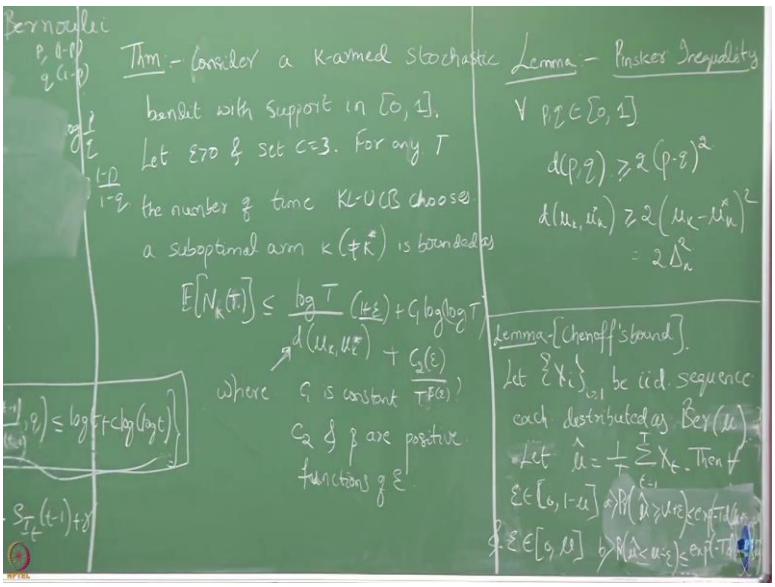
Bandit Algorithm (Online Machine Learning)

Prof. Manjesh Hanawal
Industrial Engineering and Operations Research
Indian Institute of Technology, Bombay

Lecture – 37
Thompson Sampling – Brief Discussion

(Refer Slide Time: 00:22)

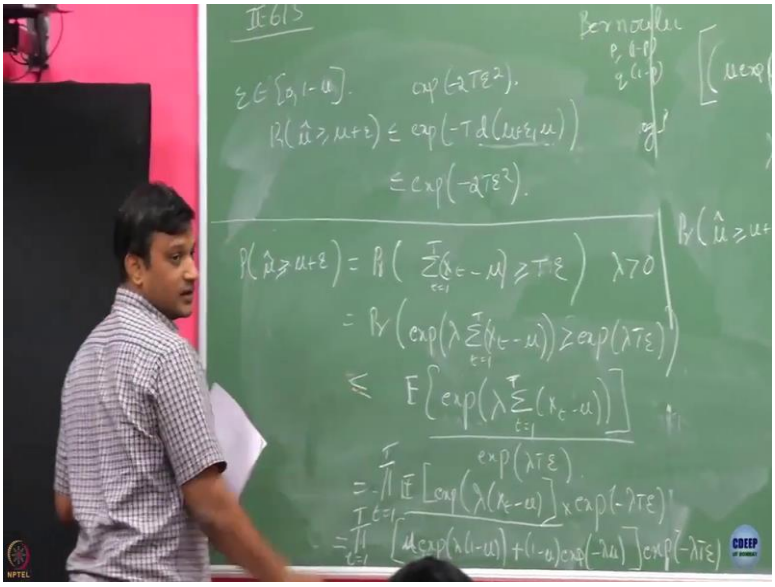


So, I am just going to state this lemma and this is called Chernoff's bound ok. So, this let me state this. So, it is just saying that let be X_i be a sequence iid random variables where each one is Bernoulli distributed with parameter mu and let mu hat denote the estimates using T samples ok. Then we are going to say that then take any epsilon which is between 0 to 1 minus mu, then it is saying that the estimate the sample mean mu hat being larger than mu plus epsilon is upper bounded by minus T divergence between mu plus epsilon and mu.

So, now just let mu let us say what is my parameter epsilon range 0 to 1 minus mu. So, when epsilon is 0 this guy is simply mu and when epsilon is largest like 1 minus mu this guy is going to take value 0 ok. So, this mu plus epsilon is never going to be larger than mu here in this case and similarly, when epsilon is between 0 and mu we can argue that the mu hat being less than or equals to mu plus epsilon is exponential minus T, again the divergence between mu plus epsilon and mu.

