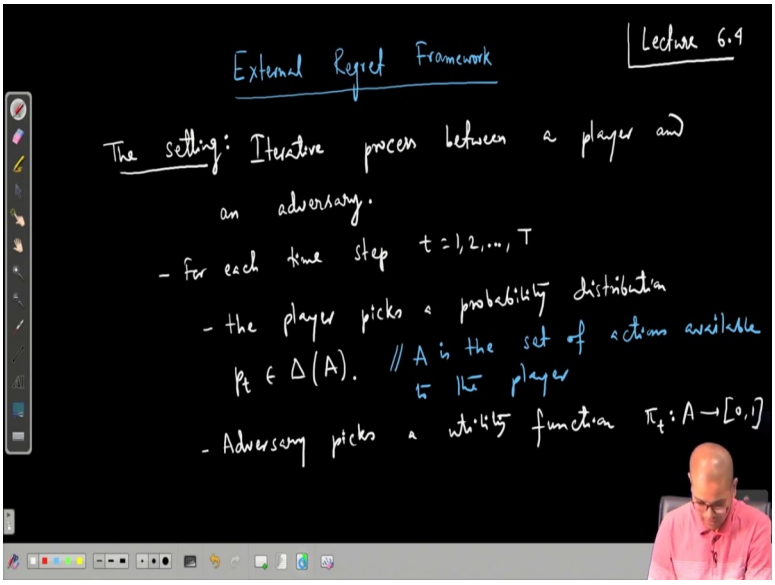
Algorithmic Game Theory
Prof. Palash Dey
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 29
External Regret Framework

Welcome. So, in the last lecture, we have finished our study of equilibrium concepts. We finished with coarse correlated equilibrium, and we left with the question that does there exist natural algorithms or natural dynamics how people behave, so that they can converge to a correlated equilibrium.

(Refer Slide Time: 01:00)



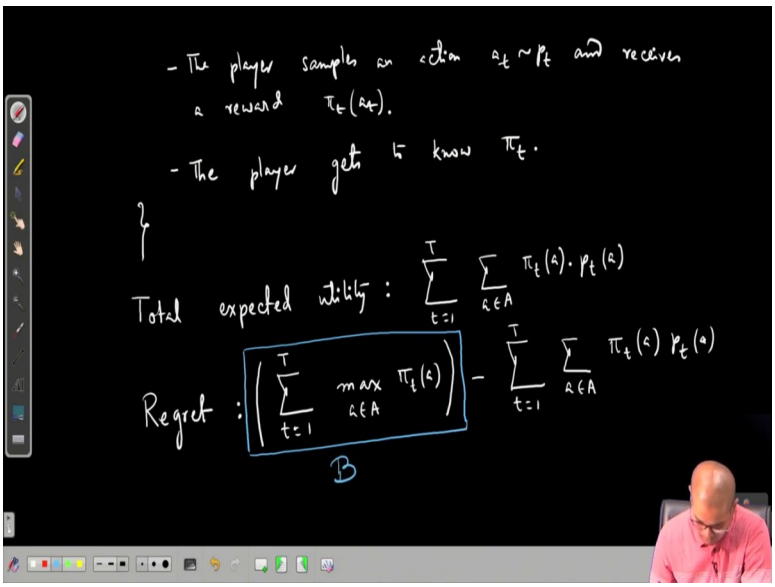
So, towards that let us take a detour and in it will seem like a detour, but we will see how that is related to Algorithmic Game Theory. That is called External Regret Framework.

So, what is the setting? So, the setup, the setting you know it is think of it as a game between a player and an adversary. So, it is a iterative, iterative let me use the term process to distinguish between say the game, the game of the game theory. Iterative process between a player, only one player is there and an adversary.

So, following things happen in every step. Suppose this game runs for T steps, so for each time step small t equal to 1, 2 up to T. So, the player picks a probability distribution p_t over the action sets A. So, let A be the set of actions available. So, let me write A is

the set of actions available to the player, ok. And you know then what does adversary does? Adversary picks a utility function π_t from A to 0, 1, ok.

(Refer Slide Time: 05:04)



And then, what? Then the player samples an action a_t from p_t and receives a reward π_t of a_t. Last step, the player gets to know this entire function π_t. So, it is a iterative process. And in every step at the beginning, in every iteration at the beginning, the player first picks up probability distribution p_t over its actions and then the adversary can see p_t, and then based on that it picks a utility function π_t, which assigns a real number between 0 and 1, this is just for normalization to every action in A.

And then the player samples action from the probability distribution p_t and receives a reward $\pi_t(a_t)$, if the if a t is the sampled action. And then the player gets to know entire π_t. You know this setting is very common in real life, day-to-day real life. For example, think of we want to go from a home to office and there are various paths available. Now, so the various paths available that is the set of actions that I need to choose from and that is my set A.

