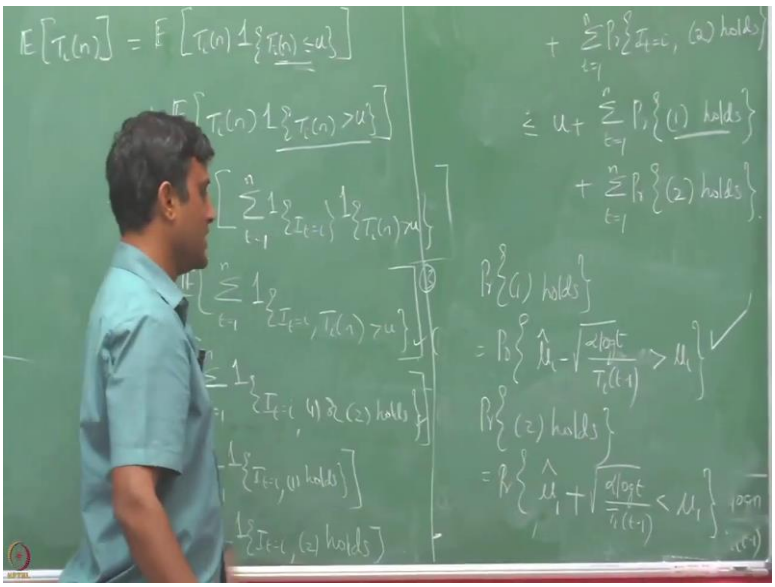
Bandit Algorithm (Online Machine Learning)
Prof. Manjesh Hanawal
Industrial Engineering and Operations Research
Indian Institute of Technology, Bombay

Lecture - 35
Problem Dependent and Independent Bounds of UCB

(Refer Slide Time: 00:21)



(Refer Slide Time: 00:24)

So, let us try to bound this term. Here we have probability mu i hat minus alpha log t divided by Ti of t minus 1 that being greater than mu i. So, just reorganize this in a

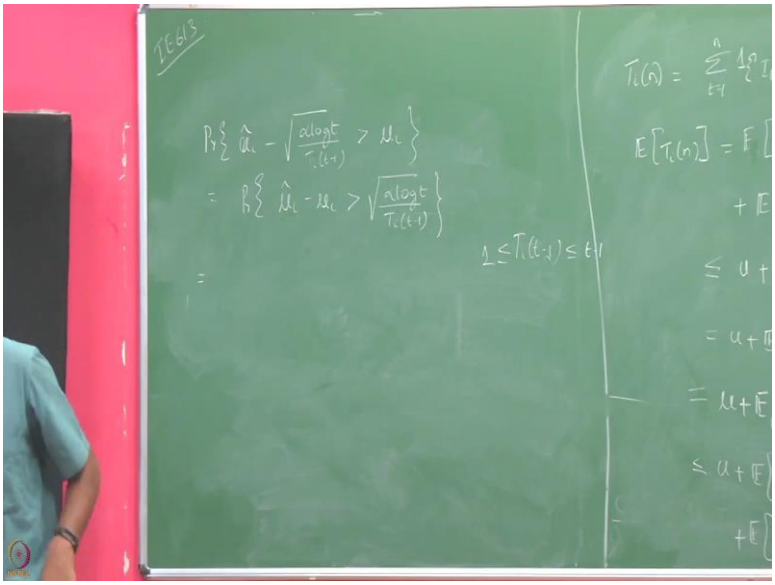
format which is familiar to us, want this to be. So, before we apply our concentration inequality on that, we have to be bit careful.

So, what we know about our if X is a sub Gaussian noise, so if you have a n samples of the sub Gaussian noise or a samples from a sub Gaussian distribution. And we want to know how is this quantity how is how is this and a probability, we know the bound is like exponential, if my distributions are all sub Gaussian with sub Gaussian parameter 1, then these are and this based on the n samples, we know this is the bound right which we have already shown.

