NPTEL

NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

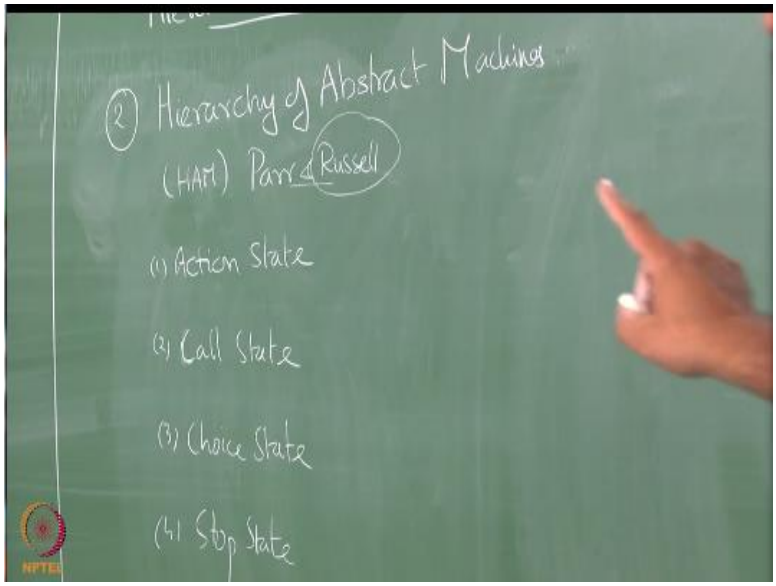
HAM

Prof. Balaraman Ravindran

Department of Computer Science and Engineering

Indian Institute of Technology Madras

(Refer Slide Time: 0:15)



So the second framework which is actually a very interesting thing unfortunately is not adopted very widely is called the hierarchy of abstract machines or HAM okay in fact with hands and become more like a curiosity in the RL community at least I do not see anybody write papers about HAMS or anything you talk to people are working in RL literature they would have read up on hands at some point but never given it very serious consideration after that but so Andy Berto thinks that the HAM framework is the most elegant and most complete of all the three major hierarchical RL frameworks are out there unfortunately people just have not built a significant structure over it except for people who work with Stewart result right.

So it was proposed by this get in on Parr right so yeah the options frame work was proposed by join a breakup, Richsatanne and Satinder Singh so the original options paper was satanne Sigh the citation for that and it was largely lastly donors work yeah when I start talking about options other thing have become a little nostalgic because it is right about the time I will start starting my PhD as well so I have seen like hundred versions of options before it finally came into this form right so they talked about α models β models and then β time models and they have a whole bunch of different versions of options before a kind of crystallized into this right.

And this whole connection of all of these frameworks options framework HAM and max Q and all of these frameworks the connections to SMTPQ-learning were actually originally observed by Andy Barto so somebody was giving a presentation in one of the meetings and then and he just said oh that is SMDP's right and then people went and clamored around then they actually figured oh yeah okay he's right I mean yeah just in some sense many of the results that they were showing at that point were reinventing the wheel that people are done with SMDP'S so anyway so going back to HAM's I believe the ham framework was the first hierarchical RL framework that came out I think in 96 or something 97 I think but the options framework became more popular later and mostly the work on the follow-up work have been done in Stuart results lab and even run part did not continue to work on our hands after that but it is a really cool thing right and it is also very complex.

So what do I do with HAM'S is that I am going to define so what I call or simple machines that encode the policies that I am trying to follow right machines finite state machines they are like finite state which is I am going to define a hierarchy of finite state machines okay that is what we call the hierarchy the hierarchy and the machines part is all clear the abstract part will become clear in a minute okay so I am going to define hierarchy of finite state machines right so the finite state machines will have the following types of states four types of states okay.

So the first state will be what is called an action state right the second is called a call state third is called a choice forth is stop okay so what does the action state do takes an input emission action okay what does the call state do takes an input calls another machine right what does the choice state do so confusing thing right what is that exactly the choice state take well no choice have to

be a decision right so what do you do with that what do you mean variance it think of it in terms of finite state machines what could it be how will you encode decision making in a finite state machine and it is closed you may give a non-deterministic transition right.

So essentially a choice state makes a non-deterministic transition to two or more states in the same machine okay so a call state calls a another machine but a choice state translations to one or more states non-deterministically in the same machine and what does the stop state do you think stops and then returns control back to the machine that called it right so because machine would be called by a call state okay and that is also this starting machine right which is where you start your policy makes sense what is the input that you are going to see for these machines states right.

