**NPTEL**

**NPTEL ONLINE COURSE**

**REINFORCEMENT LEARNING**

**UCT**

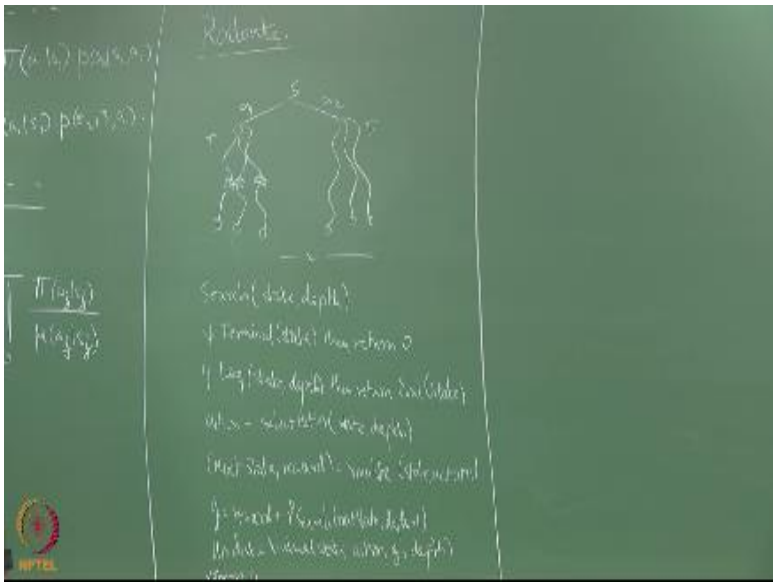
**Prof. Balaraman Ravindran**

**Department of Computer Science and Engineering**

**Indian Institute of Technology Madras**

So I am going to move on to a couple of other things about more but Monte Carlo methods which is actually not there in the book right, so rather than we need to add the you see trees paper to the model page right, bandit based Monte Carlo planning, what is so amazing about it, okay.

(Refer Slide Time: 00:40)



So there is a technique which people have used people who typically talk about Monte Carlo methods and you saw something called row loads okay, so the idea behind roller is the following so I have a heuristic way of selecting actions I am trying to solve the power problem they have some heuristic way of selecting actions okay, so what do I do is when I come to a particular state right, so I use the heuristic right, and select actions and then multiple trials from there, right so I use whatever heuristic I have I select run multiple trails so what is this the heuristic you can think of it like a policy, right.

Policy is what a mapping from states action right I am not talking about value functions or anything just a policy I use the policy RN several trajectories like they just like we talked about in the Monte Carlo case and then what I do here is evaluate all the different outcomes that came from running that particular heuristic, right and then I pick the action right that gives me the best possible expected outcome.

If that did not make sense let us try it again, okay if you have a deterministic heuristic then what you should do is probably select actions probabilistically at the beginning and then run your heuristic okay so I start from some state let us say I pick action a1 and then let us say I can pick action a2 there are two actions that I can pick, let me heuristic will suggest something well, let us say pick action a1 first and then I run, I will policy $\pi$ and likewise pick action a2 run my policy $\pi$ okay, then I take the average value I get here, so I take the average value I get here then I take the action which ever gives me the better average value, right.

So this might not be action given by $\pi$ or it might be action given by $\pi$ we do not know, right so all I am saying is given any policy $\pi$ I can do a roll out version of this, okay what is it nice thing about doing this rollout version I am guaranteeing you that whatever policy you execute by virtue of doing roll outs on $\pi$ right, will be at least as good as $\pi$ and quite often better than $\pi$ if a $\pi$ is not optimal right then this rollout execution of the policy $\pi$ will be better than $\pi$.

This rollout is this time generating multiple trajectories using $\pi$ right, it is like rolling out the policies it is something like q(a) right but this predates all of this thing so this kind of roll out the idea of execution right, this is nothing to do with the first of all this is nothing to do with MDPs

right, so it could be any kind of games that you are playing it could be adversarial setting it could be anything right, I talk nothing about value functions right, I talk nothing about returns nothing is it predates all of that okay.

This is in fact it is called rollout execution of a policy okay, roller the execution could the π could come from any heuristic right, it did not have any notion of optimality associated with it so what the roller the execution gives you is whatever be the performance of π that we had earlier I can better it if it is better able, okay if I do not get a better policy than that with a very high likelihood π is optimal.