So, when we applied this concentration bound here, we knew that there are exactly n samples that are you have been used to estimate this parameter mu i. The estimate mu i hat is based on n samples and then, we had in this bound here ok. But I cannot treat in this case this to be an alpha in this case and apply a bound like this here. Why is that? Because this number of rounds you are right, this mu i hat is estimated based on this Ti number of rounds, sorry number of samples and this Ti number of samples still round t minus 1 that is a random quantity, it is not like a fixed one.

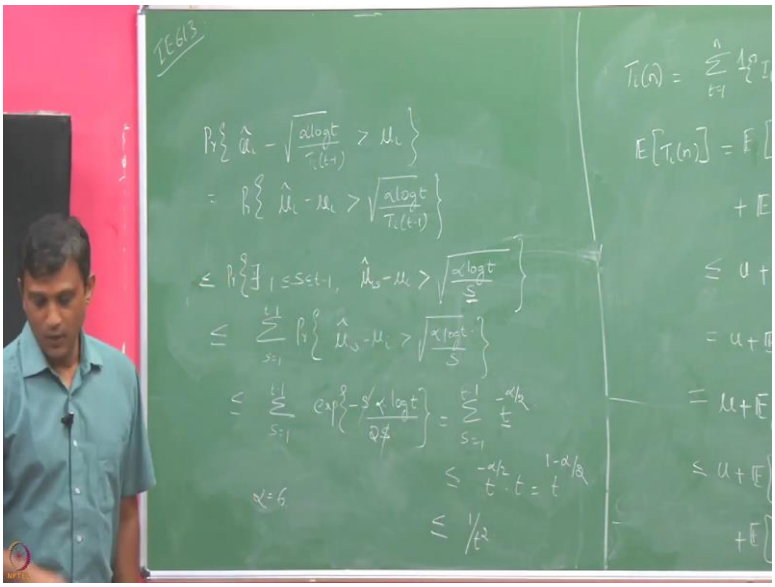
So, I have to take into account this randomness in T i before I can go and apply this concentration bound here. So, how to do that? How to get rid of the randomness in this ok? So, one possibility is that you take all possible values that T i would can take and then, use this bound on those specific values of numbers ok.

(Refer Slide Time: 03:16)



So, what are the possible values of Ti? We know that it has to be something between 1 to t minus 1 right because either I would have played 1 round at most or I would have at least or I would have played the same arm for all the slots till t minus 1.

(Refer Slide Time: 03:42)



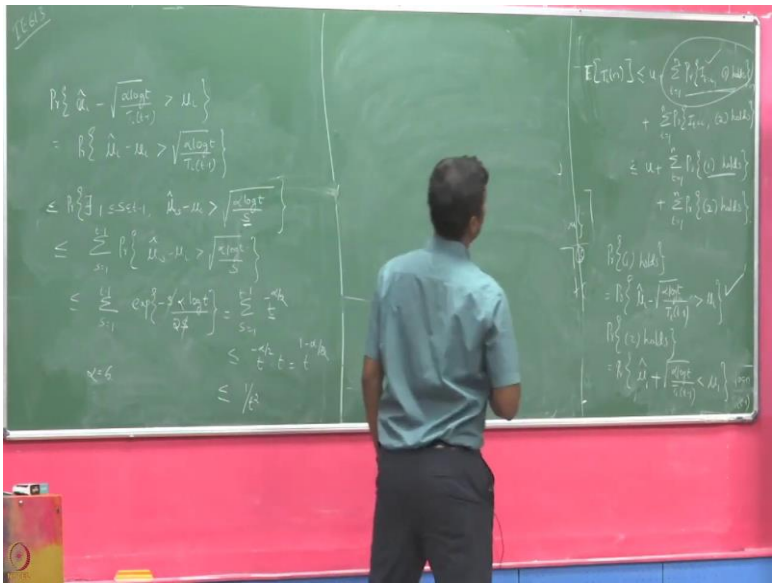
Now, taking that into account, this bound we are going to write it as there exist s between 1 to t minus 1 such that mu i s minus mu i is greater than alpha log t divided by s here. Now, recall the notation that I introduced at the beginning of the class. What this mu i hat is means? This is that my the estimate for i-th arm is obtained exactly using s samples and that is why this s is also coming here in the denominator.

