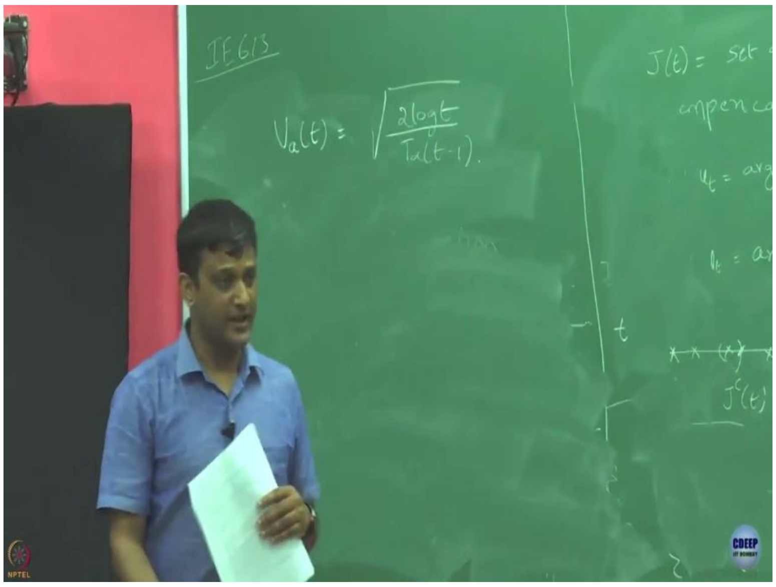
Bandit Algorithm (Online Machine Learning)
Prof. Manjesh Hanawal
Industrial Engineering and Operations Research
Indian Institute of Technology, Bombay

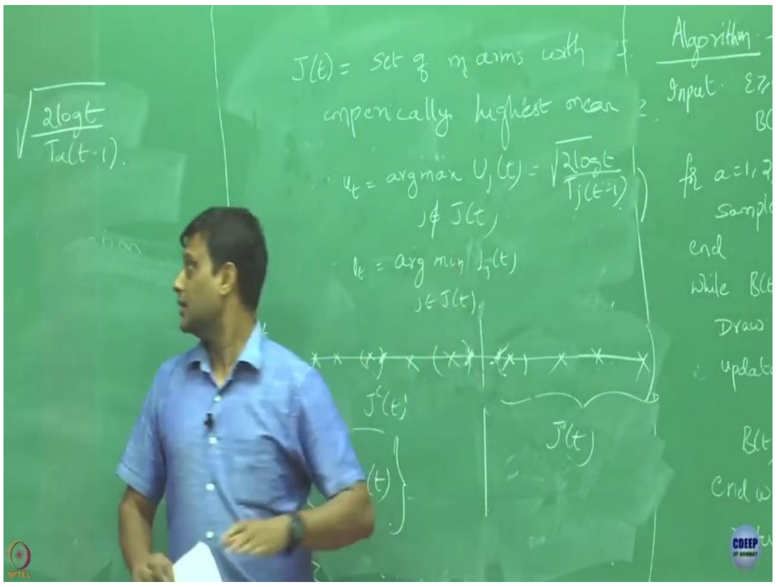
Lecture - 59
Lil' UCB

(Refer Slide Time: 00:20)



Let me define at any time $U_a(t) = \sqrt{\frac{2\log t}{T_\alpha(t-1)}}$,so this is our upper confidence bound, based on by Hoeffding's inequality which we have obtained.

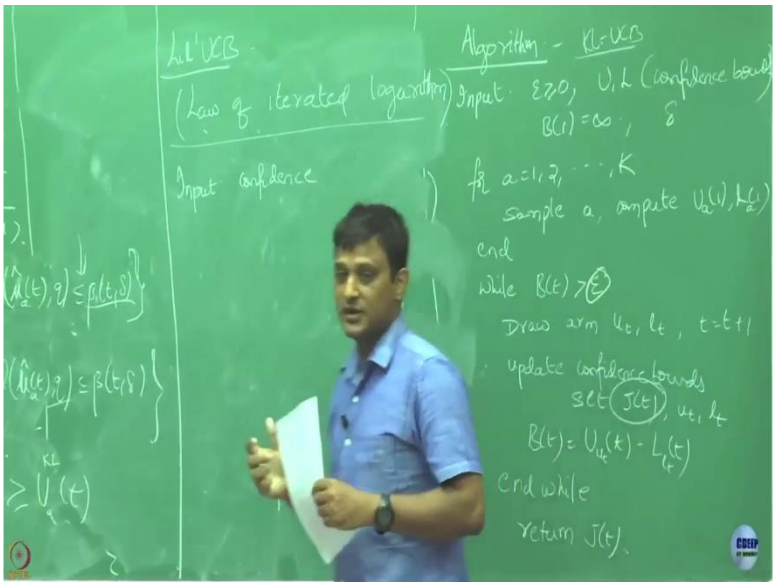
(Refer Slide Time: 00:40)


