NPTEL

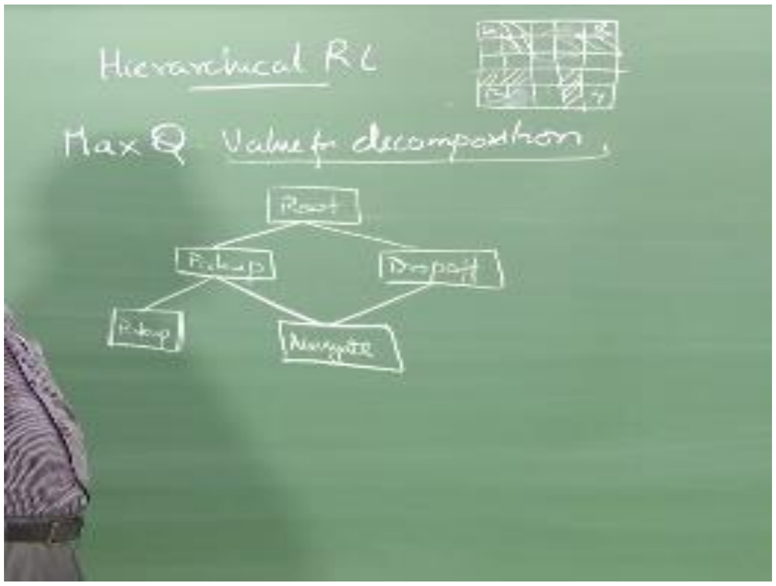
NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

MAXQ

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

(Refer Slide Time: 00:13)



Let us start looking at hierarchical reinforcement learning so we looked at several things one the need for the hierarchy SMDP's and then we looked at different kinds of optimality recursive and hierarchical and so and so forth and the we looked at SMDP Q learning we also looked at options entropy Q learning viral then amps and then SMDP Q learning applied to amps right and then we look at the third or the three popular hierarchical frame works.

And so this is called the original proposed frame work itself is called is called max Q value function decomposition right it is not called the Max Q hierarchical R frame work or anything like that the original name that Tom Detrick gave it was Max Q value function decomposition, so when I normally teach this course right when I normally teach MAX Q I leave out the value function decomposition part right.

So because that is a I mean that is the value to be had in the MAX Q frame work without even considering the value function decomposition, right so but I'm going to cover that today, at least very briefly at least for the benefit of two students here who are trying to take advantage of the value function decomposition in their project but in general I think there I something to be said in this side of deplaning and other things to actually think about value function decomposition also right.

So I am going to talk about that and I would urge people to Actually go and read more on MAX Q in fact when I was a PhD student one of the things that so aunty told me was that the MAX Q journal paper the journal version of the MAX Q paper is like an ideal journal paper right, so this is how journal paper should be written right in for CS I mean this is how journal paper should be written and so encourage all of us to read the paper just for the sake for you know knowing how to write paper how to write good papers you know as a practice for that right, so I will encourage people to read the MAX Q thing.

So do we have a T here pseudo TA yeah can you just remind them to put the MAX Q journal paper to the JR version of it, right I did not ask them to put it online right and the catches actually is a very well written it talks about it essentially it almost like this like a mini PhD thesis I mean it talks about all those issue it introduces all this recursive and hierarchy optimality everything has been was introduced in that paper.

The whole concept of recursive and hierarchy optimality and then it talked about the whole set of other issue to do with hierarchies in general not just with respect to the MAX Q frame work of course all of us can probably at best aspire to write one set journal paper like this in our research life times so it is an ideal that you keep up that but do not try to emulate it every time because it is like he write one entire thesis master thesis this is when he was a faculty, right.

He is already a very senior faculty member at that point of time he write this one journal paper all become himself no students nothing involved right and if you look at it, it almost like another PhD thesis an then he published it as one journal paper okay, so that is not ideal strategy for people looking to graduate, so I'm just minute, so even though it is a fantastic paper I encourage people to read it.