Then, I pick a path to follow today or it can be one path or it can be a probability distribution over paths, and then suppose it happens that the whatever path I choose that

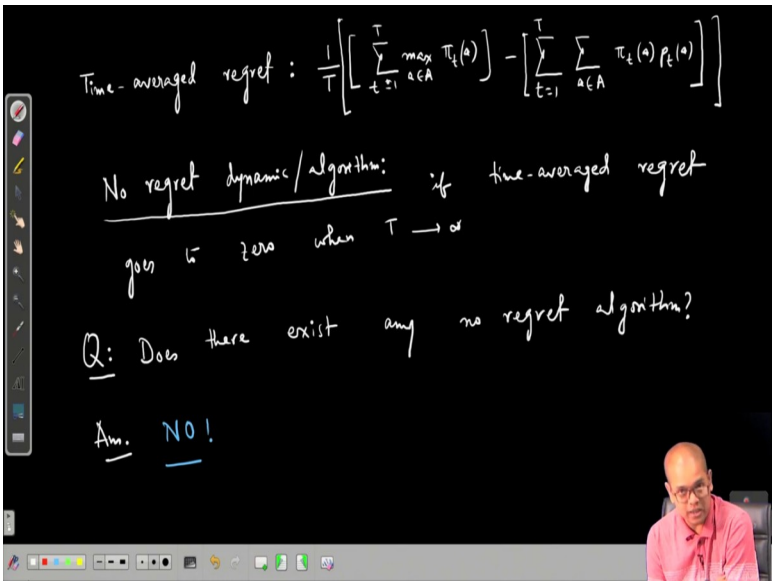
path looks very bad that day. So, it is like I am fighting with an adversary. So, the adversary picks the cost, or not cost the utility.

Utility, the adversary picks and adversary is all powerful omnipotent and omniscient. So, adversary knows my committed probability distribution p_t. And at the time of playing I pick a I sample a probable sample action a_t from p_t and I receive a reward $\pi_t(a_t)$, ok. So, what is my total utility or not; let us not call it total utility total expected utility across all iterations, total expected utility? t equal to 1 to T. What is my utility in the t-th iteration? Summation over $a \in A$, summation over all actions $\pi_t(a)\, p_t(a)$. That is my total utility.

And what is my regret? My regret is that I can daydream and I think that you know at every iteration I could have picked that action which has the highest utility. So, regret that is my regret. Regret is what is the maximum utility that is possible in every iteration $max_{a \in A} \pi_t(a)$. Whichever action has the highest utility, I wish that I could have played that action.

So, that is the best that I can get. And what I am getting is my expected utility, ok. Now, what is my goal? My goal is to have less time average regret. That means, I know that I know the reality and I am happy if I can eventually learn even or I can eventually do as good as the, as good as the base that I can do.

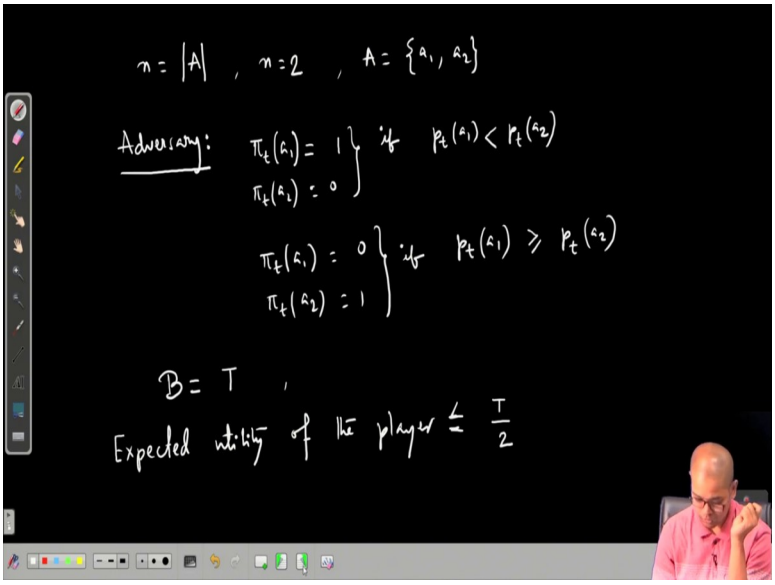
(Refer Slide Time: 11:28)



So, the that is why this notion is called a time averaged regret. What is it? You just divide total regret by time, ok. And an algorithm, a method, a methodology to play this game is called a no regret dynamic if my time average regret is 0. So, what is a no regret algorithm or no regret dynamic? No regret. If time average regret goes to zero, if or when T goes to infinity, ok.