Now, given that s is a deterministic quantity here, then I know how to apply my concentration bound on this. But before that I need to deal with this like, this I have to since I am dealing it with all possible values of t s, I have this. So, I am further going to apply union bound on this to get right and now, this is somewhat we know how to deal with this and this quantity is upper bounded as s equals to 1 to t minus 1. What is this? This is exponential and these estimates are based on s samples.

So, this is s times and epsilon is this quantity here. So, I am going to take alpha log t by s and there is also denominator s here and this will give me s equals to 1 to t minus 1 after knocking of s. So, this will give me after simplifying, I will get t to the power minus alpha by 2 ok. This I could further simplify it as; so I am adding, so this s, this is the t term here and the running variable here is s 1 to t minus 1.

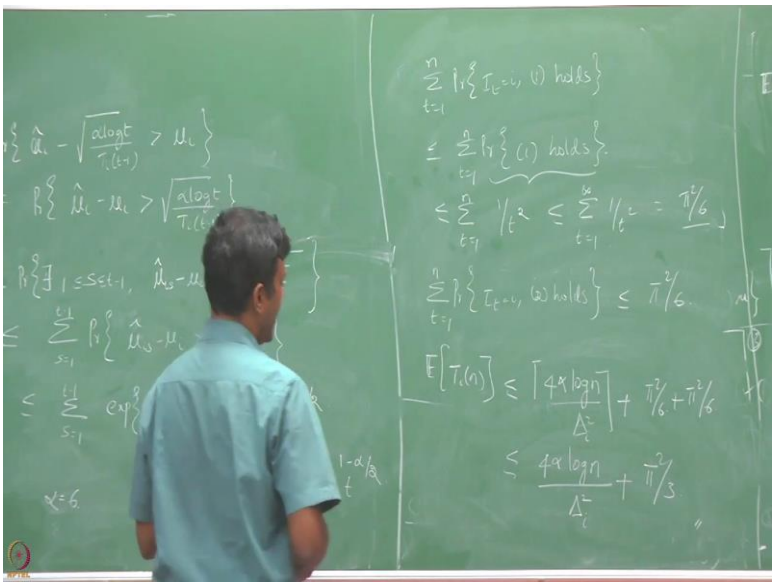
So, this term get added t minus 1 times. So, I will just make it this add this term for t number of rounds and I will get t to the power minus alpha time into t. This will give me t to the power 1 minus alpha by 2 ok. Now, I am going to choose specifically alpha to be equals to let us say 6 ok. Now, if I choose this alpha to be 6. This bound I am going to get is like 1 upon. So, alpha 1 by 3 and I am going to get it as t equals to 6. Now, what I am interested in? I am interested in now computing this probability.

(Refer Slide Time: 07:12)



Now, I have actually computed this probability, I have bound on this probability you know through this and now, I want to compute its value over t running from summing it over t from 1 to n.

(Refer Slide Time: 07:40)



