Computational Game Theory **Spring Semester, 2003/4**

Lecture 8: May 11, 2004

Lecturer: Yishay Mansour *Scribe: Eitan Yaffe, Noa Bar-Yosef*

8.1 Regret

Our goal is to build a strategy with good performance when dealing with repeated games. Let us start with a simple model of regret.

8.2 Basic Model

Assuming that the opponent has the same stochastic strategy at each step, how should we play?
Let's formalize this:

- N actions
- For each step t, we choose a distribution p^t over the N actions
- For each step, we have a loss l^t where $l^t(i) \in [0,1]$ is the loss from action i
- Our loss is $\sum_{i=1}^{N} p^t(i) l^t(i)$

Note that we do not rely on the number of opponents or on their actions. Once we assume that the opponent is constant and does not depend on us, then the opponent's influence is only on the cost of each action.
Our goals:

- Bring the loss to a minimum
- Choose the best action (the opponent does not change throughout time)

8.3 A Greedy Algorithm

One way to implement our goal is by using the greedy algorithm:

- For the $t+1$ step, we will calculate for each i:

$$L_i^t = \sum_{k=1}^{t} l^k(i) = L_i^{t-1} + l^t(i)$$

- For the $t+1$ step, we will chose the best action that we had up until now:

$$a^{t+1} = arg\min_i L_i^t$$

We can see that this is not the optimal algorithm, because although the history gives us a good hint regarding the probability of each action, this does not necessarily give the best action, but rather an approximation of the best action.
To simplify, let's assume that $l^t(i) \in \{0,1\}$ and define $p_i = Pr[l^t(i) = 1]$ (the probability of the loss of action i to be 1). The best action is $a^* = arg\min_i p_i$. Thus the average loss is p^* for each step. And so we get that the optimal loss for T steps is p^*T.
The loss of the Greedy algorithm is $L_G^T = \sum_{k=1}^{T} Greedy^k$. Define R (The regret) to be: $R = L_G^T - p^*T$. We get that the average of R is:

$$E[R] = \sum_{k=1}^{T} E[Greedy^k] - p^*$$

We analyze this by looking at each step k, separately. When k is large enough, it converges to p^*.
The average we get for the i-th action is: $\hat{p}_i^k = \frac{L_i^k}{k}$.
Thus, using the Chernoff bound, we get:

$$Pr[|p_i - \hat{p}_i^k| \geq \epsilon] \leq e^{-2\epsilon^2 k}$$

If for all the actions $Pr[|p_i - \hat{p}_i^k|] \leq \epsilon$, then:

$$E[Greedy^k] - p^* \leq 2\epsilon + Ne^{-2\epsilon^2 k}$$

Taking $\epsilon = \sqrt{\frac{\ln Nk}{k}}$, results in: $O(\sqrt{\frac{\ln NT}{k}})$
Now we just need to sum up all the losses:

$$\sum_{k=1}^{T} \sqrt{\frac{\ln NT}{k}} \approx \sqrt{\ln NT} \int_1^T \frac{1}{\sqrt{k}} dk = O(\sqrt{T \ln NT})$$

Meaning that the extra loss (the regret) is:

$$E[R] = O(\sqrt{T \ln NT})$$

Note that this bound is not tight.

8.4 External Regret

In the above analysis we assumed that the system does not change throughout time. We would like to change this assumption, but this means we must change our analysis (e.g. we cannot use p^* since it changes over time and isn't defined). First we shall consider comparing to the performance of OPT. This turns out to be a not a very informative measure. Then we shall introduce the *External Regret* measure. Consider the following example:

8.4.1 "Bad" example

On each step OPT chooses a random $i \in N$ such that: $l^t_{j \neq i}(j) = 1$ and $l^t(i) = 0$. We can see that the average loss for any online algorithm is at least $1 - \frac{1}{N}$ (on average), while OPT's loss is 0.

8.4.2 Definition

Instead of comparing to OPT we can compare our performance to the performance of the single best action: $L_{min} = min_i L^T_i$. In general, for any Algorithm (or Hypothesis) denoted H, we define its loss on step t to be L^t_H and its overall loss to be $L^T_H = \sum_{t=1}^{T} L^t_H$. The *External Regret* is defined as follows:

$$R^{ext} = L^T_H - L_{min}$$

8.4.3 Analyzing the greedy algorithm

The loss of the Greedy algorithm over T steps is L^T_G. Reducing R^{ext} means coming as close as possible to L_{min}.