J(t) = set of m arms with empirically highest mean
$u_t = \arg\max_{j \notin J(t)} U_j(t) = \sqrt{\frac{2\log t}{T_j(t-1)}}$
$l_t = \arg\min_{j \in J(t)} L_j(t)$
$J^c(t)$
$J(t)$
Algorithm
Input

So, this $U_j(t)$ is simply going to be $\sqrt{\frac{2\log t}{T_j(\epsilon-1)}}$. See, when I computed this J(t), it is just based on the empirical means. So, J(t) is; so we after.

Student: Sir.

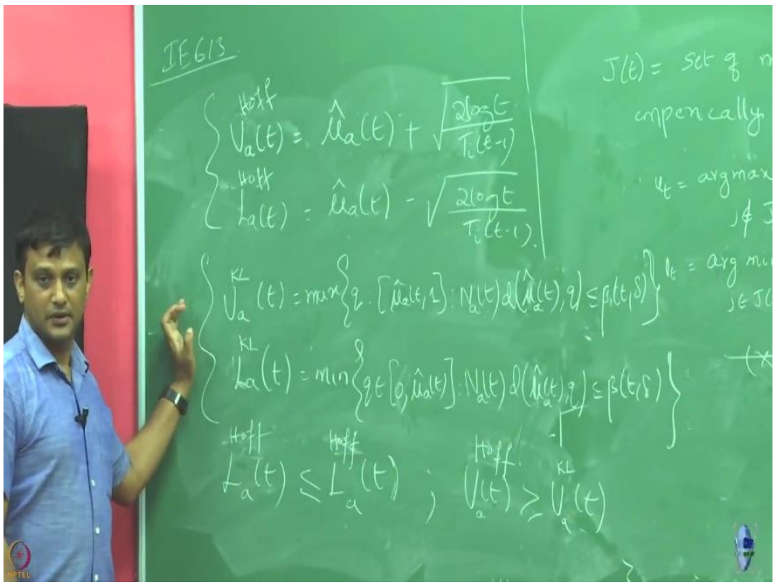
(Refer Slide Time: 01:09)


Lil'UCB
(Law of iterated logarithm)
Input confidence
Algorithm - KL-UCB
Input $\epsilon \geq 0$, U, L (confidence bounds) B(1) = ∞, δ
for a = 1, 2, ..., K
sample a, compute $U_a(1), L_a(1)$
end
while B(t) > ε
Draw arm u_t, l_t, t = t+1
update confidence bounds
set J(t), u_t, l_t
$B(t) = U_{u_t}(t) - L_{l_t}(t)$
end while
return J(t)

The; here I have not yet written right like how I am going to; so when I said set J(t) right, how is this set J(t) is set? It is just in every time t, you will have this i for all i. This is the estimates for each of my arms that is just simply the empirical average that I am going to take, based on that I have taken the first top m arms; these are the remaining arms.

Now, this is what I have used to define my J(t) set and to resolve the conflict between the edge points, I have used my confidence terms; upper and lower confidence terms ok.

(Refer Slide Time: 01:53)



$$\begin{cases} U_a^{Hoff}(t) = \hat{\mu}_a(t) + \sqrt{\dfrac{2\log t}{T_i(t-1)}} \\ L_a^{Hoff}(t) = \hat{\mu}_a(t) - \sqrt{\dfrac{2\log t}{T_i(t-1)}} \end{cases}$$

$$\begin{cases} U_a^{KL}(t) = \max\left\{q \in [\hat{\mu}_a(t), 1] : N_a(t)\, d(\hat{\mu}_a(t), q) \le \beta(t,\delta)\right\} \\ L_a^{KL}(t) = \min\left\{q \in [0, \hat{\mu}_a(t)] : N_a(t)\, d(\hat{\mu}_a(t), q) \le \beta(t,\delta)\right\} \end{cases}$$

