NPTEL

NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

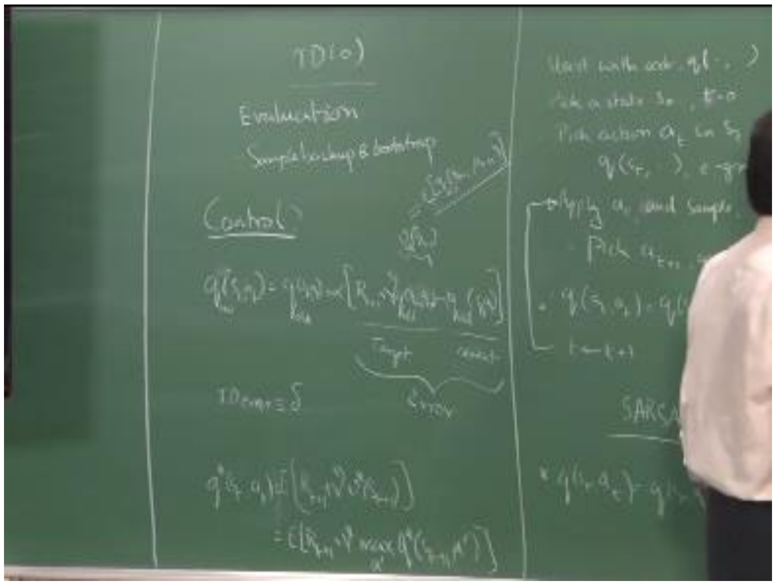
Q-Learning

Prof. Balaraman Ravindran
Department of Computer Science and Engineering
Indian Institute of Technology Madras

There is another way of looking at learning with the learning control right. So I am going to think about how can I learn control using a value iteration kind of an approach right. So what do I want, so that is my target right and that is my current estimate of something right. So remember I am learning the Q function, so what does the Q function tell me, take an action whatever is action given in the Q arguments right.

And then behave optimally that after right, the first action is whatever I take right there is no question of optimality or anything about it.

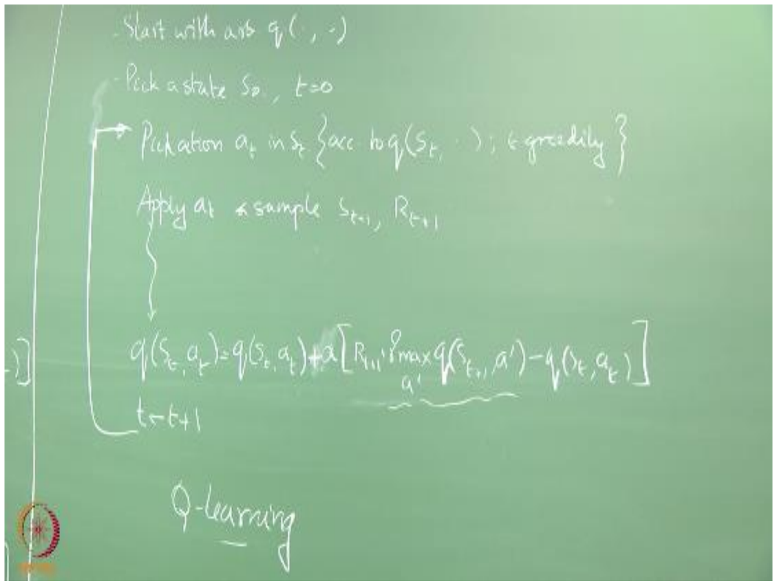
(Refer Slide Time: 01:23)



That is being the thing that I am looking at right, I should move away very quickly after writing something on the board that other vegetable waved me away right. I mean people x kind of buy that right. So what I am really looking for when I say Q* I say is expected value of RT+1 which is what I get for performing A+ gamma times V* of whatever comes afterwards correct.

But we know about V* we can write V* in terms of the Q function how is that V* max it is a best right there is no π here talking about optimal value function right. So this is some term equal to correct. Now we can now do our stochastic updating rule this is my target this is my current estimate, so putting this together I can just write a rule like this right.

(Refer Slide Time: 03:39)



```
- Start with arb $q(\cdot,\cdot)$
- Pick a state $S_0$, $t=0$
→ Pick action $a_t$ in $S_t$ $\{$acc. to $q(S_t,\cdot)$; $\epsilon$-greedily$\}$
  Apply $a_t$ & sample $S_{t+1}$, $R_{t+1}$
  $q(S_t,a_t) = q(S_t,a_t) + \alpha[R_{t+1} + \gamma \max_{a'} q(S_{t+1},a') - q(S_t,a_t)]$
  $t \leftarrow t+1$

Q-learning
```

So what will be my rule, so target-current estimate right, let us see people get that right I mean whatever you are trying to get earlier we were just trying to take the definition of the Q function is the expected value of the return. Now I am looking at the definition of the Q function as I expected optimal Q function as expected value of this expression, but now I am converting that into an update rule okay.