And take away take some good messages on it okay great so what does so we will come to the like I said we will come to the value function decomposition part little while so let us talk about the MAX Q architecture itself right, so the basic idea here is that I'm going to decompose my problem okay into set of sub tasks right, so I will illustrate this by the example that he used so he looked at something called the taxi domain.

So the taxi domain is like a simple 5/5 grid world not a very complex grid world right and then they were like GRY and D or something like that and then there were some obstacles and this arbitrarily putting the obstacles here, right so there were some obstacles like that you cannot walk through so if you want to go to R from here I basically have to drive like that okay. Right so there were some obstacles that kind of make that navigation hard.

And then your job is to control a taxi that moves around in this grid world right the taxi can go up down left right okay the normal grid world dynamics but part from that the taxi can also pick up passengers and it can put down the passengers right, so it can pick up the passengers only at one of these four places RGBY likewise it can put down the passenger only in one of those four places RGBY.

In other places it cannot pick up or put down okay, so for every move it makes it gets a -1 right and for every passenger it delivers successfully to their intended location it gets a some rewards say +5 or something right for every move it takes it gets a -1 for every passenger a successfully delivers it gets a +10 so as soon as you deliver a passenger randomly another passenger will come up in one of the other locations.

You have to go pick that passenger up and deliver it just keep continuing say you pick up deliver pick up it could not guess again and again okay so if I try to pick up a passenger from a location where there is no passenger waiting right then you get a penalty let us say you get a -2 this is to prevent you from just keeping on randomly picking up passengers so if you try to pick up a passenger where no passenger is waiting.

You get like a -2 and it drop of a passenger when where the passenger does not want to get down you get like a -5 rite some so passenger gets into your cab and you know taxi in hour and he says he want to go to B then he go to Y and ask the passenger to get done right a valid -5 is a small penalty right, and the passenger will also will not get down is not like you can dump in throw him out at Y right.

So passenger will not get down until he go to B and not the passenger it is a simple thing, look at all kinds of variation son this juts to standardize and four stall and equations about what will happen if it try to pull the passenger of somewhere else so I am giving you all the examples right all the conditions, so this is the problem and then you have to learn to solve this,, right so yiu can immediately see that there is a lot of repeated sub structure here right.

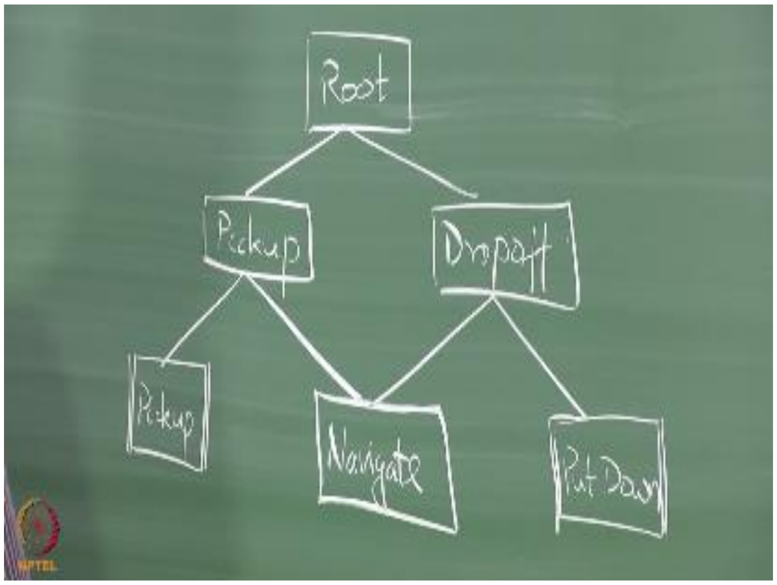
Essentially I can describe the problem to you as pick up a passenger I mean go to the place where the passenger is pick up a passenger go to the place where the passenger wants to get down drop the passenger, right so I can that is basically the policy I just have to keep repeating this over and over again but then I will have to do a whole bunch of things so what does it mean to say pick up a passenger and then drop of the passenger right.