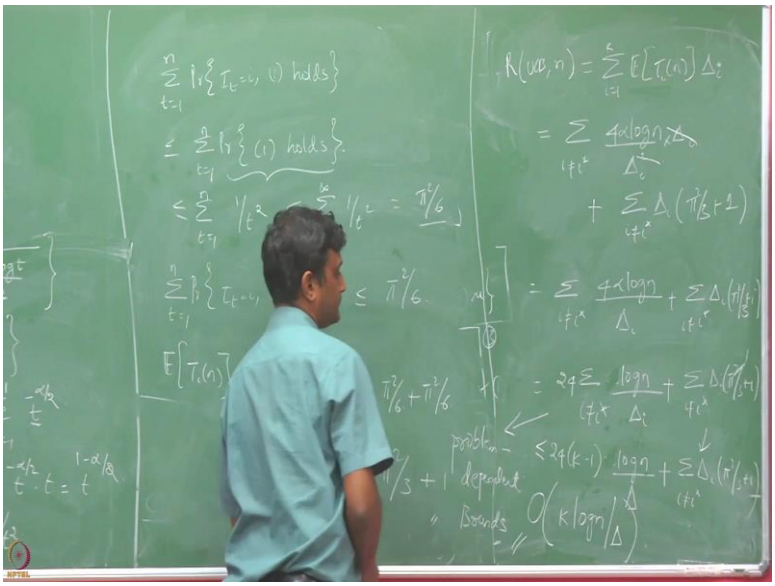
So, let us do that now. So, we have that t equals to 1 to n probability that I_t equals to i and 1 holds we have upper bounded as t equals to 1 probability that 1 holds and this one we have already shown that t equals to 1 to n and this quantity is we have shown to be exactly upper bounded by 1 upon t square ok.

Now so, this series here when it is summated from t equals to 1 to n, we do not have a closed form expression. But when we can bound it? So, when we bound it, let us say by letting t equals to 1 to infinity, we know value for this series and that is pi square by 6 ok. So, similarly you can verify this term here, the second term here by going through exactly the same step, you can verify that this is can also be upper bounded similarly as pi square by 6 ok.

So, now, putting all these things in my bound on the expected number of pulls of arm i, I will get expected number of pulls of i is what is u? Let me put it back the value of u, I have used. The u, I have use to be this quantity plus the first term here ended up yielded me pi square by 6 and the second term also added a similar term. So, if I just simplify this, this gave me 4 alpha log n plus pi square by 3.

So, finally, what we have shown here is the expected pulls of the sub optimal arm i is bounded by this quantity here, that is 4 alpha log n by del i square plus pi square 3. Now, we are almost done like once we have this, we know already how to get our regret bound right. What is our expected regret or rather pseudo regret of policy UCB or n rounds?

(Refer Slide Time: 11:16)



We have denoted it, we know that this is going to be expected number of pulls of arm i times delta i and this summation is over all i equals to 1 to k. Now, I think, I miss. So, when I remove this hill maybe I can add 1 here and this bound still holds. Now, plugging back the value of this from this, I will now take anyway I know that del i is 0 for the optimal arm, I will just skip that part. So, taking all the arms which are not optimal, this bound holds and this bound is saying that this is like 4 alpha log n divided by delta i square into delta i plus summation i not equals to i times delta i into pi square by 3 plus 1.

And just a simplification will give you this quantity here and recall that, we got all of this by setting alpha equals to 6 in this fashion. So, if you plug it back here, the bound I have here is finally, 24 times i not equals to i star alpha log n by delta plus this quantity here. And further, if you want to further replace sub optimality gaps of each arms by this is the sub optimality gap, we have defined earlier, this problem can be further bounded as this is going to be 24 times k minus 1 alpha log n by delta plus this quantity.

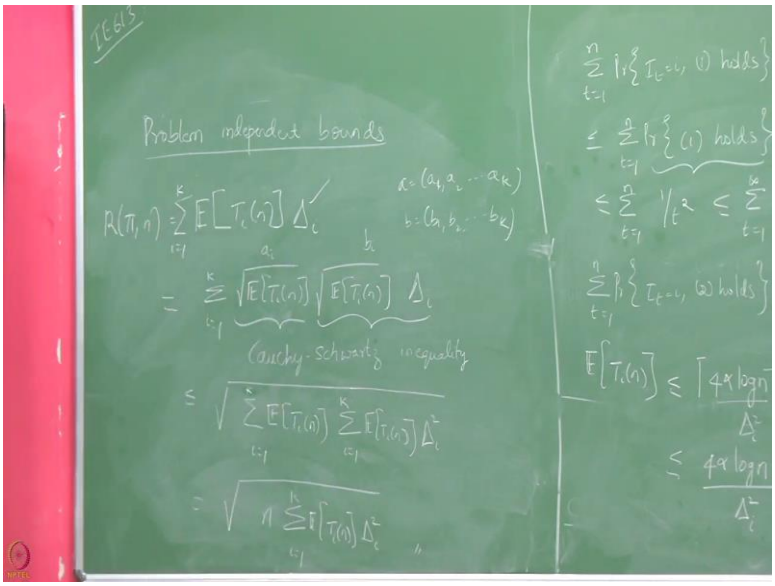
And if you look in this problem in terms of n, this problem is sorry I do not have alpha here, I have plug plugged in 6 for alpha. So, this problem like grows like logarithmic in n and in terms of number of arms, it is like grows like k minus 1 or like almost like linearly in K.