Now I can put in the rest of it around this so what is a nice thing here just like in the expected SARSA I do not have to pick the action before hand I can pick the action afterwards also right because I am only doing the max right. So I do not have to do the pick an action first and then apply it that I can move this within the loop try to remove this I can flip this around basically.

Why is it important why am I so concerned that is because yeah, so there are a couple of things here so one thing the most straightforward answer is it could come back to S in which case I would have change the value function right I would have used one year for updating this here assuming that, that is the A I am going to take right.

But then I would actually perform a different A in which case my return see what I was trying to do here was substitute my return right. So I am going to see RT+1+gamma this because that is in there is a sample of the return right. So if the sample of the return that means I should actually be sampling according to whatever updation I am making, but I would not be doing that I will be sampling my different action.

If I picked another action after I do the updation okay so I will be breaking the assumption that this is a approximation for the return. So that is the reason we need to pick the action beforehand and then stick with it right. And later on when we see how we will be using a an approximate function value function right, we talked about contextual bandits right, we talked about policy parameterization like that you can use a value function parameterization.

So when you use that you do not have to come back to the same state right because of the functional form if I change the value in one state it may change it in some other state also. So the whole policy would get up now messed up so it becomes a little tricky then. So essentially the trajectory along which I am doing the updation will be somewhat a messed-up version of the trajectory which I am actually taking right.

So it might lead to some trouble in convergence this way, but that section I will actually be performing right so it is fine. As long as that is actually performing it is okay your question is what you keep changing the policy anyway right yeah that is why I said we do not use a explicit representation of the policy anywhere so it is okay.

So I really do not know what is a policy I am estimating the value function of is just that as $\sum$ and α go to 0 it will converge to something at the end right. As long as the return I am using here when I actually roll it outright I can roll it out, when I roll out the return name whatever I am using is using all the actions I took then I am fine.

What is it, yeah we really on the initial value of Q that is a policy you start off this here. So you could actually make it a valid policy or you could start off with all zeros in which case it is a uniform random policy, it will just be call actions with equal probability to begin with right. So you can do the same thing here which is empty space okay.

No not it is equal you just pick it the sequence happens after you have done the updation okay. I am picking it according to QST before I make the update here I am picking it according to QST after I make the update QST+1 okay here when I say picker action for QST+1 that is before this updation is happening.

So if I use Q volt Qnu you will see the theorem using Q volt here I am using Qnu for picking the action 80+1, I also put that thing in bracket I will come to that why I put that pick action 80 according to QST I put that in bracket but you can see the difference between this algorithm and that algorithm right nothing much except for that simple change in the update rule if you cool α and ∑ properly it actually converges to the optimal policy.

So what will the second algorithm converge to, optimal policy okay provided cool the α properly we had decreased α properly will converge to the optimal policy and this one if you have to decrease both α and ∑ properly it will eventually converge to the optimal policy. Yeah for α I have the usual stochastic averaging formula right.

So yeah, tell me 1MIT is one one thing that you fit the conditions right, so summation α should be infinity, summation $α^2$ tribulation infinity right. So that is basically the conditions for α so as sum over all α is so essentially I had to put an αt there and sum over all αt so just summing over αt should be infinity, summing over $αt^2$ should be bounded less than infinity.

So that is essentially the conditioner because the αt equal infinity tells you that you are making enough updates and $αt^2$ goes to let mean converges to a point listen infinity means that you will eventually stop making updates what, good that is what the rest of the class is all about, but so this algorithm 1 the right is probably the most popular reinforcement algorithm people use I said when I wrote down SARSA I said this is the second most popular right.

And that is the first most popular and it is called Q-learning. So chronologically td0 came first I think which proposed it in like 84 yeah I think which proposed td0 and 84 and short convergence in 88 or something and Q-learning was proposed by Chris Watkins in 89 and SARSA was proposed by Rummery and Niranjan may be 91, 92 and I do not know the exact date sometime around that.

So surprisingly both Watkins and Rummery Niranjan are from England. So lot of the early algorithms for RL came from England. And Q-learning is amazing so what is even more amazing about Q-learning is that since I do not care what is the sequence on which you are picking the actions right here I amusing an approximation to the optimal return right I am not looking at the current written.

If I am looking an approximation to the optimal written so I do not care about the trajectory in fact in Q-learning I do not even have to sample according to any trajectory in fact the next action I execute need not even come from ST+1, I can arbitrarily reset if I have a simulation model right I can arbitrarily reset to some other state right.