This underlying states of the MDP so the inputs to the machines are the underlying state soft MDP right the outputs on these machines are come on what do action states too that is the output primitive actions primitive actions it is going to take as input primitive states as input is going to output primitive actions right.

So the only output happens in the action state so the call state does not output anything what does the call state though it just transfers control to another machine so you keep going to one machine to another machine until you get the action state when you hit an action state you meet an action and that is when you actually move in the real world okay makes sense so this is how things will work so I have this hierarchy that I have this machine's well how does the hierarchy get established here the calls right.

So the calls will give me a hierarchy I am at to make sure it is a proper hierarchy I should not have loops that if I have loops then I can just keep calling around in the loop and they can get into infinite depth into my policy so make sure it is a proper call graph right so it should be a dag right so make sure my machines are connected in a dag right and then I start in the original machine I look at what the initial state is it that is the top-level machine let u s call it some M0 so M0 will take a state right.

And then the start state and M0 will now make a transition it could be the start state could be a choice state it could be an action say it could be a call state whatever it sees the input and based on the input it makes a call right so it could start off 11 machine or the other and then that machine now makes another call again finally some level in some machine I am going to get a primitive action if you think this is very complex just think about how we talked about the getting out of the going to a central station example right.

There is one big machine that is going to centralization okay it takes as input okay you are in the ICS are building right then it is going to say okay let me call the get out of inst I machine so I will go to the gate out of inst machine get out of inst machine is going to look at the state you are in the ICS are building okay go to GC machine then they go to GC machine you look at the state I will say you are in the ICS are building okay and then it will call get out of ICS are building and that will go down it will call the get out of the class room machine right.

And the get out of the classroom machine will say oh finally k more right that will be the first primitive action you do right so the all of this processing happens and then you move right and then what happens I am still in the get out of class room machine because that is not reached a stop state and I keep moving until I reach the goal I just do not want to exit and walk out to demonstrate the point but you see what I mean I as soon as I reach the door I will actually get into a stopped state in the get out of classroom machine.

It will pop up and go to the get to yeah memories so low what machine called it is our building right so that is the machine that calls this if you go to the get out of ICS are building now what it will do is yeah okay you are no longer in the classroom right we will see that then it will say okay go down the stairs machine late in the go down the stairs machine will then allow me to take my actions one after the other so it will give me a series of primitive actions I will walk down the stairs and then it will reach a stopped state.

And it will pop back up to the gate out of building machine and then it will finally say okay walk out of the door machine then it will go back and call that machine it will go back up and now I will make a transition in the get out of go to the GC it will go backup I am already out of the

building so it will go back one more level up and it will say a great now I can go to the GC it will call the go to GC machine and then given analogy there you go if the lift working the lift is working then.

I go to the I will go to an action that says take the lift if the lift is not working I will go to the action that is say step down that is a toy state no I am just I am defining go to the lift has a primitive action and it could be a choice state now it could be a call state the choice state can make a transition to a call state I did not see it has to make each other transition to the variety of state so the choice state I will actually draw simply so this is easier to connect to it whatever I described you can understand how HAMS are going to be used concern but there is also an example from the paper so how many of you looked at the recent advances in hierarchical RL paper.

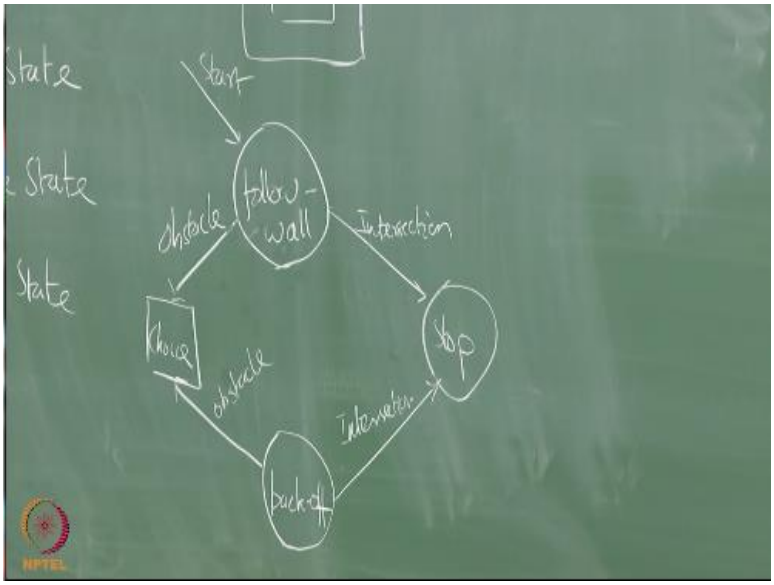
That has already been on its on model since the last class please read the first four sections you do not even have to read the 5^{th} section okay yeah I do not know hey this is off the record okay when you are editing you are supposed to cut this part of what I am going to say next not the one before the example should be that do not consider so when then so Andy and Sridhar wrote this paper right.