We will not go yeah show. So, yeah it should be mu minus epsilon and also this is mu minus epsilon maybe that is I do not know it is all clearly visible.

(Refer Slide Time: 03:24)



Let me just write it again here. So, what we are just saying is for epsilon between 0 to 1 minus mu yeah just saying that 0 to 1 minus mu I will just saying the probability that mu hat greater than mu equals to epsilon is upper bounded by exp minus T divergence of mu plus epsilon and mu ok. First thing is this bound this concentration bound we have is it any better than what we had then using my Hoeffding's inequality. What was the bound we had in Hoeffding's inequality? So, this bond we have exp minus T.

Student: 2T (Refer Time: 04:08).

So, now if you are just going to apply the Pinsker inequality, what we have on this divergence this divergence is lower bounded by?

Student: 2.

2 delta k in this case is going to be epsilon right mu and mu plus epsilon. So, this is going to be I have a lower bound on this, but this is with a minus sign here. So, I am going to get. So, you see that like the if you have if I can restrict myself to Bernoulli rewards, then I have a deviation bound which is slightly better than what I would have got using my Hoeffding's inequality.

So, the KL-UCB exactly exploit this and that is why it is going it is able to give all the bounds in terms of the divergence and this than they happen to be tighter than what we for the ucb bounds.

So, let us quickly go through the proof of this one. This is it will be a good exercise like for us to just a repeat some part what we have already done in the last class not last class the class before that. So, what we have? We have probability that mu hat I think greater than or equals to mu plus epsilon. So, what I will do is I will just unravel this quantity and this is like X_t; t equals to 1 to capital T, then mu and this is going to be I want this to be larger than n epsilon right. I have just multiplied this the denominator T I have taken it on the other side.

And, also let us multiply both sides by some lambda positive. This is what we had done in the last class also and then exponentiate both sides it will if I do that it will not change my probabilities. So, is this manipulation correct? What I have done here is simply I have substituted the value of mu hat which is nothing, but this sample mean divided by T, but I took T on the other side.

Student: T time X_t minus mu (Refer Time: 07:17).

X_t minus mu on the whole thing. So, then I just multiplied this thing by lambda.

Student: Lambda greater than zero.

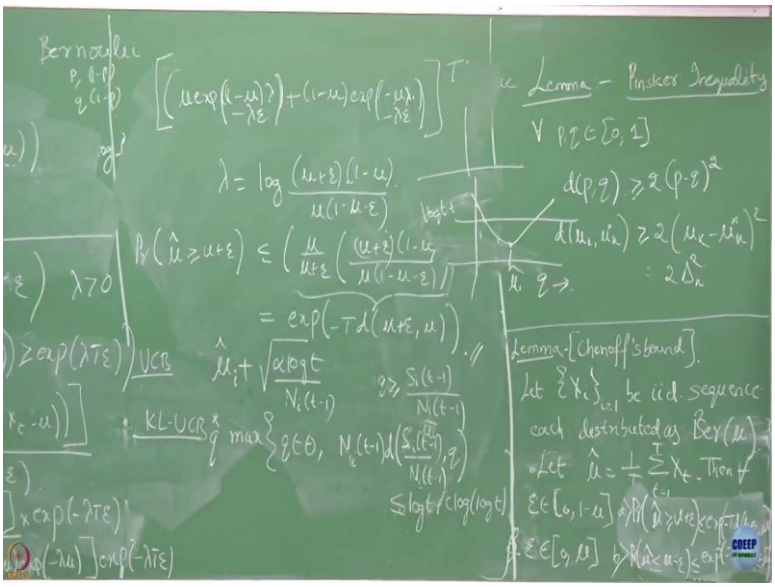
Yeah, lambda greater than 0 and then I exponented both sides. Now, we I am going to use our usual bound what was that like the Markov inequality. So, what I will get if I apply my Markov inequality. So, Markov inequality give me exp lambda where T X_t minus mu. This whole divided by exp minus lambda T epsilon.