It why, think about it is very much like doing one step of policy improvement then I have done one step of policy improvement over π if you think about it right so what did I do here this is essentially an evaluation of π like I did an evaluation of π specific for starting from one step after s right, so I just need a policy evaluation of π and there was greedy with respect to that policy evaluation so this is called rollout execution. Yeah it is all a π till you reach the goal state here. This depends, now I want to tell me what it depends on.

UCT the least where the arms and you would have from the root to the in each iteration you mark from the root to the least that is you are taking at one arm at each time step. Yeah, and the each of the individual notes you have that upper bound term there to maximize. You are confusing to many things but they are gone yeah so there was nothing good we follow at one time still be follow one that was 15 TVD still with you. Yeah. Using that here we are following many.

Basically only one π but I am doing many territories. Yeah, there it is holding one trajectory at each time step we are following one. Yeah, so we will come to that right I am not okay I am not at talk to you about you trajectories just ask me questions on what I have taught you so far. This is not a history, this is just rollouts so I am just talking to you about rollouts I have not started talking to you about trajectories okay, your question was how many times should have run those trajectories.

Of course but how much anyway wants to tell me what it depends on yeah, indirectly exactly so closer so it depends on the variability in the system, right. If this is deterministic and $\pi$ is deterministic one trajectory is enough, right if the system is very stochastic you know all kinds of random things are happening then you probably need to run lot of trajectories all if $\pi$ itself is a very stochastic policy you will have to run a lot of trajectories, okay.

So it depends on the variability in this sampling process if the variance in the samples you draw is very small it is sufficient run a small number of times, okay so any other factor that will determine this trajectory length why should trajectory length the determining factor, in directly affects you know then not about the variability trajectory length can be a factor in another way computation right, so how much computation time do you have to spare between every decision is a very practical consideration and you have to actually be cognizant of it right.

So the length will affect the variability but also affects amount of computation that you have to do, so the longer the trajectories are the fewer that you can generate. Ironically the longer the trajectories are the more you want to generate because it increases the variability but you will have to do the opposite so said any way of cutting down variability here, exactly so do not go all the way to the end right, stop somewhere in between depending on how much time you have how much computation you can spare for this and how much variability is there in the system okay, so if you can find a good heuristic that tells you how much more it is going to cost if I run till the end, right you can use that heuristic as a terminal cost and remember when we talked about why we call a value iteration dynamic programming I said you can think of the value as a terminal cost right.

So we have one step problem and then we have a terminal cost like that here you will have a many step problem and you will have a terminal cost, right so what should be the terminal cost or the natural terminal cost for us to use it is a value function right, if you have an estimate of the value function you can use that as the terminal cost, see so far I did not talk about the need for a value function and I can do rollouts just given a policy I do not need a value function but the reason for me using a value function with rollouts is if the trajectories are very long the variability is very high, right.

So the longer vector is a case the smaller the number of samples you can take refer it is going to be very self defeating, because the larger the variability you want to take more samples so to kind of get around that people started using estimates of value functions, okay. So once you have value functions estimate you can start doing something clever, okay good. So this instead of apartment rollouts was like a kind of a peripheral thing in the RL community for a while right in fact when I started teaching this course I used to give a handout from a dynamic programming book on load of rollouts because it was not covered in the RL textbook so I used to take a Xerox of well this was before every text book ever printed became freely available on the internet so I used to give a photocopy of the few pages that discuss rollout policies right.

I gave it out to the class to read about rollout policies and things like that and then things change dramatically when I go I thought his name was harder to pronounce well Chava and another Hungarian with a hard to pronounce name came up with something called upper confidence trees UC trees, okay. so UC trees at the heart use a rollout like idea right but we also use the notion of value functions and upper confidence bonds that we saw for bandits that they use the notion of up UC upper conference bound bandits UCB1 specifically they use the idea from UCB1.

If you told you there are many variations to UCB they use side is a very simple one the UCB one and then married with them what they call the Monte Carlo tree search, so I am going to explain Monte Carlo tree search first and then I will tell you what they did to get UCTs and it turns out to be one of the most powerful planning algorithms that people use in the literature, okay. I should explain what I meant by planning here because it is very unique to Richchatin.

