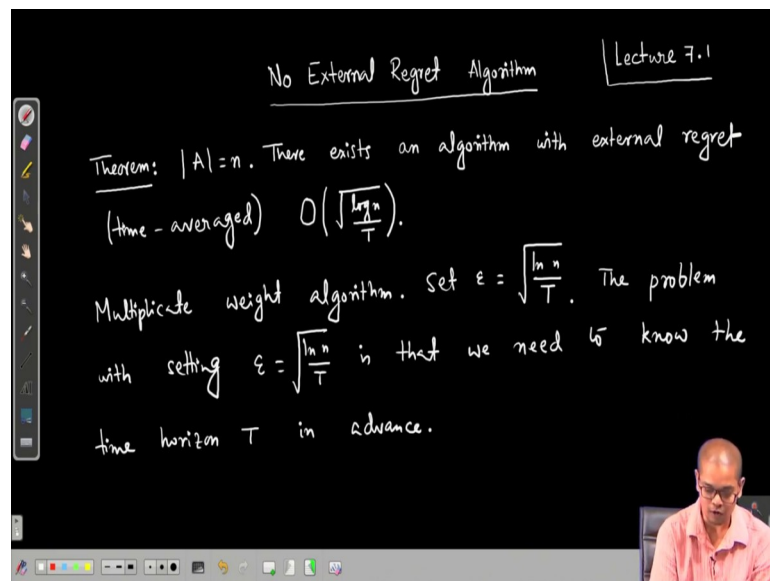


Algorithmic Game Theory
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 31
Multiplicative Weight Algorithm (Contd.)

Welcome. So, in the last couple of lectures we finished our study of equilibriums and we ended up with coordinated equilibrium and coarse correlated equilibrium and they are polynomial time solvable, we can find them in polynomial time. And, then we asked ok, but is there any natural algorithms for finding them and then we started studying various learning algorithms and no regret algorithms, and we will continue that topic of discussion in in these and next few lectures.

(Refer Slide Time: 01:02)



No External Regret Algorithm Lecture 7.1

Theorem: $|A| = n$. There exists an algorithm with external regret (time-averaged) $O\left(\sqrt{\frac{\log n}{T}}\right)$.

Multiplicative weight algorithm. Set $\epsilon = \sqrt{\frac{\ln n}{T}}$. The problem with setting $\epsilon = \sqrt{\frac{\ln n}{T}}$ is that we need to know the time horizon T in advance.

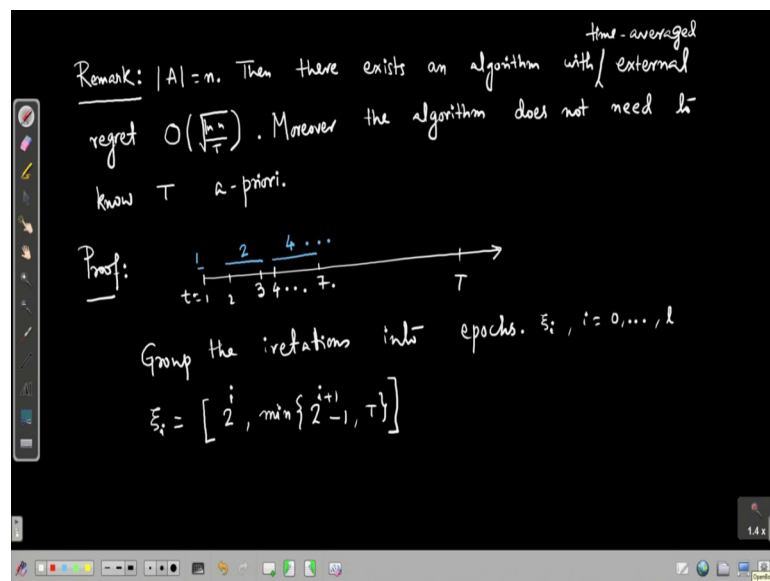
So, we are discussing no regret algorithm, no external regret algorithm; no external regret algorithm or dynamics. And, we had stated and proved the following theorem in our last class that if we have n many actions, then there exists an algorithm with external regret time averaged. Time averaged external regret is $O\left(\sqrt{\frac{\log n}{T}}\right)$.

And, the idea was that we used multiplicative weight algorithm and that algorithm is parameterized by ϵ and we set $\epsilon = \sqrt{\frac{\log n}{T}}$. Now, you see that to run this algorithm we

need to set $\epsilon = \sqrt{\frac{\log n}{T}}$; n is known beforehand, n is the number of actions available to the players, but T is the number of rounds that the player is going to play that is often not known.

