NPTEL

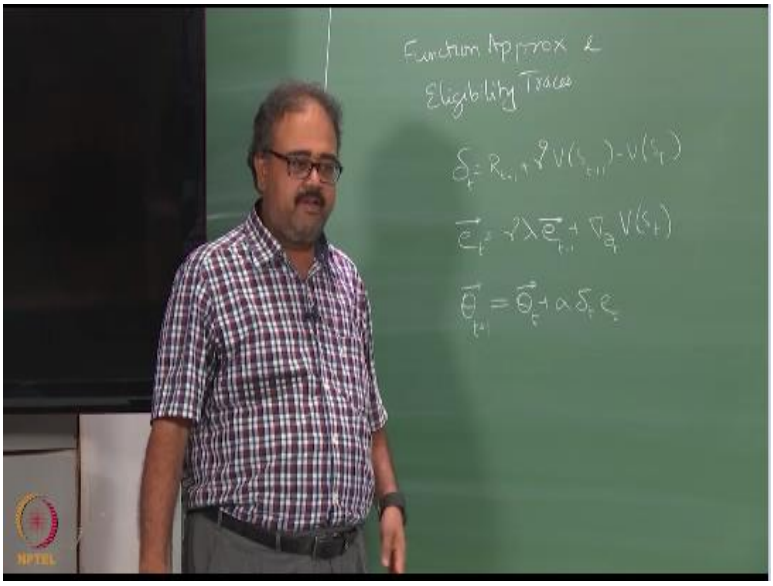
NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

Function Approximation & Eligibility Traces

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

(Refer Slide Time: 00:14)



So we have to look at function approximation and trace function approximation eligibility and traces. So what you think would be different when we do function approximation. As we remember all of we remember eligibility trace how did we do eligibility trace. Let us not worry about the forward view because we are only interested in the backward we have.

So what did we do in eligibility traces so essentially this is a record of all the states that you have seen so far right. So what do you think we should do when we are looking at function approximation, what you mean by parameters θ of the state, θ is not dependent on the state right.

Yeah, so we have ϕ times θ right, so ϕ^T same θ if it is linear so some function that translates state into a representation which denoted by ϕ and then we have θ right. So should the eligible key trace be on the ϕ. Earlier we had eligibility trace on the S right, should the eligibility trace be on the ϕ or should it be on the θ, ϕ how many of you say ϕ 1 1 and and half okay.

You either put your hand up or down do not put it here like this okay 2, 3 how many of you say θ 1, 2 I shall we cannot put answer for both he put it up okay he did not put it up for ϕ is it, θ okay. So why do you say θ which entity right needs its value to be updated right. So one way of thinking about it is that you are going to attach it with the ϕ right but then you will have to worry about how the ϕ figures into your update.

Suppose it is a linear function approximator how will the ϕ figure in your update right it is ϕ times the delta, it is ϕ delta right. Now if I put eligibility trace on the ϕ it becomes ϕ E delta right, so is it fine. What do you want to do something else if you put it on θ then what would happen to it. So for if you are looking at the linear case okay, it really does not matter whether you are tracking ϕ or whether you are tracking θ right.

They are almost interchangeable right in the linear case ϕ and θ are all fine you can either track ϕ or you can track θ, the interesting part comes in when you are having nonlinear cases where you will have to compute the gradient with respect to θ right. And then you are going to, I mean update the values of θ based on the gradient that you have completed right.

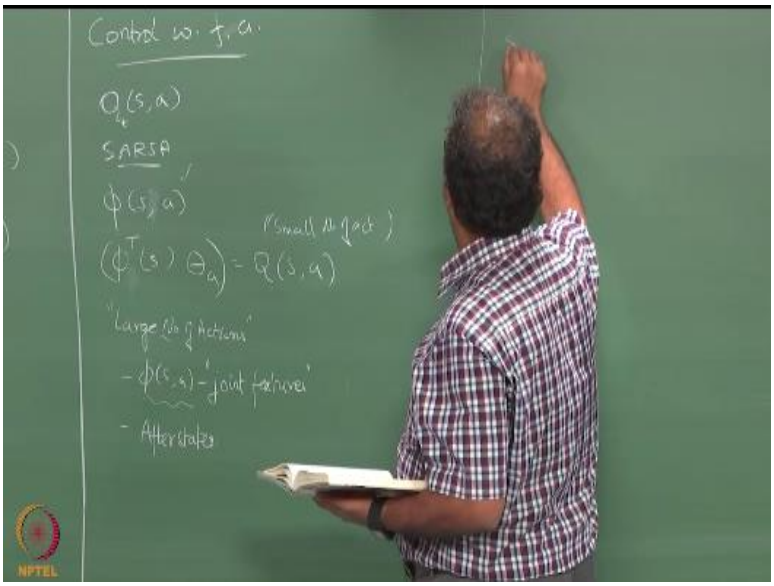
So what does the gradient tell you forget about the eligibility traces what does the gradient tell you in a normal function approximator people know this right another way of thinking about it is how much is this particular θ responsible for the output. In some sense that is what you are trying to do with eligibility traces also right how much is this state responsible for the final reward that I got right.