Claim 8.1 $L^T_G \leq N(L_{min} + 1)$

Proof: For simplicity we assume that $l^t(i) \in \{0, 1\}$. At step k, let $b_k = \min_i L^k_i$ and $n_k = |\{i : L^k_i = b_k\}|$. We define a lexicographic order over the pairs (b_k, n_k). On each step that *Greedy* losses 1, either b_k increases by 1 or n_k decreases by 1. Note that n_k can decrease only $N - 1$ times, consecutively. Therefore the loss of *Greedy* is bounded by $b_k N$. □

The problem of deterministic algorithms is that we can always create a large loss. The following (bad) example, is true for every deterministic algorithm.

8.4.4 Example

At the k-th step, the opponent's algorithm chooses action a_k. The loss is $l^k(a_k) = 1$ for action k and for the rest of the actions it is 0. The loss of the online deterministic algorithm is thus T, while there exists an action whose loss is $\frac{T}{N}$. (Since the sums of 1's is T, then the average is $\frac{T}{N}$).

8.4.5 A stochastic strategy

We will examine a strategy that has an (expected) external regret of $O(\sqrt{L_{\max} \log N} + \log N)$. What we shall do is build a distribution over the actions, dependant on the Regret.
We define l_H^t as the loss of the online algorithm in the t-th step: $l_H^t = \sum_{i=1}^{N} p^t(i) l^t(i)$ and $R_a^T = \sum_{t=1}^{T} [l_H^t - l^t(a)]$. We also define a "pseudo-regret" where we multiply our loss by $0 < \beta < 1$:

$$\tilde{R_a^T} = \sum_{t=1}^{T} [\beta l_H^t - l^t(a)]$$

It is easy to see that $R_a - \tilde{R}_a$ is small for $\beta \approx 1$. We now build the exponential weights that are dependant on $\tilde{R}_a$:

$$w_a^0 = 1$$
$$w_a^{t+1} = w_a^t \beta^{l^t(a) - \beta l_H^t}$$

According to these weights, we can define the probabilities by normalizing:

$$W^{t+1} = \sum_{a \in N} w_a^{t+1}$$

$$p_a^{t+1} = \frac{w_a^{t+1}}{W^{t+1}}$$

Claim 8.2 $0 \leq \sum_{a \in N} w_a^t \leq N$

In other words, all the weights are positive and do not "run off" to very large sizes.
The proof of this claim will appear shortly, but in the meanwhile until then we will assume its correctness.
Using the claim, we get that for each action a:

$$w_a^T = \beta^{L_a^T - \beta L_H^T} = \beta^{-\tilde{R}} \leq N$$

Comparing $\beta^{L_a^T - \beta L_H^T} \leq N$, and taking ln, results in:

$$(L_a^T - \beta L_H^T) \ln \beta \leq \ln N$$

Dividing by $\ln(\beta)$:

$$L_a^T - \beta L_H^T \geq -\frac{\ln N}{\ln \frac{1}{\beta}}$$

$$\frac{L_a^T}{\beta} + \frac{\ln N}{\beta \ln \frac{1}{\beta}} \geq L_H^T$$

Choosing $\beta = 1 - \gamma$, $ln\frac{1}{\beta} \approx \gamma$,

$$L_a^T + \frac{\gamma}{1-\gamma} L_a^T + 2\frac{\ln N}{\gamma} \geq L_H^T$$

Note that $\frac{\gamma}{1-\gamma} L_a^T + 2\frac{\ln N}{\gamma}$ is basically the Regret. Defining $L_{\max} = \max_a L_a^T$ we have:

$$\gamma L_{\max} = \frac{\ln N}{\gamma}$$

$$\gamma = \sqrt{\frac{\ln N}{L_{\max}}} < \frac{1}{2}$$

$$\Rightarrow R = O(\sqrt{\ln N \cdot L_{\max}} + \log N)$$

In each step, we incur a loss of at most 1, thus $L_{\max} \leq T$.
In each step, the opponent chooses some kind of loss that can be dependant on our distribution. Nevertheless, we are able to approach quite well, as we can see:

$$L_H^T \leq L_{\min} + O(\sqrt{T \ln N})$$

where $L_{\min} = \min_a L_a^T$.