Now, this is exponential of this sum and what is our assumption? This X_t's are iid sequence. So, this expectation of this exponential sum, can I write it as product of expectation of this exponential terms? Now, I am going to expand this. What is this expectation? This expectation is about I know that this X_t is going to be 1 with probability mu and this is going to be 0 with probability 1 minus mu.

So, if I just do that this is going to be exp. So, this is with probability mu this exp lambda 1 minus mu and with probability 1 minus mu this is going to be exp minus lambda mu,

and then we have this exp minus lambda T epsilon. So, what is this going to give? Now, you notice that this term here within the square brackets, now they are the same for every capital T every t going from 1 to capital T.

(Refer Slide Time: 10:27)



So, this is now going to be simply mu times exp 1 minus mu (Refer Time: 10:34) lambda plus 1 minus mu exp minus mu lambda times and then there is also one t, but I am just going to what I will do is.

So, there are capital T time. So, if this lambda e I can assume that this is I have been added capital T times and then I can take each lambda e inside. So, this is going to contribute lambda e here and also minus lambda e here I have done this entire thing is raise to the power capital T.

Now, this is a function of lambda and this holds for any lambda positive, right? Now, what I can do I can look for a lambda that minimizes this quantity. So, now, you can just look at this quantity and you can just differentiate and see that we if you are going to set lambda to be log of mu plus epsilon 1 minus mu I am some just writing it. This is the one that is going to give you that is going to minimize this exponent.

And, then once you are going to plug in this you are going to get. So, once you are going to get this I am skipping a details like you can now when you plug in this quantity and then simplify you can get it. This quantity to be upper bounded by mu plus mu plus

epsilon. So, here I have written inequality, but this is not equality right, this is inequality here. This is the Markov inequality here.

So, when I plug in this quantity here what I am going to get is 1 plus mu plus epsilon 1 minus mu into and mu plus 1 minus mu epsilon to the power yeah. This power is 1 upon mu minus epsilon and this whole power to the power T. So, I have just what I have just done is I just lambda this quantity you have plugged in this and just simplified then it is just going to get this.

Now, you can again go back and notice that this quantity here is nothing, but the divergence between mu plus epsilon and mu and this is what we wanted to show. So, I have just skipped this like whatever this bound we have you can just again manipulate it and write it as divergence between mu plus epsilon and mu. So, you know the formula for divergence between mu plus epsilon and mu right? Just go and see that that could can be expressed in this form.

So, that is why and this is what we had made a claim fine and similarly for and make sure that for the choice of epsilon if epsilon happens to be in this range it is going to be less than 1 minus mu here that means, this quantity here 1 minus mu minus epsilon is going to be non-negative already. So, every quantity here is a positive quantity and you could write this. And, now similarly for this case you can again show that when epsilon happens to be between 0 and mu we can come up with a similar one fine.

So, I just want to stop this discussion about KL-UCB at this point here. So, what we have just discussed about KL-UCB is? We have given the algorithm and we have just discussed that it is based on this tighter concentration bounds and this tighter concentration bound is happening because we are using only Bernoulli distributions there.

And, now once we have this bound then in the analysis we can expect that if you are using this bound divergence will come and we have we have stated now already that the expected number of pulls upper bound is coming in terms of this divergence and this happens to be a tighter bound and what we have for the UCB.

So, but you need to note one point here is that in the case. So, what was the primary difference between UCB and KL-UCB?

Student: (Refer Time: 16:41).

The way we are choosing the index, right. So, let us rewrite this. So, the index in the case of UCB was mu hat plus some alpha log t by number of pulls. So, this is let us say for arm i; this is for UCB whereas, for KL-UCB. What is? This is max over q belongs to my range and what was this and I wanted to see N of a this is for i let us say i t minus 1 divergence between.

Student: (Refer Time: 17:32).

N i t minus 1 comma.

Student: q.

q and this is less than or equals to.

Student: (Refer Time: 17:41).