So, question, does there exist any such no regret algorithm; so, any no regret algorithm? And the answer is no. What so? Why no? So, let us see. Let us see an adversary, which will guarantee that this time average regret will not go to 0, as t goes to infinity.

(Refer Slide Time: 14:52)



So, n is the size of action set. So, assume $n=2$, there are 2 actions. So, there is an adversary. What is adversary's job? Adversary's job is to pick the utility function pi t in t iteration cleverly.

And what it knows? It knows the action set A and it also knows the committed probability distribution p_t of the player. So, and it needs to define the utility function π_t of, so suppose A equal to a_1 and a_2, it has only two actions. $\pi_t(a_1)$, it will define it to be 1 if $p_t(a_1) > p_t(a_2)$. It will define $\pi_t(a_1)=1$ and $\pi_t(a_2)=0$.

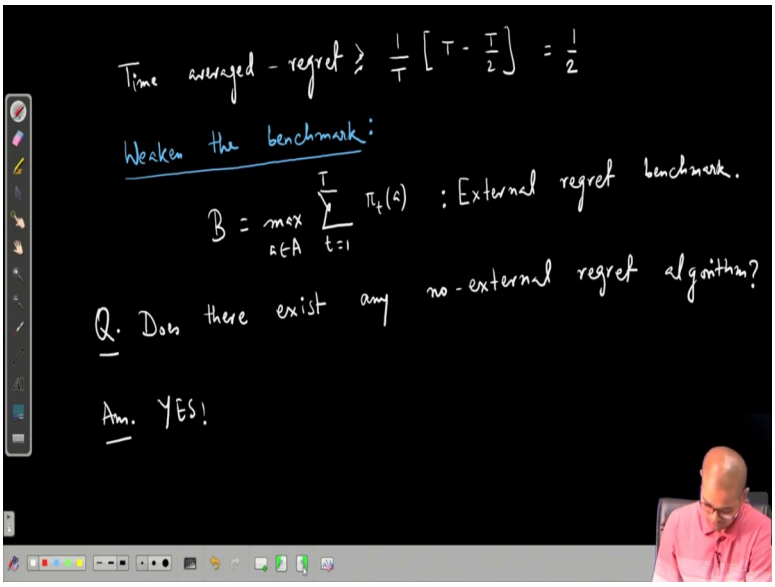
This utility function it will pick if player 1 assigns more probability to a_1, and no this. If player 1 assigns more probability to a_2 that at which is at least 50 percent or more than

50 percent, then the adversary will define the utility of a_2 to be 0 and utility of a_1 to be 1 and the reverse other way. So, $\pi_t(a_1)=0$ and $\pi_t(a_2)=1$, if the other thing happens; that means, if $\pi_t(a_1)\geq\pi_t(a_2)$.

Now, what is the benchmark? This is called benchmark. So, let me see. The thing that we are comparing with this is called benchmark B. So, what is the benchmark? B equal to capital T; because, in every iteration there is an action, which gives utility of 1 and so maximum utility can be what can be obtained is capital T.

And what is the expected utility of player? What is the expected utility? You know this is less than equal to. In every iteration the player plays the action which gives utility 0 with probability at least half. So, it can, it plays the action which gives utility 1 with probability at most half, so in every iteration its expected utility is at most half. And because there are t iterations, its expected utility is at most t by 2.

(Refer Slide Time: 19:26)



So, what is the time average regret? Time average regret is 1 over T; this is greater than equal to $T-\frac{T}{2}$. So, it does not matter how the player will play. This adversary is so strong that it will ensure that the time average regret of the player is at least half, good. So, what is wrong? So, what we do is that we weaken the adversary. So, the idea is

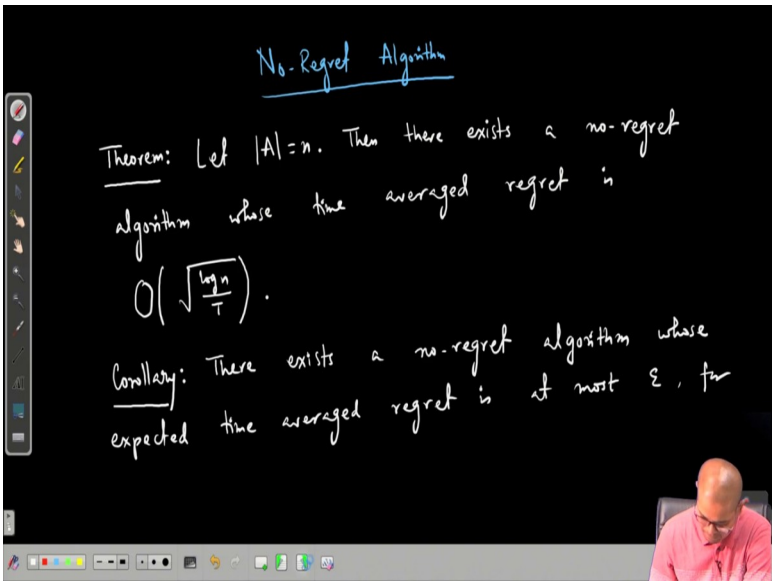
weaken the adversary or weaken the adversary or benchmark, or not adversary, weaken the benchmark, weaken the benchmark not adversary.