So Rich introduced this difference between learning and planning into the community, so learning is when you try to solve an MDP without knowing any of the parameters well, it is online that it is completely sample driven and so on so forth. While planning is when you have a some kind of specification of a model right, and you use the model somehow to solve their problem so all of these Monte Carlo methods if you use a sample model.

Remember we talked about sample models and full models right, so if you use a simulation model or a sample model it comes under the regime of planning because you are using a model okay. So whatever we talked about can either be planning our learning algorithm depending on how you draw your samples, okay if you draw your samples from the real system then it becomes a learning algorithm if you draw a samples from a simulation model then it becomes a planning algorithm, right.

But there are other methods that use the model more integrate a more integrally in their method as opposed to whatever we have spoken about so far, so there are many Monte Carlo planning methods so we will talk about Monte Carlo tree search. Let me see if I can might what I really need to get my classes. The will sorry hand it must be much worse than usual today but let us put it down to variability in the system. So you have to go and heat the paper to get a better estimate so essentially objective one subroutine of this right, so this is a search function I am writing where is a function called search, okay it is tarts at some state and some depth at which it is going to be called that right.

So I have some kind of a depth to which I will search like some depth to which I will search and I start by checking if this is a terminal state, okay if it is double a state I just returned a reward of 0 in the simple case but i will return the terminal reward so whatever is the terminal reward, right. So in this case all the terminal rewards are 0 we are assuming that all the terminal rewards are 0 otherwise I will return whatever is the reward okay, and then I check if I have reached my required depth right.

So if this say something is a leaf state that means it is a state at which i will not do any further processing right then I evaluate whatever the state is whatever is the value of the state I will return that value this is like the terminal cost that we spoke about it, if I stop here I leaves the value of such state as the terminal cost right, so we will evaluate that state somehow and will return the value, okay. So what is next okay, now it is not a lipstick okay, it is not a terminal state so that means we have some assume so we are not here we are not here so that means you are somewhere here right,

So somewhere there what do I do is elect an action right, so how do I select an action sorry, based on the yeah, so heuristic function so whatever was the original heuristic function given to me, I will selected action based on that heuristic function right, I will select an action based on the heuristic function then I will take one step I will simulate the trajectory so this essentially what I'm doing it I simulate the trajectory right, I find what the next state is I also find one is the reward I get for taking that action okay. Well if you can actually behave in the real world go ahead and behave in the real world but how are you going to reset you back to that thing and generate another trajectory right, so if you want to do something like this you better have a simulation model right, so this is the whole idea behind rollouts so you need to have a simulation model okay, that is a point I missed or but you need to have a simulation model can absorb obvious, right.

When otherwise how am I going to generate multiple trajectory starting from here, if it is a real world then I have to roll my robot back there and then say okay run again okay, then catch it does not put it back there and say run again and so on so forth that is not going to work right, or imagine playing games, right just like playing with a kid right so the kid play some moves and say no, no I want to take that back after four moves so there will be like that right, anyway so you do the next state and you get the reward based on a simulation models.

And then what do you do you take your return right, initialize it with the reward right and then what do you do gamma times search from the next stage essentially you are going to face the entire path until you reach a terminal state. Once you reach a terminal state you keep popping back all the rewards that you get not a terminal state once you reach either a leaf state which means you have reached the depth at which are going to cut off or you reach the terminal state at what you do at that point is you keep passing the rewards back up the calling stack, right.

So this is like a recursive call I will keep popping the reward back up so this summation when it finally finishes computing will have the return the complete return or the kind of an adjusted return why I am saying adjusted return because if i come to the stopping to indicate condition here I will use some kind of a value function estimator some evaluation that I have so it is like a corrected return or a truncated return.

So i will have that as my g right, and then what they do I update the value of the state and action, right I say this value of state and this action you took and that is the total return and they said depth at which the action was taken from the state okay. So why is the depth important here was it the first terminal thing I would not even reach the right the state is a terminal state I return 0 at the very top I would not even reach the update condition. Yeah, why do you how will the depth give you how many states you move through okay, and why do I need that so it turns out that we are not even making any mark of hidden assumptions here, so the so closer I am to the end right so the value function will be different right if I take if I have five more steps to go or if I only take a scene 5 steps I have a long longer direction to go right, so I lead to maintain the distinction with the value function right, I am still not come to UC trees.