We shall now proceed to prove our above claim about our weights that states that:

$$0 \leq \sum_{a \in N} w_a^t \leq N$$

Proof: Trivially, $0 \leq \sum_{a \in N} w_a^t$.
We are left to see that the weights are in fact bounded:

$$l_H^t = \sum_{a \in N} P^t(a) l^t(a) = \sum_{a \in N} \frac{w_a^t}{W^t} l^t(a)$$

$$W^t l_H^t = \sum_{a \in N} w_a^t l^t(a) \tag{8.1}$$

We can give a linear bound for the function β^x (relying on it's convexity) for any $\beta \in [0,1]$. For $x \in [0,1]$ we know that $\beta^x \leq 1-(1-\beta)x$. For $x \in [-1,0]$ we get $\beta^x \leq 1 - \frac{1-\beta}{\beta}|x|$.
Now, by the definition of w_a^{t+1}:

$$\sum_{a \in N} w_a^{t+1} = \sum_{a \in N} w_a^t \cdot \beta^{l^t(a) - \beta l_H^t}$$

By the above properties of beta we get the bound:

$$\leq \sum_{a \in N} w_a^t (1-(1-\beta)l^t(a))(1+(1-\beta)l_H^t)$$

By opening the parenthesis and discarding $(1-\beta)l^t(a)(1-\beta)l_H^t$ since it is negative (the product of positive and negative)

$$\leq \sum_{a \in N} w_a^t - (1-\beta)[\sum_{a \in N} w_a^t \cdot l^t(a)] + (1-\beta)[\sum_{a \in N} w_a^t \cdot l_H^t]$$

Using equation (8.1):

$$\leq \sum_{a \in N} w_a^t \leq N$$

□

8.4.6 Example

We have shown a probabilistic algorithm whose bound is relatively tight. An example for the tightness of its bound follows:
Let the loss of one action be $\frac{1}{2} - \epsilon$ and the loss of the rest of the actions be $\frac{1}{2}$. Taking $\epsilon = \frac{1}{\sqrt{T}}$ and randomly choosing between these two actions, will not be able to incur a loss of less than $\sqrt{T}$ the overall loss.

Up until now we've discussed **External Regret** whereas:

$$R^{ext} = L_H^T - L_{\min}$$

i.e. Our loss is not measured according to our algorithm, but rather in relation to each separate action taken. We can easily define other measures...

8.5 Correlated Equilibrium

- Game with M players
- A_i - N actions of player i
- S^i - The loss function of player i:

$$S^i : A_i \times (\times A_j) \to [0, 1]$$

Definition L et Q be a distribution over the joint actions $(\times A_i)$, such that for each player i and for each action $\alpha \in A_i$:

$$E_{a\sim Q}[S^i(a_i, a^{-i})|a_i = \alpha] \leq E[S^i(b, a^{-i})|a_i = \alpha]$$

In other words, this means that given an action a_i from the distribution Q to player i, then this is also his *best response* to play it!
We can formalize this also in a different manner:
Let us define the function $F : A_i \to A_i$, then Q is a *Correlated Equilibrium* if for each player i and for each F we have:

$$E_{a\sim Q}[S^i(a_i, a^{-i})] \leq E_{a\sim Q}[S^i(F(a_i), a^{-i})]$$

Furthermore we will now define that Q is *Epsilon-Correlated* when for each player i and for each F:

$$E_{a\sim Q}[S^i(a_i, a^{-i})] \leq E_{a\sim Q}[S^i(F(a_i), a^{-i})] + \epsilon$$

This means that F "exchanges" the actions of i according to the suggestions of Q. We bound the gain from using F by ϵ.

We define the *Swap Regret* to be:

$$R^{swap} = \max_F\{L_H - L_{H,F}\} = \sum_{t=1}^{T} p^t(i)[l^t(i) - l^t(F(i))]$$

Claim 8.3 *Let us assume a game with M players where each player plays a strategy that has $R^{swap} \leq R$. Let p be the empirical distribution at time $[1, T]$ (This means that each step has a vector of actions and for each one of these vectors we will give a probability). Then:*

1. *The average loss of the player according to p is his loss in the "game".*
2. *p is ϵ-correlated for $\epsilon = \frac{R}{T}$. This is true because every player can gain no more than R by using F.*

8.5.1 Internal Regret

We also define *Internal Regret*: For $a_i, a_j \in A$, we swap $a_i \rightarrow a_j$. (We swap only between a pair of actions).