They were asked to write this paper on hierarchical RL because somebody wanted this nice survey article and stuff like that so Andy wrote the first four sections right he just did a fantastic job like I keep telling you right clearly the way he has condensed the all the different hierarchical flavors this is amazing right they talked about all this MDPS and then put everything on a same footing and so on so forth and then the editor is where after these people come on we need the final version of the paper send it to us send it to us and stuff like that so Sridhar instead of doing a proper survey of the recent advances in hierarchical RL this kind of concatenated all this work right.

The paper said his group has published right and then did a bunch of things on the recent advances in hierarchical RL so that is the section 5 right which is also the title of the paper that is supposed to be the crux of the paper but then he just did a very quick job of this comparing all

these papers together that is why I am saying I mean it is a very skewed view of what is recent in RL from section 5 on voice is basically what is recent from cedars lab in RL okay so for the lot of errors in the first version of the table that is because they were pushing them so ham I was there when they were writing the table they are pushing them really hard so what happened was they sent out a previous version of the paper to them even though they had a edited version when they finally sent the paper to the publishers they sent the earlier version of it anyway yeah.

(Refer Slide Time: 13:46)



So going back let us let us take another example that they have here so the problem was to do some kind of a very complex so they had a very complex grid world task okay each of these corridors will be divided so there will be like 10 grid cells here right this is a pretty large grid well right so some 300 x 400 grids or something like that and all along the corridor okay there were small obstacles like they in a specific shape there is like this frame like this they put obstacles all around the corner.

So whenever you are going around right so you may hit an obstacle and you are supposed to find your way around the obstacle you are trying to learn your way around the obstacle so what you can either say that you are going to follow the wall you know so if I hit the obstacle here I can

essentially follow the wall and I will get out right or I can back up go back the way I came turn into arbitrary direction and move to see if I elevate the obstacle is the two ways you can get around the obstacle.

And so they wanted to come up with a machine that learns how to do this right so what they did was they defined it as follows so they have a we have a follow wall 300 or 400 and not of it is repeated right and once you know how to avoid one of those that bracket shaped obstacles then why do you have to relearn everything again and again right there is a whole idea I am going to hierarchies and you do not want to keep running the same thing again and again so I am going to follow the wall right start somewhere I just keep following the wall then what I do right so if I hit an intersection so this is a call state okay.

So I am calling the machine for following the wall so there might be something that needs I need to do because if you assume it is actually a robot I might need to position myself a distance away from the wall and then keep moving in things like that so when the follow wall command the machine returns I will be in some state right so I could be either an intersection I am assuming that as long as I can validly follow the wall this machine will run right when this when it comes back to this machine right in the follow of all machine terminates and comes back to this machine I will either be at the intersection or I will be at a obstacle.

Now an intersection I stop basically I go back up because when they come to an intersection right now I have to make a choice of whether I have to go this way that way or this way right so that choice will be determined by a different machine okay so my job here was to walk from this point to this point that is the goal of this machine you can think of this as now we get a corridor this machine is essentially navigating corridors right so if I start it off here it will go all the way here it will stop here and it will tell me okay I am done and if you start it off here may be if you go here or here one of those two things you give it a proper direction you say go in this direction it will go all the way here it will stop and then it will come back okay.

So that is the goal of this machine it will have we get a corridor okay so it will stop follow wall will stop if it reaches an intersection because there is no wall here so it will stop it will say okay I

have reached the intersection and then if M reach the intersection I am happy I will just stop and we will give it back if I reached an obstacle then what we do I have a I have a choice okay so what can I do I can two choices I have I can pick a different wall and follow it right or I can back off these are the two things I can do.