$$L_a^{Hoff}(t) \le L_a^{Hoff}(t) \;;\; U_a^{Hoff}(t) \ge U_a^{KL}(t)$$

This is not just this, sorry it is actually the mean value plus the confidence term and the lower confidence term is the mean value minus the same log confidence. So, this is the input to you can use whatever the confidence term you are going to use and then plug in here, then you are going to get this algorithm ok. But, when I said this algorithm is KL-UCB, this algorithm KL-UCB is going to use a confidence terms which are based on KL-divergence, that came from other set of concentration inequality that is large deviation principles.

This confidence terms came from what? Like as I said Hoeffding's inequality, but we said that how good the algorithm depends on how tight this confidence terms are, if the confidence terms are tight maybe I can also get a better performing algorithm. So, other possibility for; let me call this as UCB here, other possibility for this terms are; they are called KL (Refer Time: 03:29) and what is the confidence term?

We used in our KL-UCB algorithm, it looks something like this (Refer Time: 04:45). So, remember this is the confidence terms, this is the indexes. So, so this is about we called

UCB indexes right; indices, we ranked in the UCB, we used only upper confidence bound to rank the all the arms and then played the one with the highest index.

In the KL-UCB , the use this has the index and we ranked the arms based on this value and we play the arm with the highest index. Now, we are just saying; when I am going to define this pure exploration algorithm, we are actually using both this upper and confidence; upper and lower confidence bounds and we are going to either use this pair of upper and confidence bound or we could use this one which are coming from two different methods.

Now, the question is which one is tighter? Is this one is tighter or this one is tighter? So, did viscous said this? Right; so, suppose if these are the tighter ones which confidence; which confidence bounds we like to use? So, naturally we want to use the second one right because if they are going to have a tighter one. So, it can be shown that what we are going to have is this L-UCB.

So, what we are saying is; the upper confidence term, we get from KL-UCB is dominated by that we could have obtained from the UCB; that means, this is the tighter one right and also on the lower confidence bound, we are saying that the lower confidence bound we get from the UCB method is may be instead of UCB, I just call Hoeffding's.

So, this algorithm says the; now algorithm complete. So, now, I am going to in the KL-UCB version of this, we are going to use this confidence terms. So, just I replace; this U and L functions are defined like this, at every time and using that; you are going to compute this U_t and L_t functions. Now, once you are able to compute U_t and L_t function, you can compute this B_t function and then your algorithm completes.

So, as you might have noticed; when you are trying to implement compared your UCB against KL-UCB , you observed this to be better than the UCB, but you are able to compute this sufficiently, this indices. This indices computing it is much easier right; it is just like empirical estimates and then adding this term where as this itself is an optimization problem; we are, all of you are able to compute this fast?

Student: It took time.

It took time ok, but you optimize the time or you just let it run?

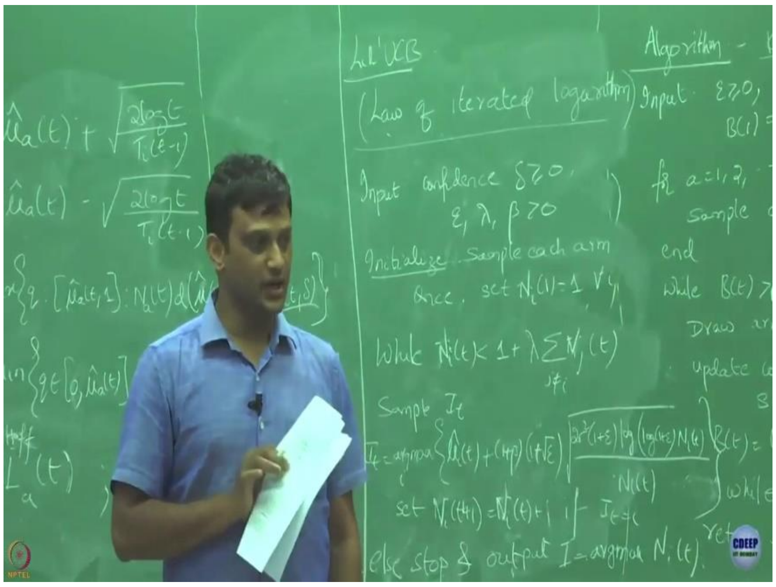
Student: (Refer Time: 09:28).