So, the problem with this setting is; the problem with setting $\epsilon = \sqrt{\frac{\log n}{T}}$ is that we need to know the time horizon T in advance, which is not known often. So, the question is can or does there exist an algorithm which does not need to know T , but still can achieve an time averaged external regret of $O\left(\sqrt{\frac{\log n}{T}}\right)$ and that we will see today. That is a clever and often convenient trick.

(Refer Slide Time: 05:18)



So, let me not write it as a theorem. Let me just write it as a remark or because we can use the multiplicative weight algorithm at that setting of a put setting $\epsilon = \sqrt{\frac{\log n}{T}}$ and sort of tweak it to sort of engineer it so that we do not need to know T beforehand.

So, as usual suppose I have n actions, then there exists an algorithm with external regret with time averaged external regret $O\left(\sqrt{\frac{\log n}{T}}\right)$. This part is from before; moreover the algorithm does not need to know capital T a-priori. Proof: so, suppose this is the time

axis and this is time t equal to 1, 2 and so on. Here is capital T the idea is that break the time into epochs break or group the time or iterations into epochs.

The 1st epoch is of length 1, 1st epoch is only time 1; the 2nd epoch is of length 2, it is time iteration 2 and 3; the 3rd epoch is from iteration 4 to 7 of length 4. So, the 1st epoch is of length 1, 2nd epoch is of length 2, 3rd epoch is of length 4 and so on. The length of epochs doubles every time.

So, epochs let us call it ξ_i for i equal to, i equal to say 0 to 1, we will see what will be the value of l . So, ξ_i is the iterations in 2 to the i and minimum of $2^{i+1}-1$ and T . So, except the last epoch it will contain up to T because there is no time after that, but before that the size or the number of iterations in an epoch will keep on doubling. So, that is thing.

(Refer Slide Time: 10:24)

In the beginning of each epoch, we reset the weight vector of the MW algorithm to the all 1's vector.

In the i -th epoch, we use $\epsilon = \epsilon_i = \sqrt{\frac{\ln n}{2^i}}$. Let OPT_i be the pay off of any fixed action in the i -th epoch.

$$OPT - \sum_{i=1}^T r_i \leq \sum_{i=0}^l \left[OPT_i - \sum_{t \in \xi_i} r_t \right]$$

$$\leq \sum_{i=0}^l \left[\epsilon_i \cdot 2^i + \frac{\ln n}{\epsilon_i} \right]$$

And, then what we do is that in the beginning of each epoch. So, ok in each epoch; in the beginning of each epoch, we reset the weight vector of the multiplicative weight algorithm. Recall the multiplicative weight algorithm maintains a weight vector a weight for each of the action and it is initialized to 1.

So, at the beginning of each epoch we initialize, we reset the weight vectors of weight vector of the multiplicative weight algorithm to the all 1's vector. As if we are restarting the multiplicative weight algorithm and we use various epsilon in various epochs. So, in

the i -th epoch we use epsilon equal to epsilon i which is $\sqrt{\frac{\ln n}{2^i}}$ to the i , ok and let OPT_i be the payoff of any fixed action in the i -th epoch, ok.

So, borrowing notation from last proof; so, what is regret? Regret is $OPT - v_i$; v_i is the expected payoff in the i -th iteration of the algorithm is i equal to 1 to T . So, this is less than equal to OPT is less than equal to summation of OPT_i . So, we make this sum epoch by epoch. So $\sum_i^l OPT_i - \sum_{t \in \xi_i} v_i$.

Now, from last analysis we know that this is less than equal to $\sum_{i=0}^l [\epsilon_i 2^i + \frac{\ln n}{\epsilon_i}]$. Now, what value of ϵ_i we have chosen? We have chosen $\epsilon_i = \sqrt{\frac{\ln n}{2^i}}$.

(Refer Slide Time: 15:20)

$$\begin{aligned}
 &= \sqrt{\ln n} \sum_{i=0}^l 2^{\frac{i}{2}+1} \\
 &\leq 2^{\frac{1}{2}+2} \cdot \sqrt{\ln n} \\
 &\leq 4 \sqrt{T \ln n} \\
 \frac{1}{T} \left(OPT - \sum_{i=1}^T v_i \right) &\leq 4 \sqrt{\frac{\ln n}{T}}
 \end{aligned}$$