So what is back off back off is another machine like is telling you it moves me back in the direction of K okay so this is remember this is not a real machine okay and this is a sample thing so the choice state I can either say follow wall okay or I can say back off or I can even say stop if you want I mean I do not care right so the choice state allows me to go to any one of these three things and what we do it back off so it is another machine I call it when it returns I either reached another obstacle or I have reached an intersection okay.

Reach an intersection I stop basically it could very well be that I went all the way back so that I went back to where I started right or I reached another obstacle so I came back and try to do another choice take pick and different wall and follow or whatever right so sorry what is that choice I make it work it just says choice this is non-deterministic transition so in this case it can go to any one of the other three states in the machine it could go to follow wall it could go to back off or it go to stop or you can limit it you can say no you cannot stop you can go to only one of the other two options.

(Refer Slide Time: 20:08)

That I can say that okay you can either go here that is your non deterministic transition so from choice you can either go to follow all or you can go to stop do not think of it as a realistic way of solving this problem please okay just a simple illustrative machine it might not be the actual machine okay any questions on this let me at least try and finish this today and then I can start max Q tomorrow so give me a few more minutes follow another machine just like a hierarchy that is what this call states do their action right there call states where does actually actual happen somewhere down there right.

So follow wall might say okay rotate my motor wait stayed in between this olive oil and stop all of us another machine like this okay so there will be a follower wall machine okay that will come somewhere like this right so maybe not a choice here no they did not have a symbol for action States will be some action it will say move right that might be actions for moving right moving left or whatever so you come here and then you decide whether to move right move left and so on so forth.

So there will be another machine with action states in there so this machine has no action states all fine everybody find on what was that the call states or follow okay so in this we have call

states choose states stop states no action states the action states will come further down somewhere which will be things actions like move left move right or turn motor so much depending on whatever it is your primitive action okay so that will come in some other machine we are not bringing about that here is my question to you where is RL here what do you learn here where is the SMTP here okay were different hieratical framework is no longer options here we should forget options when I am talking about hams there are primitive actions but where are my where is my SMTP here.

So what do what do you think I should be learning which machine to cover or basically what do I do in the toy statement toy state need not be machines it could be other action states also it could be I can come to a choice state and then say okay if I am here then go left if I am here go right I mean I can look at the actual the what they call it the core MDP state the basic the base MDP state basically I can look at the input in the choice state I can say go left go right I mean the code could be primitive actions also I could be transitioning to action states.

So I will have multiple choices that I could do in the choice state I really have to learn what is this choice I should make right but where SMTP that I am going to learn on where is the sm give me a specification of the SMTP so what are the states of the SMTP so if you remember in the options case the states of this SMTP you are the same as the original states right and the actions of the SMTP you are essentially the actions plus options I totally read metaphor option 4 actions and 6 options totally the SMTP will have ten actions right so likewise where is the SMTP in this case what is the state space.

The call states and the choice states no the underlying MDP is we can throw it away is it that does not form part of a syndicate so I need to have my base MDP states in my SMTP because I have to learn a policy that is a function of where I am in the world but I cannot ignore the base MDP state or the core MDP state I need the core MDP state but what action I pick also depends on what machine I am in right it not only depends on what machine I am in it also depends on one who called it right once you are outside the building here depending on whether you are going to the main gate or velocity gate depending on which machine higher up call down call the

gate out of building machine I would have to take a different action it does not just depend on whether I am in the gated of building machine it also depends on which machine called me.

Right in fact that is a stack I have to keep a stack I have to keep a call stack which machine called which machine which machine called which machine so that call stack has to be part of the state space because what decision I make depends on the entire call stack so now what are the things that need to be part of the state space the state of the base MDP and the call stack right grades and what are my actions for my SMTP that is the reason I did not answer his question I cannot answer this question until you answer my question what are the actions here reactions I will tell you primitive actions or not the actions that I am going to look at in this SMTP.

I will ignore the primitive actions in the SMTP because the primitive actions are all already predetermined if I land in an action state I limit a primitive action right I have no choice over that I'm giving you hints so what should be the actions transitions possible out of the choice state there for two of those actions will be follow a wall and backup there will be two of those actions that will go into my SMTP right once a call follow wall it is like an option I have called now it will keep running until it reaches a intersection maybe that might be other choice states within the follow all option in which case I will come back to me and I will have to make a decision what to do that okay.