So, earlier finding the index of arm i you just doing this, now to find an index of arm you are basically trying to see if solve an optimization problem right. This is an optimization problem right you are trying to find a q which maxima which is the one which is satisfying this condition. And, for q I already discussed for q which are larger than this quantity that is where this condition is going to be violated because this quantity is going to be increasing in q sorry, this quantity is an increasing function for q larger than this term.

So, so we have seen that for q greater than or equal S i t minus 1 divided by N i t minus 1 this guy is an increasing quantity. When x q equals to exactly this is 0, but if you increase q beyond this it is increasing. So, we know that at some point if you keep on increasing it has to violate this condition and the point right away at it violates that is the maximizer.

So, but so, in general like if you just want to implemented a plane optimization problem it could be pretty computationally intensive. So, when you are doing it in the code you have to be careful like if you just write this it is going to take a lot of time to run. So, you need to exploit the fact that this guy here is increasing in q and see at what point it is going to be violated and then find the index based on that.

So, this actually this is like for a given p if you look it as a function of q it is a convex function ok. So, let us say it is a convex function. If it is a convex function you can always try to see at what point it is going to cross violate this condition, but you should not just plainly you call some module which implements the convex optimization and tries to find a q here. If you just to give it to some module which is finding use this convex optimization methods, it could take quite some time to solve this.

So, try to exploit the monotonicity property of this divergence and based on that quickly try to find out this q quickly. If you do that you should be able to reduce your run time of the algorithm significantly; if you are just going to use some convex optimization module to run this it could take a quite some time.

Student: (Refer Time: 20:35) equal to 1 this continuously equal to?

At q equals to 1.

Student: The function (Refer Time: 20:44) log 1 minus q log p 1 minus p by 1 minus q (Refer Time: 20:51).

1 minus p. So, I am not following where it blows up?

Student: The divergence term.

Yeah.

Student: Summation p i log p i by q 1.

Yeah.

Student: If q is 1, then 1 minus q will become 0.

Yeah q equals to 1.

Student: Yes.

But.

Student: For every i it is goes up to infinity so, (Refer Time: 21:10) be some q for which this is equal to log p, but say log t. So, what exactly I have been maximizing every arm will be (Refer Time: 21:19)?

No, like every arm why should it. So, t is changing in every round right?

Student: Yeah.

And, this is also increasing function as you increase q. So, at some point it should be exceeding this.

Student: Yeah.

So, that is the point so.

Student: (Refer Time: 21:34) q.

Yeah, that the whatever is the maximizer that is we are going to call it as an index of that arm right. So, this is like q star is the arc max let us say which is that value that is the index for me.

Student: Then what about the criteria then if it is increasing (Refer Time: 21:54).

If it is increasing, right?

Student: It will always be increasing.

Yeah, it will be increasing for like this is fixed, if you are going to increase in q it is a increasing function. So, fine like yeah you can write like exactly at what point, but we can just write less than or equals to till this point.

Student: Ok.

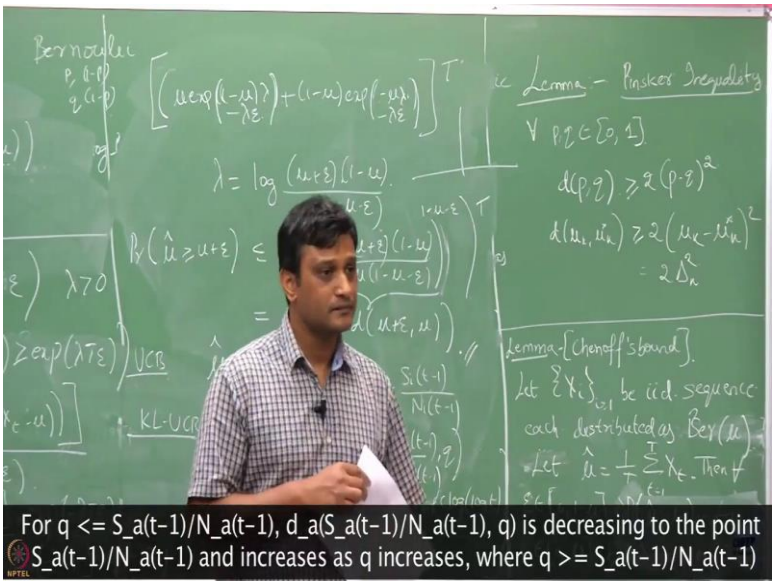
So, maybe even if you want to just find exactly what is the q that is going to exactly equals to this that is like a finding a 0 crossing of a function right even that could take quite some time or maybe like maybe if just see if you can use that property to quickly figure out what is this q.