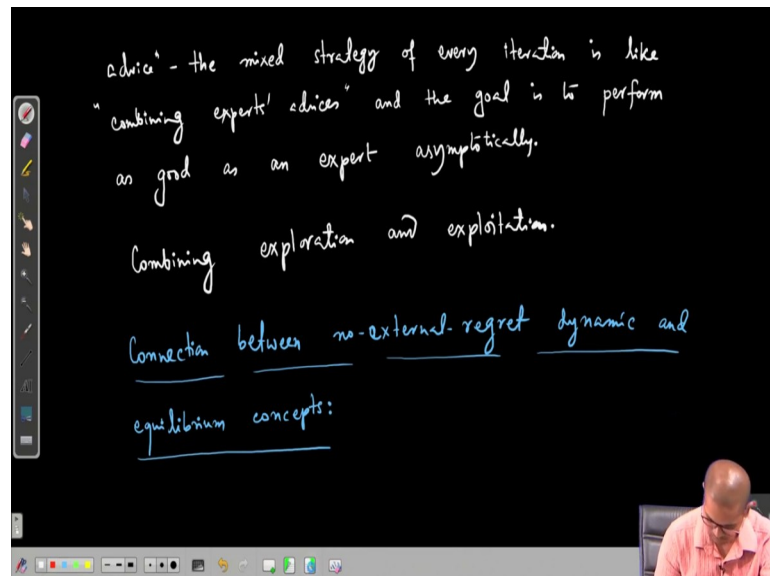
Combining Expert Advice: The problem of designing a no-regret algorithm is also called "combining expert"

So, what we get is this is $= \sqrt{\ln n} \sum_{i=0}^l 2^{\frac{i}{2}+1}$. So, this is a geometric series and geometric series is dominated by the last term. So, this sum is less than twice the last term. So, this is $\leq 2^{\frac{l}{2}+2} \sqrt{\ln n}$. Now, $\leq 4 \sqrt{T \ln n}$.

So, the time average regret $\frac{1}{T}(OPT - \sum_{i=1}^T v_i) \leq 4\sqrt{\frac{\ln n}{T}}$. So, we may not need to know T and we will pay an extra cost of factor of 2, but it does not matter in asymptotic notation it is still $O\left(\sqrt{\frac{\log n}{T}}\right)$ ok.

Now, here is another important perspective into no regret algorithms which is called combining expert advice. So, let me write combining expert advice. You know the problem of designing a no regret algorithm, problem of designing that is a useful perspective is sometimes called combining expert advice is also called combining expert advice.

(Refer Slide Time: 19:07)



Why? Because at every iteration the mix strategy that the player is committing can be thought of as an experts advice and the goal is to perform as good as the best expert asymptotically. So, the mix strategy in every iteration, mix strategy of every iteration is like combining expert advices.

So, you know there are couple of many, there are many experts who are suggesting various strategies and the player needs to combine them in a randomized order and then pick according to the distribution. And the goal is to be as good as the best advice which will be, which will be known only after playing the game. So, the mix strategy is like

combining various experts' advice and the goal is to perform as good as an expert asymptotically.

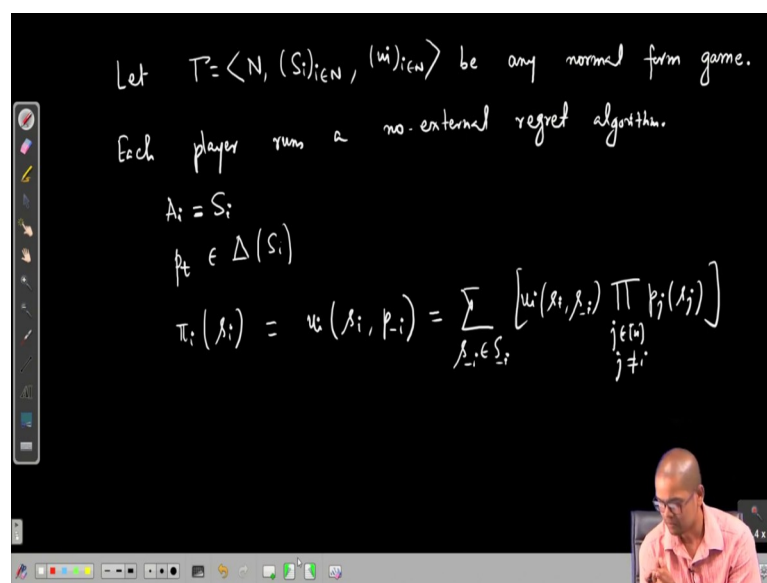
And, that is exactly what the multiplicative weight algorithm is doing. If you look at carefully it increases the weights on the actions which has more utility. So, in the same sense it is giving more weightage to action which has performed good in this iteration or till now and that will in turn will increase its chance of picking that action in the next iteration.

So, it is in the same at the same time it is both excluding various other options. It is for no action it is discarding, it can still pick the action. It is not making the probability of any picking action 0. So, it is keep on exploring, but it is also at the same time exploiting what it has learnt till now. It has learnt which actions are giving more utilities and so on. So, that learning goes into updating the weights accordingly.

So, this is like combining exploration; that means, explore various actions and learn exploration and exploitation, ok. So, very good. Now, we see what is the connection of this no regret dynamics with the equilibrium concepts. So, connection between no regret no external regret dynamics and equilibrium concepts.