8.5.2 Reduction of External Regret to Swap Regret

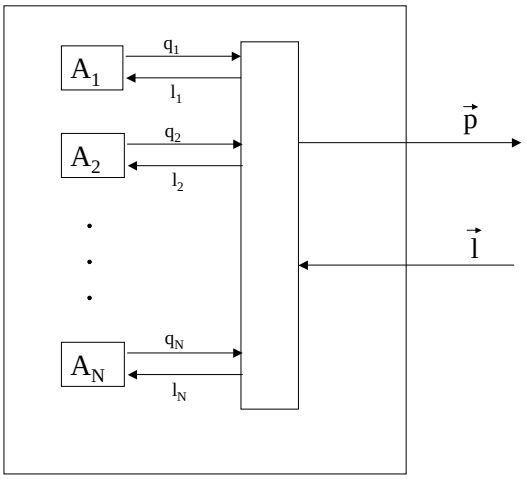

Figure 8.1: Reduction of External Regret to Swap Regret algorithm

Having already discussed R^{ext}, we now present a reduction from R^{ext} to R^{sw}:
For each action i, there will be an algorithm A_i. Intuitively, the goal of A_i is to avoid regret by replacing action i with any other action. We construct an algorithm which combines N algorithms, as shown in figure 8.1. Each algorithm guarantees us of a small R^{ext}. Each algorithm outputs a vector of what it would like to play, and we need to return to each separate algorithm its loss. We need to wrap up these algorithms in some sort of interface which will calculate the distribution and return the loss. Thus we have two important actions to do:

1. Calculate p^t from $\vec{q_1^t}, ..., \vec{q_N^t}$
   For this we will choose a distribution p such that: $p = p \cdot Q$ where $\vec{q_i}$ is the i-th row of Q. Specifically:

$$\forall j \; p_j^t = \sum_{i=1}^{N} p_i^t \cdot q_{i,j}^t$$

   This means that choosing an action j according to p is equivalent to choosing an algorithm A_i according to p, and then choosing action j according to A_i.

2. "Distribute" the loss of $\vec{l^t}$ to $\vec{l_1^t}, ..., \vec{l_N^t}$.
   Upon receiving $\vec{l^t}$, we return $p_i^t \cdot \vec{l^t}$ to A_i. The loss that A_i "sees" is:

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

   Thus, for each A_i and for each action j we have a bound on the regret:

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

   (R_i may be dependant on T, N, or a loss as before, but it is not dependant on the game itself.)
   When we sum up the losses, we get that for any point in time:

$$\sum_{i=1}^{N} p_i^t(\vec{q_i^t} \cdot \vec{l^t}) = p^t \cdot Q \cdot l^t = p^t \cdot l^t = l_H^t$$

   Therefore we get in total:

$$\sum_{i=1}^{N}[\sum_{t=1}^{T} p_i^t(\vec{q_i^t} \cdot \vec{l^t})] = \sum_{t=1}^{T} l_H^t = L_H^T \leq \sum_{i=1}^{N}\sum_{t=1}^{T} p_i^t \cdot l^t(F(i)) + \sum_{i=1}^{N} R_i$$

   However, $\sum_{i=1}^{N}\sum_{t=1}^{T} p_i^t \cdot l^t(F(i)) = L_{H,F}^T$ and so this results in:

$$L_H^T \leq L_{H,F}^T + \sum_{i=1}^{N} R_i$$

   Recall that we previously proved that: $R = O(\sqrt{T \log N} + \log N)$ so by summing over all R_i we have that:

$$R^{sw} = O(N\sqrt{T \log N} + N \log N)$$

   Thus,

$$R = O(\sqrt{L_{\max_i} \log N} + \log N)$$

   And finally,

$$L_H \leq L_{H,F}^T + O(\sqrt{L_{\max_i} \log N} + \log N)$$

   Since in our case, $\sum L_{max_i} \leq T$, the worst case is when $L_{max_i} = \frac{T}{N}$.

Prior knowledge of L_{max}

We need to know L_{max_i} in order to define A_i. We will change our previous algorithm for External Regret so that it won't need to have L_{max} beforehand.
We start with $L_{max} = 1$, and each time we reach the bound, we multiply L_{max} by 2 and start over again.
We now have that the regret is:

$$\sum_{j=1}^{\log L_{\max}} O(\sqrt{2^j \log N} + \log N) = O(\sqrt{L_{\max} \log N} + \log T \log N)$$

It is easy to see that our new bound is a bit worse than the previous, but here we do not need to rely on knowing $L_{\max}$.
Using the new algorithms, we get that the worst case is (still): $L_{\max_i} = \frac{T}{N}$ and thus:

$$L_H \leq L_{H,F} + O(TN \log N + N \log N \log T)$$