Define. So, before we were saying that I want to compare my utility to be as to the condition when I would have picked the best action in every iteration, now I am saying is that, no. So, let us compare me with a player who plays a fixed action, but that across all iterations. But that action can be whatever.

So, let me write. So, if some player is playing action A, then what is its utility? So, $\sum_{t=1}^{T} \pi_t(a)$ is the utility of player if it is playing action B, good. And then I want to pick that action which is the best, ok. So, this particular framework is called external regret benchmark. This particular benchmark is called external regret benchmark, ok.

Question, does there exist any algorithm or any dynamic which achieves no external regret, that means, the time averaged external regret going to 0 as time goes to infinity? Does there exist any no-external regret algorithm? Answer is yes, which we will see just now.

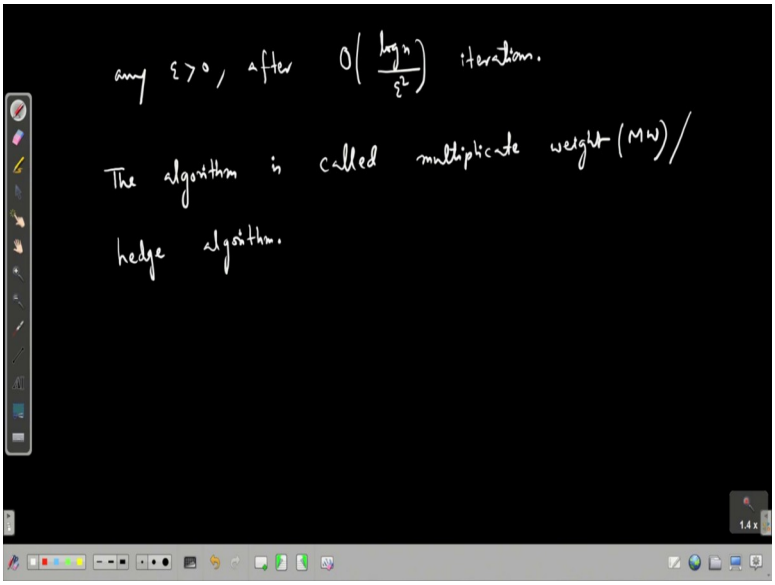
(Refer Slide Time: 23:23)



So, no, this also called no regret algorithm. So, if not mentioned, otherwise it is assumed that no regret means that no external regret. In a subsequent lectures, we will see some other kind of regret that will be different benchmark and we will see. But for that, for the time being let us now focus on no regret algorithm.

So, what is the main theorem? What is the crown jewel? Theorem, let size of A be n, then there exists a no regret algorithm whose time average regret is $O\left(\sqrt{\log \frac{n}{T}}\right)$ ok. So, as you can see that as t goes to ∞, then this external regret goes to 0. So, it is a corollary from this theorem, that there exist a no regret algorithm whose expected time average regret is at most epsilon, of course for any epsilon greater than 0, after $O\left(\log \frac{n}{\epsilon^2}\right)$ iterations, ok.

(Refer Slide Time: 26:39)



And these are very famous algorithms. This is because; this regret framework is very useful. And it has been independently developed not because of game theory and it has been this later we will find this connections. So, this no regret algorithm and this regret framework you will discover in many other areas of computer science and mathematics. So, we are doing this thing now. And at the end we will see what is its connection to game theory.

That is why we started that; why these are, why or the unsatisfaction with equilibrium motions where that you know although the correlated equilibrium and coarse correlated equilibrium were polynomial time solvable they can be computed in polynomial time. You know in many real life human being are the players and they do not solve a linear

program to find what to do. So, they sort of repeat their action every day and learn from there.

So, does there exist such learning framework or which is in some sense natural which led them discover this correlated equilibrium or coarse correlated equilibrium over time. So, the algorithm is called, the algorithm is called multiple, it has many names, multiplicative weight, MW or it is also called hedge algorithm or and it has also some more name. So, we will continue from this in the next class. And in next class, we will see this description of Multiplicative Weight Algorithm this is a very simple algorithm, very powerful, and we will see the proof of this results also, ok.

Thank you.