So that is what I am going to decompose this problem as,, so I'm going to have a route task that is going to keep doing the same task again and against o the route tasks is the one that I will initially execute right so the route task is going to have just to sub task pick up the passenger drop of the passenger that is all you have to do so what we are supposed to do now, go pick up the passenger.

What I'm supposed to do now, go drop pf the passenger that is basically what I have to do right, so what are the things pick up can do pick up can well pick up okay can pick up when I pout this double things let us say there is a primitive action okay it is not going. So pick up is actually an action in the MDP rig tans then what else it need s to do, it needs to navigate to where the passenger is before it can pick up, correct.

So I might be here and I will say pick up the passenger from blue so essentially what should I do then, I should first go to blue and then execute the pickup action right if I try to execute the pickup action here it will fail and so likewise what about drop of, Navigate again it needs to navigate right it figure out to go to the destination of the thing and then do a the opposite of pick up which is but down.

(Refer Slide Time: 10:18)



Right anything else navigate we have to implement navigate also rise so navigate we need to tell navigate what are the things it do so this is kind of like a task graph right this is like a call graph in enough you think about in a program that if you can write a call graph so there are different subroutines at your calling and which subroutine calls which is then you can basically have a call graph right.

So this is like a call graph and so is a nice thing about it is it basically gives you this structure then it is very clean to understand like in hams it was little confusing to main you could take the hand the correction of hams and then draw a call graph also but here you draw the call graph itself directly so it is little easier you know interface if you will for specifying what the hierarchy should be right that for MAXQ is it is kind of more intuitive then HAMS to handle and there is a couple of one the small thing I need to point out here so navigate essentially is not a single sub task.

Right so it needs an argument I need to tell it back to navigate to right so once I decide I am going to pick up some body from B the I will say navigate to B once I decide I am going to drop of somebody R then I will say navigate to R so it will take an argument right so one way to think of this argument is have 4 locations here GBY therefore I will have 4 different sub tasks one that says navigate to R one then navigate B navigate Y and navigate G right so I am going to have 4 different sub tasks from this.

That is the one way of thinking about it in fact that is not a bad way in this particular instance because they are different problems it is not like I can learn something from navigating to G that I can use for navigate into R may be I could but this is a small real small word right much larger world you might think of sharing knowledge of cross the different navigate positions but now you do not have to worry about do you just think of this is 4 different navigates okay.

So any questions about this set ate right so this as to be necessarily a dag because if it as any cycles in it then your domed that is what you think about it direction of arrows is like this right root can call pick up and drop of pick can call pick and navigate drop of can call navigate and put down navigate can call north south east west after what action we will come through the structure of these things right so once a primitive action is executed you just emit the primitive action you go back this right.

So now the question is from pick if I call navigate when do I go back to pick up or from root if I call pick up when I do back to root right I mean I can in this particular case you can say that but how am I defining it in the structure right so that we will have to look at obviously the navigate has to go back after it has to reached B suppose I call navigate B after it reaches B it has to go back right.

Or suppose I call pick up right after the passenger in the taxi it as to go back right likewise when I call drop of after the passenger as successfully left the taxi it as to go back right I keep calling this again and again and say one thing which I want you to notice is that I am having one of those fandom] phone rings I guess felt that you feel that the phone is ringing but you do not have anything I your pocket phone is there no apparently they have actually beginning to figure out.

That it is a real neurological condition that people are pashing this fandom vibration in the like they just think that because they are so used to carrying the phone is around in the pockets so I they suddenly feel that the part of the tie where the phone is contact with they actually feel that vibratory signal this is so they actually began to observe this elsewhere as well so I am not crazy I have good company so the thing that you notice here the way I have draw these books right.

Does not near necessarily imply any ordering for example in the pickup task I first have to navigate only then I can pick up first and then I navigate right so this order is not like left to right ordering anything like wise in the drop of task fist I will navigate the I will put down so this is the order is not any indication of the actual sequence any indication of the actually sequence in which the actions will be called okay so what does this the laying imply it just say that whatever I solution I come up from the pickup tasks I can only use navigate an pick up as my actions right whatever solutions I come for the drop of task I can only use navigate and put down as the action so that is all it implies okay.