So, we will see that this no external regret dynamic is very closely connected with coarse correlated equilibrium. If each player runs a no regret algorithm under in certain situation, then you know they converge altogether to a coarse correlated equilibrium. So, what is the setting?

(Refer Slide Time: 24:44)



So, let as usual $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ be any normal form game. Now, each player runs a no external regret algorithm. Now, when do we say that each player runs a no external regret algorithm, what is the setting? What are the, what are the set of actions available? So, A_i is the set of actions available to player i in its, when it runs the no external regret algorithm and we set A_i to be equal to S_i . The set of actions available to player i is actually the set of strategies.

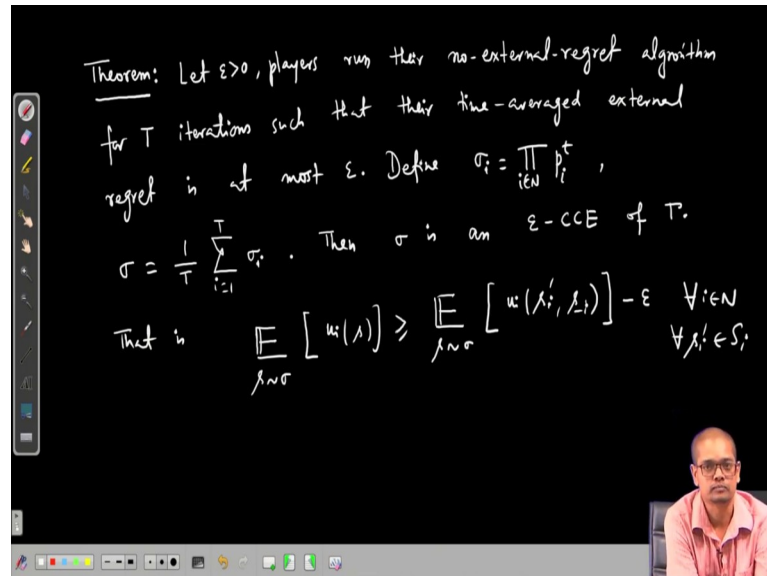
Now, each player will in the t -th iteration will pick will commit to a probability distribution which is over its action set which is same as strategy set and then the adversary is supposed to pick a utility function. And the utility function that is picked by the adversary to the i th player is this $u_i(s_i, s_{-i})$.

Let me write this way, π_i of the payoff of an action s_i is defined as utility of that player i receives when it plays on strategy s_i when other players are playing according to the committed probability distribution p_{-i} , which is $s_{-i} \in S_{-i}, u_i(s_i, s_{-i})$. What is the probability that s_{-i} is played? It is each player p_j play picks $s_j, j \in [n], j \neq i$. So, this is the utility.

This is the utility that the adversary picks and then player plays draw picks a action according to the committed probability distribution and receives the utility. Now, what is the big deal? How this is connected? How each player running their running a no regret

algorithm make them converge to a coarse correlated equilibrium, that is the following theorem.

(Refer Slide Time: 28:26)



Theorem: Let $\epsilon > 0$, players run their no-external-regret algorithm for T iterations such that their time-averaged external regret is at most ϵ . Define $\sigma_i = \prod_{t=1}^T p_i^t$, $\sigma = \frac{1}{T} \sum_{i=1}^T \sigma_i$. Then σ is an ϵ -CCE of Γ .

That is
$$\mathbb{E}_{j \sim \sigma} [u(j)] \geq \mathbb{E}_{j \sim \sigma} [u(j'_i, j_{-i})] - \epsilon \quad \forall i \in N, \forall j'_i \in S_i$$

So, let epsilon greater than 0 and players run their no external regret algorithm for T iterations, ok. T iterations such that they are time averaged external regret after T iterations averaged is at most ϵ .

So, fix and positive fraction ϵ greater than 0 and each player run their no external regret algorithm for T iteration such that after capital T iterations the time average external regret of all the players is at most ϵ , ok. So, then define distribution σ_i which is the product distribution of this probability distributions p_i^t . p_i^t is the probability distribution that player i commits in the t -th iteration $i \in N$, ok.

And, this and then sigma is the average of this σ_i 's. Then, the claim is σ is an ϵ coarse correlated equilibrium of the strategic form game Γ , that is expected utility of player i when it plays according to σ is greater than equal to s sampled from σ and this player i plays some other strategy s'_i and other players are play according to this σ .

And, the utility can increase by at most ϵ . So, I need to supply an ϵ , this should hold for all $i \in N$ for all $s'_i \in S_i$. So, here is a beautiful connection that for pick any epsilon and let the players run their no external regret algorithm till all the players external regret becomes at most epsilon. Then you what you do is that in every iteration you take the

product distribution of their committed probability distribution and take their average, then that average distribution sigma is an epsilon CC.

So, the so, we will see the proof in the next class ok.