So essentially the SMTP consists of I am going to call this as HI is the call stack like it is a history of the things I cause it is essentially the call stack right and si which is the core MDP state right these are the this cross this right so this will be my action space there will be the call stack times the current state that will be my state space right and my action space would be all the outcomes of the choice sate CI right so all the possible outcomes of the choice states CI will be the actions I can take right and of course there will be some limitations I can take a particular action only if I am in a choice state corresponding to that machine right.

So I need to know which choice state I mean so I not only need the call stack what else I do I need I need to know what state machine state I- choices the call state could be an action still provide it is something which I can reach to from a choice state if I have a machine that is

completely specified without any choice state none of the states in the machine will appear in my actions right I could define a machine without a choice state I mean I could know the policy completely beforehand I do not have any learning to do for a particular policy for example we have been talking about options right.

So if I am trying to encode the policy of an option it will be a machine without a choice state suppose I would defend an option get out of this room right the machine for corresponding to that will look like okay since the world right so okay that is a if you are in this state then back off right until you reach here and then turn right and then walk out right so it will be essentially I will be no choice states here this entire machine will have no choice states it is possible and none of these things will actually appear in the run of the states.

That I have in this machine will appear in the actions for the SMTP right way depending on where you are you can choose to go left or right depending on their constant good that is that is a it gives you the freedom for defining it that way. I am not talking about learning recursively optimal policies here I could learn hierarchically optimal policies and I amusing HARMS it is pretty powerful frame allows you to do all kinds of things the options think that question does not even arise because I have frozen my hierarchy.

We are talking about this in the last class also it kind of becomes degenerated the recursively optimal is equal to hierarchy t optimal in the options case because I cannot change the policies we have to assume whatever policy is given to us is optimal but here I can actually if I act if I start conditioning it on the call stack and they can potentially think of learning hierarchical optimal policy on the train so I need the machine state also.

I need to which choose state I mean there could be multiple choice states in a in a machine whether I need to know which choice state I mean also and so I take all of this together and then actions are essentially other machine states and right so SM or machine states so actions are this machine states it could be one of these states that I have to go to right so this is essentially my SMTP and the transition function of defined by okay if I choose to go from here the action I take

is follow wall then that machine will call it will keep running till the end of that machine's execution and then it will come back here okay.

So that is essentially the our transition SMDP states of these finite state machine states one of these machine strains which state I am here right when I actually come here which is the state I am in when I make the call now that is the machine I mean this will be the machine state I am in so which I which is the path through the three would be there the Dagwood be that and this will be where in this machine I am okay so I mean I might be kind of fudging the notation a little bit you can read the paper and find out what the exact rotation is but so that is I am just trying to give you the intuition not trying to knock down lock down everything.

So one of the things you wanted to tell you and how would know now that you know what are the states right in what are the actions we can learn all the policy you see SMDPQ learning's right so what is the weird thing about this what this is another thing I wanted to mention so whenever I make a call action right or a choice action right whenever I go to a call or choose state there is no transition in my core MDP state right only the Machine state and the call stack could change.

So the only time I like or MDP state changes is if I visit the action state anywhere in the in any of the machines I do not care right whenever I am in the process of executing the policy if I reach an action state then right the core MDP state will change otherwise that component will stay the same right and only these components will change okay so this is I should kill idea right and so the thing about this is so what will be the equivalent of an option in this case.

So as option in encapsulation already learnt policy right so what is the equivalent of an already learnt policy here the third any choices is what without any choice takes any sequence of action states you listen without any choice state in between right I mean those would constitute your policies or an execution of the policy because when you are running this machine if there is a sequence of states that you visit sequence of action states you visit without a choice state in between that will constitute a learnt policy make sense that is there is already prepackaged I have

no choice there that is like an option once a pick option I just have to take the action suggested by the option so like that.

So that is the thing so as you can see this is a little more complex than the options framework right so people are not ready it did not take to it too kindly but if you think about it is very powerful framework it allows you to work in partially observable states right so when you do not need to know the full state spaces because one of the nice things is you can use this kind of finite state machines to define more complex policies right you can define non Markova policy CD you can have finite state machines with his memory right you can define on Markova policy so the representation itself is very powerful but unfortunately this gets very complex so people are not very fond of this I will stop here I will continue with the max Q thing and also more discussions on hierarchy specific abstractions and other things.

IIT Madras Production

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

www.nptel.ac.in