Ha?

Student: (Refer Time: 09:29).

You did manual search. So, I mean you; did you notice how much time it reduced before you optimize and after optimize if it was taking before 2 hours after optimize how much time it took? Just take few seconds? But, binary search how did you set the precision like the precision was good enough or? Yeah, ϵ you sets some 1 or 0.001. 10 to the power, so, even for that it was converging in few seconds. Ok fine. So, you will face the same difficulty here, but you need to resolve that ok; when you are going to improvement this algorithm fine. So, this is the algorithm we are actually performs good, there will; we are not, I am not going to write now give what is its bound; what is its sample complexity, we will discuss it in the next class; so just like this is an algorithm now. So, we are going to look it another algorithm called Lil UCB.

(Refer Slide Time: 10:37)



Lil'UCB
(Law of iterated logarithm)
Input confidence $\delta \geq 0$, $\epsilon, \lambda, \beta \geq 0$
Initialize: Sample each arm once, set $N_i(1) = 1$ $\forall i$
While $N_i(t) < 1 + \lambda \sum_{j \neq i} N_j(t)$
Sample I_t
set $N_i(t+1) = N_i(t) + 1$ if $I_t = i$
else stop & output $I = \arg\max N_i(t)$
Algorithm
Input $\epsilon \geq 0$,
for $a = 1, 2,$
Sample
end
While $B(t) >$
Draw
update
$\hat{\mu}_a(t) + \sqrt{\frac{2\log t}{T_i(t-1)}}$
$\hat{\mu}_a(t) - \sqrt{\frac{2\log t}{T_i(t-1)}}$

So, Lil stands for here Law of iterated logarithm; so I am just going to discuss this algorithm again and we will look; we will not discuss much about its sample complexity ok. I just want to, but I am have you asked you to compare this algorithm with KL-UCB algorithm. Then, you see that both of have its own issues; then we have worked the inputs are needed for; what are the tuning parameters in the KL-UCB algorithms, what all the things you need to tune? ϵ is the one thing you need to tune. There is also; so, if I am going to use KL-UCB , there is also this β sequence right, this is another thing I need to tune. So, how did you tune this β in the KL-UCB algorithm before? It was some.

Student: (Refer Time: 12:04).

You just make it 0, that is; so you have to see like that is one tuning parameter you have to set. Had you not set it to 0, but you set it to some other value. So, there was I think; if you are right, it was log(t) +3 log(log (t)) right? Did you any of you try if instead of setting that constant to be 0, if I just set it 1, did it improve or worsen the situation? You should try this time ok. See that if I have just going to; so, whoever you gets the best performance, so do mention what is the value you are going to get for see there. I mean when you submit the report; just mention what is that value you are going to see ok; so will see. So, you have to set this parameter ϵ with that is without and you have to set this ϵ without knowing it right. So, this is one hidden in this algorithm or how to set this appropriately, this is the tuning parameter.

And another thing is we know that setting that constant to be 0 works better, but it is not necessary that every time 0 setting constant is a better, may be some in some cases setting it to be some positive value may work out better. And U and L; these are anyway, but the main tuning parameters are ϵ and this β. So, see that in your when you implement that, if you play with this parameter; see if you are going to get a better sample complexity and report that. Like, if I am going to; if I when I set this ϵ, it worked best for this problem setting.

So keeping that in mind; let us see this algorithm, what all things that it this algorithm needs to be tuned? What kind of algorithm we like? The one with the minimum tuning parameters or with lot of tuning parameters?

We want it to be the one with minimum right like; that means, your intervention in the algorithm is minimal and these are all supposed to be learning or artificially intelligent algorithm. So, you your interference should be smaller and they should be; if at all there are parameter, they should be automatically tuning themselves instead of you manually interfering ok. This algorithm, it is going it is going take input as; so whether they; where is the input? Like, I say these are all for a given δ parameters right where did the δ come into picture here?

So, the δ is observed in this β functions right; how, was the δ influencing this β functions ok? So, so just when you are going to implement check that; how this δ, where you are going to use this δ? So, if you are going to increase your δ; what should happen? Increasing δ means what you are able to tolerate more errors right; that means, it should; your algorithm should terminate fast. So, play with that and see I think in your question, we might have fixed that δ to be something. If you are fixed fine, but if you want yourself check whether if I am going to change δ anything varies in the algorithm, you can just try out that as well.