I told you about this right, so when we have a fixed horizon when I say this event is going to stop at time t=10 or something like that right, so if you are one step away from that you will behave in a different way, if you 100 steps away from that will behave in a different way so I am just giving you the number of steps that I have gone so that if you want to specialize your value function based on the number of steps that you have taken you could do so, okay.

But there is another reason for carrying the depth on but for the before we get to it okay, before I talk about upper confidence bounds and stuff like that so the Monte Carlo tree search algorithm okay, we require you to carry the depth run only for the diversion as you can throw away the depth in the update, right. So you need the depth here, so you can throw in a depth in the update if you are assuming everything is completely markup okay,  yeah it does not matter wherever you are in the system right, so you do not need depth in the case that is what I am saying even in this.

Yeah, but only in the previous one you need left leaf right, so this is fine so when you call this you call it with your initial state and the depth of 0 right, so here when you call it you call it with the s and a depth of 0 okay, and you could call this multiple times from data, so every time you call it, it will traverse one trajectory and it will give you one return corresponding to that, right.

So every time you call this it will give you a return corresponding to that and then you can choose your best action based on the returns that you are getting from that, okay is it clear right.

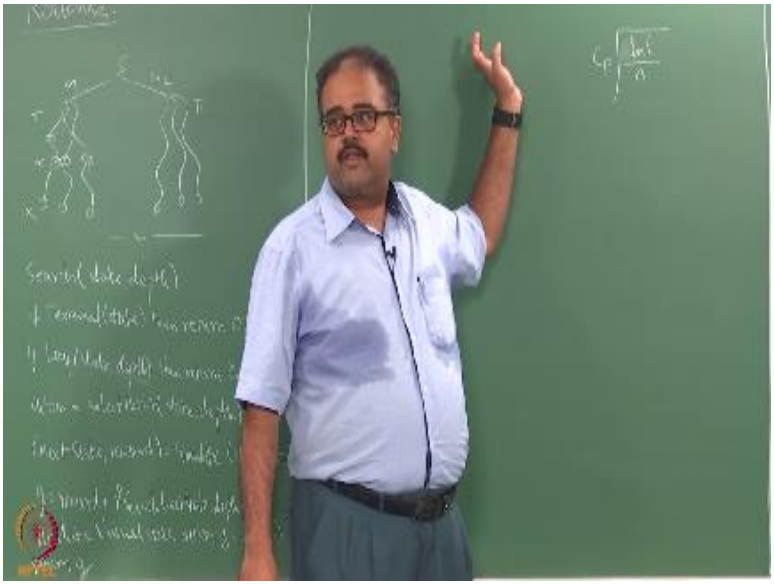
So from here you will run this multiple times right, now is take the returns again and again and from there you choose your best action. Other yeah, like I told you right, it depends on how you choose the actions if your policy has sufficient acceleration then you are fine right, if you policy a sufficient exploration you are fine that policy does not have sufficient exploration then it becomes an issue right, and how will each is the best action anyway whichever next state gives you the maxi so basically you will simulate each action once okay, and then see which are the next states you land up in which other next state gives you the highest value you pick that action we go okay, once or multiple times you can always if you remember how you select actions using the value function right, you have to remember what we covered earlier so I said how to recover a policy given a value function.

I given how to behave greedily with respect to a value function not a Q function v function so you could do that or you could modify this whole thing to give you Q functions instead of V functions highly modified is to give you Q okay, so this is this is a basic structure of what is called as Monte Carlo tree search okay, and depending on how you select actions which you can get different kinds of algorithms out of it and the most popular one is called UC trees right.

Where the select action is going to use a upper confidence bound like formulation so I am going to look at the expected reward for taking an action plus a term for the confidence one right, so what is the term going to depend on number of times you have pulled that arm anything else Subha, yeah, so there is one magic constant that comes in there okay, which is what they needed to actually help them prove some results okay, and they had one son bound on that $\pi$ constant.

Yeah, I forget what is it bound was remember what the bond was yeah, so there is a bound on that $\pi$ constant essentially it is exactly like you are all you see bandits right, the expression will all be the same except that instead of just having root of.

(Refer Slide Time: 27:22)



so it will 2 lnT by right, that used to be the that is to be the constant that you add right, so here what you do is instead of that you have some magic constant CP times root of ln T/n okay, so when you select action so instead of selecting actions randomly or I are going to fix $\pi$ so what you do is you look at the current estimated value function for each action add CP root ln T/n okay, where CP is a constant that is going to depend on the problem that you are solving okay, so the higher the variability in the system the larger the CP has to be right.