And then sample one action from there update Q-learning and then go and take action from somewhere else, some other state update you think you can just do a proper asynchronous generalized policy iteration kind of an approach, I just sample from all over the place just take one step each and keep updating it.

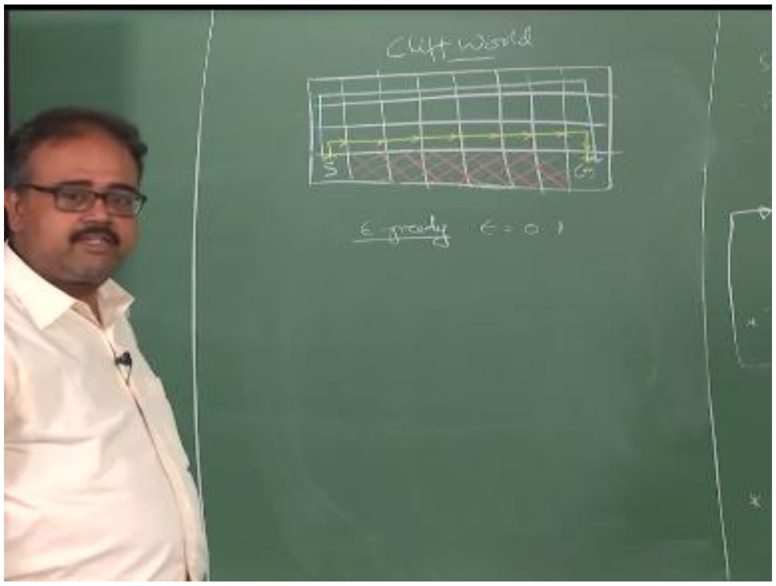
But still it is better to update it along trajectories like we saw in the argument for RTDP right, if you updated along trajectories it is more meaningful because the values should have more likely to have changed along those trajectories right. So that is essentially why we sample according to the trajectories.

So this thing need not be an ∑ greedy sample at all right. So I can sample actions arbitrarily I can just take pick an action AT and ST randomly okay, I does not have to follow any rule. So what does this do it just tells me which action I am going to update next right. If I am not sampling according to trajectories it only tell me which action I am going to update next.

So why do you want to pick according to this, so that you can just like an RTDP so you can focus the updates along the most most likely paths, so what the reason you do it ∑ greedy fashion is that you quickly want to converge to the most likely path that you are going to take under optimal policy and you want to do the updates along that most likely path okay. And that is why you use the ∑ greedy sampling for Q-learning.

And so what are the advantage of SARSA, SARSA you can continue to explore right and while exploring you learn something that is taking into account the exploration. So what does Q-learning do that is why I am saying you can arbitrarily with Q-learning but if you choose to behave according to the Q function you are learning, here it does not take into account the fact that you are doing exploration okay. So why would that matter.

(Refer Slide Time: 16:50)



So it is a simple grid world problem called the cliff world. So those squares marked in red of represent a cliff so if the agent wanders over the cliff is going to get a reward of -100 and to top it off agent will also die okay. Basically episode will end the agent will get a reward of -100 and it goes back to the beginning and you start all over again maybe incarnate and you start going all over again.

And otherwise you get a reward of -1 for every time step that you move okay. So everything is deterministic so for every time step that you move you will get a reward of -1 okay and yet we will try to get from this start to the go as quickly as possible no gamma here gamma is one right whenever I say no gamma does not mean gamma 0 means gamma is one okay.

And so what do you think will happen and while learning I use a ∑ greedy action selection ∑ let us say I fix it at 0.1 I am not going to reduce ∑ I will fix ∑ at 0.1 okay. So what do you think is the policy what do you think Q-learning will do what do you think SARSA will do. Well we said both the learn optimal policies and what is optimal policy here.

Alright so, here you go up there you go left right, right right, right, so this is the optimal policy of course if I want to write down the optimal policy elsewhere maybe I can just write down arrows everywhere does not matter right. So I can do down arrows everywhere or right everywhere except the last column where I go down this is optimal policy and there is no question about there is this is the optimal policy right.

At least for these states right for these states you have a choice you can either go right or down for these states this is the optimal policy right. Okay will Q-learning find this optimal policy yes, will SARSA find this optimal policy. In this case it would not because I have fixed ∑ at 0.1 right. So what will SARSA learn, will the show up if any ∑ is at a 0.1 let us say. Yeah it may take a safer and safer row.

If I give you more rows right it will probably go but then this is too small right it basically needs to exploratory moves to fall off the cliff and I given it like 10 moves in which to take this two exploratory moves. So it is very high probability it will take those two exploratory moves right. So it will try to stay as far away if it just goes here right this will be just two moves right now this is three moves.