Confidence δ and parameters ϵ and λ is going to take this two; now initialize. So, the way I defined this KL-UCB algorithm; it is trying to identify the top m arms, but you can just set m = 1, in that case it is trying to I just identify the best arm. So, this algorithm I am writing it here; it is just trying to identify the best arm ok, not the top m arm. So, it is just it focus is to identify just the best arm.

So, what this algorithm is doing is; it initially samples everybody once and then this is the stopping criteria for this algorithm; so, the like here the stopping criteria was whether the B_t is going to be ϵ or not? So, here it is going to check; s, one this how this guy condition is going to be violated, if there is; if for some N_i happens to be larger than the sum total number of pulls of all the others. Whenever that has happened then this criteria is violated and then it is going to stop, otherwise it is going to continue ok.

So, when it will continues; it is going to always play the arm which is again this algorithm defines some UCB index, that is coming from some newer concentration bound that is called law of iterated logarithm, not going to that; this is another concentration bounds like based on Hoeffding's and what are KL-UCB . So, this they are going to define UCB index like that and what volume of they are going to play I_t, they are going to update the count of that; for others no count is updated.

So, here it should be N_i and then once it so happens that number of pulls of a particular arm happens to be larger than the sum of pulls of all the other arms, then this guy is going to stop. And then it is going to output the one which has been pulled in maximum number of times at that point as the best arm.

So, it must be the intuitively it makes stopping criteria makes side; if it so happens one guy has scored so much that it is larger than the sum of rest of the guy score in the class may be that guy is best. So, it is just going to use that algorithm, but when they do; some how they are going to they exploit this law of iterated algorithm and show that, this algorithm stops with finite time and it has a good sample complexity.

Especially, I think their result the sample complexity result holds for two arms, but you can check how good it is; compared to the KL-UCB algorithms ok. So, now there are certain parameters again here to be tuned; what are the things to be tuned?

Student: (Refer Time: 22:36) that is the σ?

so another thing they are going to assume that my distributions are σ-sub Gaussian, that parameter is that σ ok. But they are also assume that even though it is sub Gaussian, but the means are all in the interval [0, 1); the support could be large, but the means are restricted to the interval [0, 1) fine ok. So, this is for a sub Gaussian distribution with parameter σ square; so everything is defined right ok? Let us see this.

Let us focus on this λ; suppose if I increase λ, do you think its sample this stopping criteria is going to met early or late?

Student: Late.

It is going to met right like I want the best guy to be much much better than the rest all put together, but if I this λ happens to be smaller, may be it will come down; so then again this becomes a tuning parameter. I mean I cannot like; a priorily decide what is the best λ here right. So, you have to maybe just place it this also.

They say; one you are going to say, this guy is going to be superior to this or the other way around; see that what are the parameters you need to tune and see that by tuning them. Suppose let us say this guy beat this guy, now this guy is poor champion right; let us try to give him some advantage, try to tune any of his parameter and see that this guy overtakes this guy ok.

So, just like do this I mean it may not be, it may happen that one guy beats the other guy too much amount, but if you see that they are very close. So, that there is not much difference between this guy happens to beat this guy, this sample complexity turned out to be just 50 here and for this guy turned out to be just 60 and try to tune this parameter and see it can come down by 50 and just some simple manipulations that is just for the experimental purposes ok.

Now, the two algorithms I said these are kind of already old algorithm the one the KL-UCB , I said that come in 2013 and this Lil UCB, it came in 2014. Subsequently, people have developed many algorithms ok. So, if any of your interested to study this pure exploration, you can take this as a project topic.

Earlier, this was not listed or may be like you are not aware of this topic right. So, because of that if anybody has not taken, but you want to look into this, you can still take it. I do not recall all the names, but I think there are quite a good few more algorithms that have come which have also some better theoretical guarantees and also; I do not know the empirical.

Empirical, there are debates; some people say it is hard to beat KL-UCB says thus most of the time, but I remember like somebody said there is one more algorithm it has come recently which does sometimes better than these may be that is open for explorations. So, if you; one of you are much interested, you can do all the comparisons of these algorithms and may be write a nice report of comparison and doing it a regret job of doing comparing it under different different environment setting.

So, ok; so let us stop here.