Because you want to explore more right, higher the variability in the system the larger the CP will be okay this is something you typically end up with determining empirically but there is some bound on how large you have to search and so on so forth right, I can get you that numbers later. But you do this CP into root ln T/n plus the q value right, and then you select actions according to that and then go back and pick your best action.

So what this is by U so you have you have your explorer exploit problem here right, so you need to explore many different actions in this actual selection space so that when we go back and select your best action you have selected every action offer enough right but you do not want to keep exploring all the time because when you go back you will not talk when you have wasted too many sample suppose I give you like a thousand samples that you can take you would like to explore the more promising actions more than the obviously hopeless actions so if you do some kind of uniform acceleration you are going to skip your budget too much right across all of these things so doing UCT like exploration gives you much more focused learning right so just like we have in the UCT case this gives you much more focused learning okay, and this is one very elegant way of converting a bandage solution for solving the full RL problem right. Only to carry of car is you need a simulation model you need a simulation model to be able to do this right, if you have a simulation bowl you can do UCT style planning and turns out that in many cases you can come up with some kind of a rough simulation model, okay.

And you can get away with it and so it turns out now it's become one of the very popular solution methods and in many robotic situation where you have a simulator it turns out to be a very, very robust planning algorithm right, and a lot of robotics now like to use UC threes in their process okay any questions on this so we will be putting up the paper for you to read I will not say that the paper is easy to read okay so most of where unlike his lectures most of Chavez writings are very hard to penetrate so how many of you watch Chavez lecture.

One, linear bandits I told you to go to see a video on linear bandits right its links on the Moodle page not sure it's in Moodle all the recharge our lectures are there is it okay, so those are the three classes that Java took right so you should see it not enough to just see the link rather watch the video so many people are telling me three links are there and when I asked have you seen the video only one hand goes up did you see the video of Chavez you are not herein the RA all happened you did not understand anything okay.

You were here when they will trap happened here you are going to say something, you have live demo  hopefully you will understand a lot more now and the pleased to see that so any questions

on this you see threes yeah, if you can come up with the parameter is representation for the value function okay, that is well behaved in the action dimension then it will work so we will come to that aspect of it later in the thing where I talked about using this kind of parameter is representations which since you asked.

I am telling it will work if you have a parameter is representation so when you do this update value right, so essentially that update our actual set of parameters that represent the function just like we talked about in reinforce right, where we update the policy parameters you need the value function parameterized like that then you can get it to work and give it more conditions on the well-behaved nests of the value function.

Oh, in the continued action space is a good question so let us think about that so that is the thing I have right, so this N is essentially the number of times I have basically taken action so that is basically my depth kind of thing and oh, yeah n is your depth so that is the reason you need to pass depth everywhere right and ln T is the number of times you have taken that action.

Total number of yeah that is what it is right the depth will be the total number of pulls yeah, now T is the number of times have taken that particular action right, so there is that you be n is the number of times I have taken that you sorry, sorry T is the total so T will be the depth is I have to keep track of n how many times have taken the action so far right, so one possibility is to say that I am going to do this in a parameter explosion but glad to see whether that will work or not right.

So yeah so without the exploration bonus this will work in the country action space for this we will have to figure out how to make it work okay fine, all set okay we can move on to the next topic so this is end of chapter five so starting chapter six but chapter five in the second edition of the book has one additional section okay so where they talk about some of the recent work primarily being done by with certain and his group on I'm trying to reduce variance in Monte Carlo sampling right.

So one thing which they talk about is when you really have long trajectories right, but you have a small discount factor I say gamma is like point three or something so the contribution in the

return of the subsequent terms is going to die down very quickly right so you really do not need to maintain this full product that you could get away with doing something that is truncated right I am looking at the Pxi/Qxi right I do not have to do the whole product I can do something truncated.

So how do you do that in a principled manner and give some kind of guarantees about convergence and stuff like that the he gives a very, very high level quick discussion about that in the last section in the chapter in the book right, so again you can read it for information that means I will ask questions on it in the exam is what it means if you want to know what it translates to but there are other things you say ask you to rate which I did not say our for information so those are fair game.

**IIT Madras Production**

Funded by
Department of Higher Education
Ministry of Human Resource Development
Government of India

www.nptel.ac.in