So, order wise if you ignore this term here which is usually small because this sub optimality's gaps could be I do not know whatever be the sub optimal gaps; at least it is not growing, it is like a constant term here right that depends on your problem. And if I write order wise in terms of my k and number of rounds, this problem I can write it as this is like order k times log n by delta. This we have already discussed when we introduce this algorithm.

So, finally, we ended up showing that my regret of UCB algorithm is order k log n by delta or more precisely, it is given like this and it is clear that my UCB gives sublinear regret. Now, here the problem the regret bound, we have it depends on the specific instant of the problem right. So, recall that we said that for a when we say a problem instance is fixed that is the mean values associated with the distributions are fixed ok. If that is the case, once the mean values are fixed, the associated gaps are fixed and this bound is in terms of this gaps and such bounds we call it as problem dependent ground or instant dependent bound.

So, once you fix a problem here that is your bandit instance, this del i's are fixed and you are expressing it in terms of those values. Now, you may be interested in now knowing ok. This bound is logarithmic in n, when it the bound is expressed in terms of the problem specific constants. What if I do not know what is the underlying problem instance and I want to get a bound which is holds uniformly across all the problem instances ok, such bounds we are going to call it as problem independent bounds.

(Refer Slide Time: 17:39)



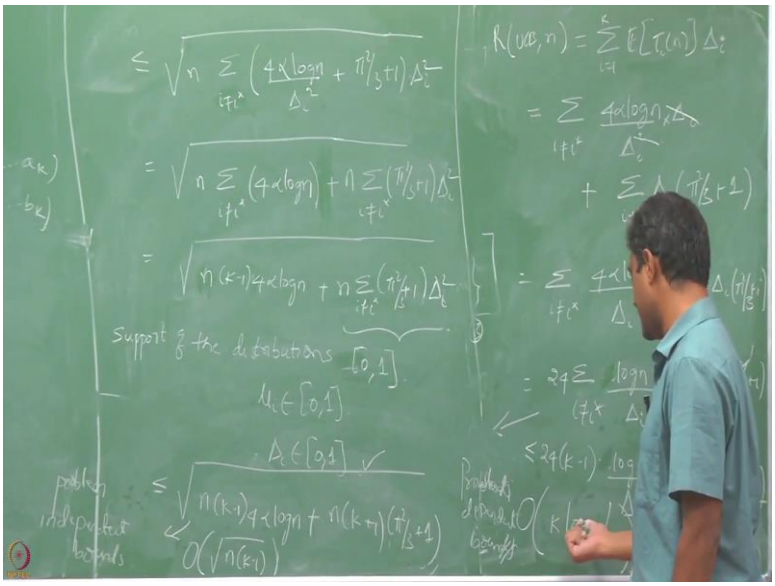
Now, how to get a problem independent bound? That is a bound which holds irrespective of what is your problem instances, So, that we can get by the same analysis, but bit exploiting our regret decomposition theorem a bit better. So, what is the regret decomposition result we have? We have that for any policy pi and n, we know that the regret can be defined as where expectation of Ti n is the expected number of pulls of i thumb right. Now, can we write this bound, use this bound to get a bound which does not depend on this delta is? That is the problem instance is ok.