So that is the reason it will try to go as far away as possible because the reward is sufficiently negative I said -100 right so that is really really bad negative reward. So it will try to go further and further away so this is the difference between SARSA and Q-learning right. So SARSA takes into account the fact that you are doing the exploration right, while Q-learning might learn the optimal policy right, it would also end up killing you more often while learning the optimal policy right.

There are also more you are name answers for this because if you continue to explore right not this is not exactly name but slightly less compelling reasons. If you are going to continue to explore after you finish learning right it is better to learn the policy through SARSA then through Q-learning, because Q-learning assumes that eventually you are going to be executing the optimal policy right.

But that is a reason for us not to execute the optimal policy greedily ever why, more but better than forecast is everyone there is a reason for us not to use the optimal policy greedily even after sufficient learning non-stationary. It is not the stochastic I am worried about it is a non stationary I am worried about right.

So if you are convinced that the world is not changing then great you can start behaving greedily after some time, but there is a chance if the world might actually some things might happen right, or suppose you are robot learning in a world right. So it is possible that obstacles can be moved around right. So it do not want to become so rigid that you are completely unadoptable when things change right.

So that that is they way, so you want to keep the option of being able to explore and discover new things. So you do not want to turn off your ∑ ever, but in such cases it is better to use SARSA than Q-learning, because Q-learning kind of ignores exploration at all right and it converges to an optimal policy and then it sticks there right.

So right and so the problem with the converging to optimal policy in sticking there is this right, if I want to keep my ∑ on and I will keep falling off the cliff by every one single move I make wrong where exploratory move I make I will keep falling off the cliff. So on the expected reward I get for executing this optimal policy will be far lesser then what my agent believes it should be getting because agent is ignoring the exploration okay.

So that is the crucial difference between Q-learning and SARSA that is why you would like to use SARSA in many cases. Q-learning is actually good in yeah in some aspects yeah, there is no danger and the state space is very large. So why should the size of the state space matter see it is exploring why I mean Q-learning can explore much more than SARSA it right because it ignores exploratory policy completely you can actually have a region became completely randomly and still learn the optimal policy in Q-learning.

In fact that makes Q-learning of policy method right you can do arbitrary exploration and still learn the optimal policy right. So I am at the estimation policy for me is the optimal policy and the behavior policy can be something completely random. So that is one reason you would want to use Q-learning in a large state space because I can behave arbitrarily.

So that I get to explore a larger fraction of the state space, everything will be slow both will be slow. Why would you think Q-learning will be slower compared to SARSA. So I am not a 100% sure about the convergence rates between ∑ I mean Q-learning and SARSA I am not sure about the convergence rates, I do believe that Q-learning converges faster than SARSA in general.

And I am not sure about large state space issues but I will have to look it up. Yeah but you should know that these are the basic algorithms right if you want to operate in large state spaces you will be using something else over on top of this, this would not be using plain vanilla Q-learning and SARSA in large state spaces.

So there are other issues that will come into play but I am not 100% sure about the comparison of rates of convergence between Q-learning and SARSA right great. So any other questions on these two okay, so I have this three actually we talked about expected SARSA as well right and in fact we can think of all kinds of hybrid algorithms right.

So for example, you can think of an algorithm that will take one step two steps according to the current policy and then you say max right. So what do I mean by that it is something like this right. So this is, this part is on policy right, so that class is off policy right. So that is why it is a hybrid kind of thing so you can do all kinds of, different kinds of samplers for your value function okay.

So in fact it is more common to use the Q function there, so that is still on policy everything is on policy but I am using more of the actual samples and less of the bootstrapping right. Why I say less of the bootstrapping because it is discounted by a larger power of gamma. So the total contribution of the bootstrap thing will be coming down okay.

So I can do all kinds of modifications to the return, so in fact there was a time when we are trying to get a reinforcement learning agent to play what is the thing called a hockey right. So somebody in CF, I wanted to do a project I wanted to get an RL agent to play air hockey in fact, it was building a physical system that was going to move the slider around and he wanted to use RL to control it.

And then he turned out in the simulator at least that he had built using a three-step return like RT+1, RT+2, RT+3 then the Q function of the rest they turned out to give a better performance or whatever reason right. So it is kind of a tied to the the stochasticity in the system and how far ahead is the system predictable and so on so forth.

So I think it turned out that case for at least for three steps into the future the system was more or less predictable and therefore he could use this kind of writer. So it is very complicated things right it is not easy to figure out what is the right look ahead that they should be using for these returns okay. So they more or less done with chapter six.

IIT Madras Production

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

www.nptel.ac.in