So, I want to saying is try to find a good way to implement this like it could it could take some time. Just use the monotonicity properties may maybe like you can just. So, equate it to 0 and found a 0 crossing if that gives you faster q value or you can just do a kind of search.

Student: Rapidly (Refer Time: 22:53).

Yeah, you can just rapidly increase your q and see if it crosses then bring down your step size ok. So, I think even if you have just going to call a zero function that is it will maybe do a similar thing, but that module can be if you see if you can fasten it somehow by your own method.

(Refer Slide Time: 23:15)



Student: (Refer Time: 23:16) they could be. So, this is convex function that we have a minimize also. So, if q is less than S a t minus 1 S a t by this N a t.

Yeah?

Student: Even then you are going to increase.

Yes.

Student: Algorithm might a q it is (Refer Time: 23:37).

On the other side.

Student: Yeah, obviously.

Right that could possibly happen, right. So, so what he is saying is. So, let us say this is my that ratio let me call this as simply mu hat mu I hat at that time this ratio that is the. So, mu i hat. So, I will have something like this at as q goes and this upper bound is a constant let us say this is my log t plus log t whatever it is. So, the way at the point at which it is violating maybe beyond this point or beyond this point. So, which one you are going to choose?

Student: Like upper point.

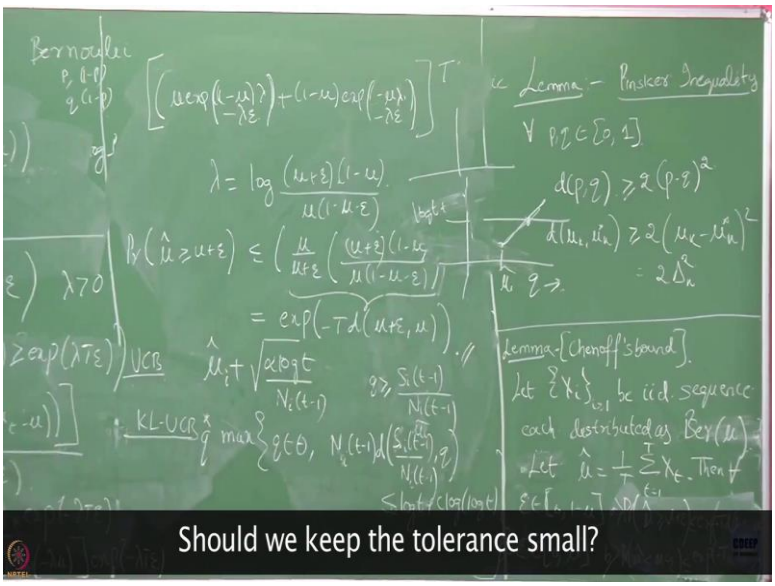
Upper point. So, that is why I like just ah. So, at why not the lower point also?

Student: (Refer Time: 24:31) maximizing.

Yeah, you are just maximizing over all the points. So, you just now already you have to take the lower point was just focus on this part, yeah. So, another one thing like if you know if you just do not exploit this property and then just see and give the constraint like this is my constraint I want to keep it below this and look for the maximum value of this function.

It is going to find this extrema, but if you just to call a standard optimization model it could take a quite some time to do this. But, now you know the property you have to just like you do not worry about this you are increasing and just want to see where you are going to cross that point. Just quickly find it out.

(Refer Slide Time: 25:15)



Student: (Refer Time: 25:15).

Yeah,. So, yes you need to like keep that tolerance very small.