So, let us see how we can do that. So, this expected term here, we are going to write it as Ti n and expected value of Ti n. This is just like rearrangement nothing changes here and once I do this, I am going to apply Cauchy Schwarz inequality on this. By treating this quantity as let us say b_i and by treating this quantity as a_i. So, we and treating this a_i as the i-th component of a vector and that vector is of dimension K. So, treating this a to be a_i and b to be b 1, b 2 up to b k.

So, this is like now where a_i is defined like this and b_i defined like this. So, this is a inner product between these two quantities and by Schwarz inequality, we know that this is upper bounded by sum of square of this term right. So, because of that the first term is going to give me Ti n, i equals to 1 to n and the second term will give me i equals to 1 to sorry this should be k; here i equals to 1 to k expectation of Ti n times del i square ok.

Now, I know that the expected number of pulls of arms i when I summed over all arms, it has to be equals to n right and what about this term? Let us let me keep it like that for time being now. Yeah. Now, we have already the bound on expected value of Ti n let us plug in that here.

(Refer Slide Time: 22:23)



So, once we do that, our bound is to be like n what is the value of goes to K. So, I will again since for the optimal arm the delta is going to be 0, I will only consider i not equals to i star and for this, we know the bound is like 4 alpha log n by delta i square plus pi square divided by 3 plus 1 multiplied by delta i square ok. So, now, if you simplify this, we are going to get n i not equals to i star. This gets knocked off here 4 and log alpha and then, we will have this another term n times i not equals to i star pi square by 3 delta i square ok.

So, now, further simplifying it, this will give me 4 k minus 1, sorry n 4 alpha log n plus n times this I will keep it just like that ok. So, now, notice that this delta i square got knocked off with this delta i square term here and we are left with del is only in this part of the bound here ok.

Now, we are assuming that my distributions are all sub Gaussian right, after centralizing them and each one of them, without centralizing them, they have their own mean and they have some associated sub some sub optimality gaps here and because of my sub

Gaussian distributed with the non-centred version of this can any mean value which could be between 0 to infinity. This del is can also be any values any real numbers.

But however, suppose, we assume my problem class is such that all the distributed all the distributions are such that their support takes, their support is over some fixed interval; let us say the support of the distribution ok. So, because of this, if this is the case, then all mu i's are also in the interval 0, 1 and so, are delta is ok. So, in this case, we could further upper bound these quantities by replacing this delta i squares by 1 and in we will get a bound which is n times k minus 1 times pi square by 3 plus 1.

So, now, for the class of all distributions, for the class of my bandit instances, where my distributions have this support 0, 1, now we have this bound which does not depend on what is a particular problem instance from this class right and it this upper bound only depends on my number of arms this alpha whichever I chose in my algorithm and the number of rounds for which I run. So, such bounds, we are going to call it as problem independent bounds that is if we can give our bound which does not depend on which particular instance of the problem, we are talking about.

The bound count, so just to be clear contrast, this bound here with the bound I have got here for a special case of distribution with bounded support. Here my bounds did depend on my specific problem instance. The problem instance are coming through here delta i's and also or delta here; whereas, these deltas are not here. Now, in this case, in this problem dependent bound, i, we got a regret bound which is of the order k log n; whereas, in this version of problem independent bound this regret is of order square root n k minus 1.

So, this is the main difference between problem independent and problem dependent bound in the problem dependent bound the problem specific problem instance comes in the picture. And usually, we get a regret bound which is of the order k log n; whereas, in the proper independent bound, we will get regret bounds which is of the order square root n times k minus 1 ok. So, both of them are sublinear. In this case, this goes very fast when you divide by n, it goes very it decays very fast to 0, but this one bit decays slowly.

So, that is obvious because your this bound holds irrespective of what is your problem instance; whereas, this bound does not depend this, this depends on the specific problem

instance; whereas, this does not depend on which problem instance. So, this holds uniformly across all bandit instance ok.

So, we will stop here. In the next class, we are going to see what are the bounds we have; are they really optimal; is UCB algorithm is really optimal or we should be thinking some better algorithms ok. We will stop here.