So do not read anything into the order in which we are dawning the box okay so to specify this MAX Q hierocracy so what do I do I take my original problem right and I am going to break it down into a set of sub task so right I am going to say that I have a original problem M so correspond to some MDP I have my original MDP so I am going to break it down into M_0 M_1 after some M_k right so I'm going to break it down into k sub tasks right and then each of the sun tasks right I am going to defined some quantities just like we did for options.

Right for each of the sub task I am going t defined some quantities so for each sub tasks sub task Mi is going to have what it will have it will have set of actions Ai right so that we can define exactly here so let me properly follow that notion yeah so each sun task is defined by this people rights so where IS the set of states in which the sun task is active right the essentially these are the starts in which the sub task can be running so for example for the pickup task right what you think would be the set of states which the pick task can be excited.

All the slow that taxi should not have passenger if you already have a passenger if you try to pick up another passenger that's I not allowed right so the states of the taxi world should not only be location of the taxi but also whether there is a passenger taxi or not in fact they can defined it other it as the location of the taxi the location of the passenger right and how many values can the location of the passenger take 5 will the passenger could be in RGBY or in the taxi right if the passenger is in RGB or Y then the pickup Action is valid but if the passenger is in the taxi then the pick action is not valid right likewise.

How many values can the location take of the taxi all the value 25 – 8 so that many states that taxi can be in okay so there is anything else I need to for the state what destination of the passenger right I need to know where the passenger wants to get down so how many value that can take that can b only 4 not 5 okay so that is only 4 RGB or Y so these are the state variables location of the taxi location of the passenger and destination of the passenger okay.

And so there are punch of thinks that I can execute the pickup action on of the location of the passenger and the location of the taxi is the same right or I can execute it successfully right. I can still try to execute the pickup action anywhere where it might give me a penalty right. Likewise what about navigator option, when you said valid any time navigator is valid at any time okay that is fine.

And what about put down, they are all primitive actions, so they do not really do not worry about them a sub task, but you can also define them as sub tasks right. So wherever the, now you can try to put the passenger down anywhere it is just that he will hate you right, I mean and you will get a -10 or something right.

So you can try to put the passenger down anywhere that is fine right. So likewise, so you can think of SI right, what is AI give you, AI tells you what are the actions that are available to you right. So AI would be a union of, would be a subset of the union of all the Ms and the primitive actions right.

So basically Ai can take from M_0, M_1, M_2, M_3 up till Mk or N, S, C, W pickup and put down. So I have six primitive actions in my MDP so anyone of those six, any subset of those six primitive actions plus any subset of Mi+ …. MI_k, M_k I am going to assume that I cannot call things that are numbered earlier right.

So you have to think about how will you do the ordering to enforce the dag structure right. I need a dag structure right, so I cannot have any task called M_0 for example right. So I cannot have any of the sub tasks, I cannot have them call M_0 as my, as an action, because M_0 is a root action, that means I will be setting up a cycle.

So one way of enforcing that you do not set up a cycle is to say that the Ai cannot consider any Ms that are numbered below Ai right. This is one way of thinking about, but you do not have to necessarily name it in that order right, you can number it in any order you want just make sure it is not a dag, it is a dag sorry, make sure it is a dag.

And here is a interesting, what is this TI, RI hat, RI bar it is a reward associated with the sub task okay, it is a reward associated with the sub task right, it is called the pseudo reward. It is called the pseudo reward and so for example, if I want to navigate to G right, I will get a pseudo reward of something when I reach G right.

Is there anything else that I would need here, I will also need to know which of these states are made terminal states right. So where do I end the sub task right, so anything there is outside the SI will be terminal, I cannot run that sub task there right. So under the definition, so when I say navigate G it is possible only if my taxi is not a G right, because G is a terminal state right.

So one way of thinking about it is that SI gives me the active states where TI gives me set of terminal states or I could define TI and whatever is not in TI gives me the set of terminal states, I mean set of active states or I can specify SI and say whatever is not in SI is a set of terminal states and which of these are desirable terminal states that is specified by the pseudo reward right.