Student: Yeah.

Yeah. So, otherwise when do not even if do not go for arbitrarily small and also do not keep it too large.

Student: Is that specified or not specified.

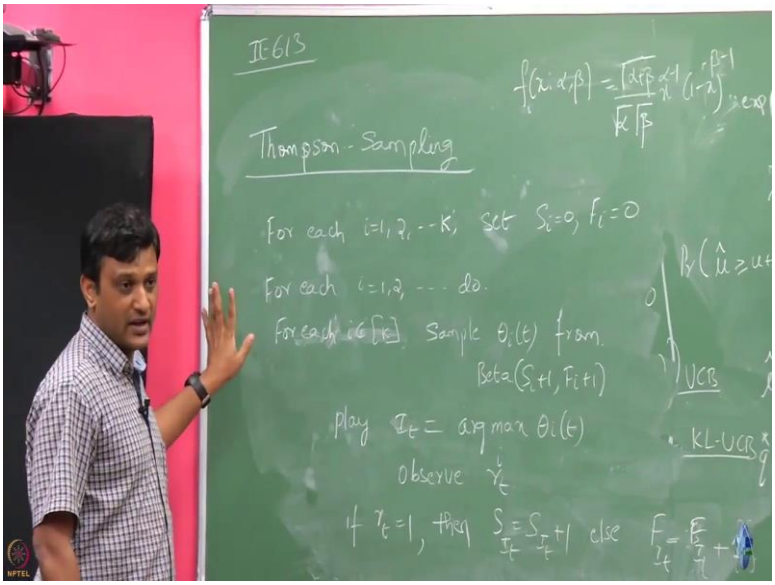
It is not specified that is that is we leave you to your judgment what should be a good value. So, if you are going to choose it very small right even then you have to do to many hollowings and it is going to take a too much time if you keep it the tolerance very large we will quickly find it, but it may be pretty off. It may just end up here.

Student: Ok.

Ok, fine. So, I want to just quickly discuss this Thompson sampling and we will not go any much details into its algorithm. So, there is another set algorithm called Thompson sampling which is based on Bayesian approach. What does that mean? It assumes that the parameter.

So, when I say environment, it is nothing, but a set of parameters right which are defining the means of that distribution. It assumes that those parameters is itself drawns from some distributions and it is going to assume some prior on that and based on your observation it keeps on updating that prior. And, then draws parameters from that prior and then looks for the parameter which has the highest value.

(Refer Slide Time: 27:02)



So, I am just going to quickly write the algorithm and we will discuss it. So, all of you know beta distribution? What is a beta distribution? So, beta distribution with parameter alpha and beta has a pdf which is as following. Yeah, x to the power alpha minus 1.

So, beta distribution with parameters alpha and beta. So, this is a pdf of a beta distribution with parameter alpha and beta has a distribution which is given by x to the power alpha minus 1 and 1 minus x to the power beta minus 1. So, you understand this notation.

Student: Gamma function.

Gamma function, all of you know what is the gamma function. So, it is and it is the proper pdf and defined for all x what is that x between 0 1, ok. So, the Thompson sampling uses this beta distribution, i equals to. So, this is a simple version I have defined here for the case of Bernoulli reward distributions. What it is doing is for every

time it keeps a track of how many ones or a zeros I observed from a particular distribution. So, 1s corresponds to success, 0s corresponds to failure.

So, then it want it is going to assign a beta distribution for each i with a parameter Si plus 1 Fi plus 1, what is this? This is Si corresponds to number of successes number of 1s it has observed, Fi corresponds to number of 0s or number of failures.

And, now it is going to pull an observe a sample from that beta distribution and then it is going to see which is the one which has the highest value of the sample and then it is going to play that. And, if it observes one it updates the success of that arm by 1, otherwise it updates the failure of that arm by 1.

So, what it is basically doing is, it is going to assign priors to each arm through this beta distributions and every time it is going to update those priors based on the success and the failures I am going to assign observe for those arms. And, then it just repeats this process every time playing the arm which gives me the highest sample reward from each of these beta distributions.