $a_i^{t+1} = a_i^t$
  else he plays a random $a_i^{t+1} \in A_i$

It is easy to see that if a pure Nash equilibrium is reached and holds for two steps, it is never changed. This is the absorbing state of the process.

Claim 10.2 *If a is a pure Nash equilibrium and $a = a^t = a^{t-1}$ then $a^{t'} = a$ for every $t' > t$.*

10.5.2 Convergence of the Algorithm

- if at time t, a^t is not pure Nash Equilibrium since $a_i^t \notin BR_i(a_{-i}^t)$ then $a^{t+1} \neq a^t$ with probability $\frac{|A_i|-1}{|A_i|} \geq \frac{1}{2}$
  This means that if some player wants to change his action, with high probability the others will see this at the next step. Then, they also will try to change their actions.
- if $a^{t+1} \neq a^t$ then at time $t+2$ the players will choose pure Nash equilibrium a with probability $\frac{1}{|A|}$
- if at time $t+3$, a is picked again, the process reaches the absorbing state.

Thus, from any state there is a positive probability to reach the absorbing state in at most three steps, and therefore this state will be reached with probability 1.

From the claim 10.2 and the convergence properties of the algorithm immediately follows:

Theorem 10.3 *The algorithm converges to a Nash equilibrium and the expected convergence time is* $O(|A|^2)$.

Note that if each player has m actions, $|A| = m^n$. Thus, the time complexity of the algorithm is exponential in number of players.

10.6 $\epsilon - Nash$

We now switch to mixed strategies, and present an algorithm that converges to an $\epsilon - Nash$ equilibrium.

10.6.1 Model

- Assume each player sees the expected value of the strategy $E[u_i(P_1...P_n)]$, where P_i is a mixed strategy of player i.

10.6.2 Algorithm

- At the first step, player i chooses a random $P_i \in \Delta(A_i)$
- player i does the following at time t:
  if $P_i^t \notin BR_{i,\epsilon}(P^{-i})$ then pick P_i^{t+1} at random.
  (Note that the player only checks whether his action was in $BR_{i,\epsilon}$. This is because the

probablity of exactly hitting a Nash equilibrium is 0, and the goal of this algorithm is to find an $\epsilon - Nash$ equilibrium.)
else:
if $P^t \approx P^{t-1}$ then choose $P_i^{t+1} = P_i^t$
else pick P_i^{t+1} at random.

- We define $P \approx Q \iff \sum |P_i - Q_i| \leq \epsilon$
  (The idea behind this check is that if the previous action was an $\epsilon - Nash$ equilibrium, and this action is close enough to it, it will also be one.)

10.6.3 Convergence

The proof of convergence to $\epsilon - Nash$ equilibrium is similar to the pure equilibrium case. The difference is that in this case one needs to show what is the probablity that if all players pick P_i^t at random the resulting P^t is an $\epsilon - Nash$ equilibrium.

Claim 10.4 *For two probability functions* $D_1, D_2 \in \Delta(X)$
$\sum_{x \in X} |D_1(x) - D_2(x)| \leq \epsilon \Longrightarrow \forall f : X \to [0,1] \ |E_{D_1}(f) - E_{D_2}(f)| \leq \epsilon$

Proof:
$|E_{D_1}[f] - E_{D_2}[f]| = |\sum_{x \in X} D_1(x)f(x) - D_2(x)f(x)| = |\sum_{x \in X} f(x)(D_1(x) - D_2(x))| \leq$
$\sum_{x \in X} |f(x)| \, |D_1(x) - D_2(x)| \leq \sum_{x \in X} |D_1(x) - D_2(x)| \leq \epsilon$ □

Claim 10.5 *For two product probabilities* $p = (p_1 \cdot p_2 \cdot ... \cdot p_m) \ q = (q_1 \cdot q_2 \cdot ... \cdot q_m)$
$\|p - q\|_1 = \sum_{a \in A} |p(a) - q(a)| = \sum_{i=1}^m |p_i - q_i|$

Proof:
Consider switching from $p = (p_1 ... p_n)$ to $p' = (q_1, p_2, ... p_n)$
$\|p - p'\| = \sum_{a_1 \in A_1, a^{-i} \in A^{-i}} |p(a_1)p^{-i}(a^{-i}) - p'(a_1)p^{-i}(a^{-i})| = \sum_{a_1 \in A_1} |p_1(a_1) - q_1(a_1)| \cdot \sum p^{-i}(a^{-i}) =$
$\|p_1 - q_1\|_1$ □

Claim 10.6 *Let* $\epsilon - EQ$ *be the collection of all* $\epsilon - Nash$ *Equilibria.*
$\mu(\epsilon - EQ) \geq (\frac{\epsilon}{n})^n$ *(with respect to uniform probablity)*

Proof: Let p be a Nash equilibrium

Define $Q_i = \begin{cases} [p_i, p_i + \frac{\epsilon}{n}] & \text{If } p_i < \frac{1}{2} \\ [p_i - \frac{\epsilon}{n}, p_i] & \text{If } p_i \geq \frac{1}{2} \end{cases}$

$\times_{i=1}^N Q_i \subseteq \epsilon - Nash \Rightarrow \mu(\epsilon - EQ) \geq \mu(\times Q_i) = (\frac{\epsilon}{n})^n$ □

- And this concludes the proof for the convergence of the algorithm for the case where the profit is exactly the utility.