So for example, when I start running pickup any point of time the passenger vanishes, any point of time the passenger appears in my taxi I will end the task right. But if the passenger had appeared in my taxi only by virtue of V executing a pickup action at the location where the passenger is that is a desirable thing for him right, not from any other way of getting the passenger on the taxi right.

So when he contrived example here, but that is basically the thing. For example, we can define navigate in a slightly different way right. So navigate is active as long as I am not in RGBRY anywhere else navigate option works fine right. So that means navigate has to terminate if I reach 1/RGBRY does it make sense right.

So navigate has to terminate when I reach 1 of RGBRY, but which one of them is useful for me right, that is determined by whether it is navigate B or navigate Y or navigate R, if it is navigate R, if I terminate in R I will get a reward, pseudo reward of say +5. If I terminate in GBRY I will get no reward right, I will just get the -1 cost that I paid to go all the way to BYRG right, does that make sense yeah.

So in that case you have to run the same task again or the new source and new destination right, sure. So if you deliver the passenger to the wrong destination yeah, so you will had at the same sub task again with virtue arguments. Yeah, so if we can define navigate as ending at only the successful termination state then it can lope in 19 it reaches.

No I did not care right, to think about no, to even if I, no there is a two things here. So even if I define navigate to end at the successful state, navigate does not know where the passenger is going right, navigate is told to go to B, so the drop of task might have made a mistake, the passenger might want to go to G, but I am tell the navigate function to go to B which case it will anyway go to B and you cannot penalize it for going to B, because that is what it was told to do right.

So that is the different issue, so what the problem you bought up is a different issue, that the navigate is going to the wrong place right. But as long as you say navigate B and it reaches B I will give it a reward, if I say navigate B and it reaches G then I should not give it a reward right. I can keep it running until it reaches P that is an option I am just saying give it s negative reward, I mean give it a positive reward only when it reaches B to tell you the use of the pseudo reward function versus the terminal states right.

If you define the terminal states to be exactly those states where the reward function is 1, then there is really no difference between the two things. I am just saying that you could define multiple terminal states and then you can use the pseudo reward to distinguish between preferred terminal states and non-preferred terminal states.

Taxi is navigating and the drop off function has be the drop of task has pseudo made, so the entire cumulative V_1 is my RI'. No, no RI' is only for the function, for that function MI that is all I am solving. So when you drops off correctly you get just +5, for that task. So it is not +5 for the whole problem, see I have to learn the policy for navigate also right.

So for learning the policy for navigate I will use that +5, but for solving the whole problem for the value function for drop off or for the value function for the root I will not use the +5. For every step it is acquiring -1, for every step it is acquiring -1, that will be present in all the sub task, because s you see code MDP is reward that I will not touch, that will be available in all the sub tasks right.

But the -1, the +5 sorry will be available only in the navigate, the pseudo reward will be available only in the corresponding sub task okay. So I will not be using that for the higher function, we will come to that in a minute right. So max is little bit more complexity in the learning part of it, because of all this pseudo rewards and other thing.

So far, so in both the options case right and in the ham case you are eventually, even though if we are talking about hierarchies and other things you are eventually learning on a single MDP right, you are eventually learning on a single MDP, in the max Q case you are learning on a collection of SMDPs, you are not learning on a single SMDP like in the ham case in this.

Each one of this is actually an SMDP so M_0 is an SMDP, M_1 is an SMDP, M_2 is an SMDP every one of this is an SMDP and what you are trying to do is collectively learn a solution for all the SMDPs in one shot right. So it makes it a little bit more complex as well as the learning goals right. And so the pseudo rewards is used for the solution of the corresponding SMDP.

So if RI hat is there, that will be available only for solving MI. So there will be a reward function R corresponding to the original task M and that is available for all the sub tasks. So M_0 to Mk we will use R right, any MI along the way we will use only RI hat okay make sense.

IIT Madras Production

Funded by

Department of Higher Education

Ministry of Human Resource Development

Government of India

www.nptel.ac.in