So it turns out that the right way to track eligibility here okay is not to just remember one for the state well I mean bump it up by one if you encounter the state again and so on and so forth. But

for every θ keep track of the gradient I heard gradient from somewhere yeah who said gradient okay. Yeah keep track of the gradient as before your Delta is right might think about it if I use a linear function approximator so what will happen to this ϕ this will become ϕ.

And if I had use the lookup table representation for the linear function approximator I remember I said an indicator function right. So ϕ is a indicator function it will be one for that state that was visited 04 it just exactly reduces back to our original 3D lambda case right. So it will be +1 for that state that I visited 0 everywhere else. So this is the accumulating trace equivalent for this okay.

(Refer Slide Time: 06:27)



So let us do control with function approximation right. So what are the things we need to be careful about, we have to take care of when you are doing controller function approximation. So let us start so you have to do Q(s, a) you have to keep track of Q(s, a) and then you can do SARSA you can do Q-learning and so on so forth.

Let us take SARSA, SARSA is alright doing SARSA with function approximation. So what are the things that we need to take care of before their itself see we are ϕding states and actions into

the function approximator and we are going to get the Q value outside right. So what would that mean when I wanted to ϕd the state into the function approximator we thought of an encoding for states right.

Now I want to ϕd states and actions into the function approximator I need to think of an encoding for actions as well. So that is the first thing we have to think of what I need an encoding for the action that is well right. So instead of looking at ϕ(s, a) right, but then often if you think about how the actions are right we end up working with a small set of actions right we think of all the grid world is not south east west right.

So the number of actions tend to be typically small right and also all the MDP is that we work with also have small connectivity right so each mdp does not go to 100s of other states right from each mdp state you can go to only a few other states. So these things are more locally compact and you would not have too many actions right, in that case what you can do is you could do not have to explicitly encode the actions here you could just have different set of features for different actions.

Alternatively you could have the same set of features for different actions but I could have a different set of weights for each action. So this is essentially like saying that suppose I have four actions I will have four different function approximators right, but they will be trained in an interlinked fashion.

So what do I mean by they will be trained in interleague fashion do you remember the TD error for SARSA right, so I will have to use Q(s', a'), so when I do the Q(s', a') I will use the θ A' network right, I will use a θ A' representation but for Q(s, a) I will use the θ A. So that error term I will get I will use for changing θ A right.

So I will be using θ A' for changing θ A, so there will be trained in an interconnected fashion. So when A changes the error for A' will change and so on so forth, so this training is being interconnected fashion but representation which storage which they will be stored as four different networks, four different whatever right, four different approximators right.

So that is one way of thinking about it this is fine for a small number of actions right, so we have large number of actions how will you handle it or as SARSA was mentioning if you have continuous actions like cycling we talked about cycling recycling has continuous actions right or any kind of control algorithm have continuous actions and or think of go what happened to game 5 we did not look at.

α1 still one yeah you have to hang on to hopes wherever you can find it yeah 4, 1 I need to read up, have you been reading the analysis of his games anyone, slightly if there are people I mean you do not have to do not read the go guys analysis, but there is a website called go growth okay, he has given a very nice, he talks about this thing he in fact in game three commentary he said there was a one point and he started physically ϕ ling sick.

Because α go played so brilliantly either I have been following game 1 game 2 it was not so bad but game 3 there was one point where α go made such a brilliant move that he physically felt sick thinking that this is the last resistance for human beings mean last whole lot supremacy from human being slipping away because the move was apparently so good.

So a certain nice analysis of all the games right so it is called go grew so that is good I do not want to go and read what he says of wood came fine. Anyway suppose you have lots of moves like go right do you think it is good to maintain separate function approximator something wrong there right why still doing generalization across the states yes.

So if you think about all kinds of navigation problems you have been talking about going north and south may not have anything in common right I mean they are probably very different things it it makes sense to maintain different θ right I will come back and ask equation where it does not make sense even in that case but in cases like go where no actions also have some amount of similarity between them right.

It really does not make sense to not generalize across the action domain also right. So in which case how will you go about doing it yeah, so you actually do this right anything else people do

not answer this rock correctly I am going to leave right. So that is another way of the finishing function approximation I am control where you do not actually represent the what we normally think of as a Q function right.

But if you stop and think about it values of after states are in reality Q functions some form of Q form because is their values of state action pairs right values of after states are actually values of state action pairs and you are using your after state as an encoding of a state action pair implicitly you are doing that. So the couple of advantages of using after states one it allows you to generalize across later actions that if the actions cause similar changes right.

Then similar right not same why similar, similar include same I mean this is more than same right in the in the dynamic programming case it was same changes here at the similar changes because I amusing a parameterized representation for the after states as well right. So after state will also be represented as ϕ of something right.

So similar an after states will have same value so it is essentially actions that give you similar effects right even if they are from different states and if they are going to land up in a similar kind of an after state then you are going to have the same value for them right. So this allows you to generalize across actions in a nice way second thing, what is the second advantage.

So think about how you use it normally right so you looking it up for from the state, action right so it is not like I look up look at the after states and try to figure out what state action generated it right I usually query on a state action pair and from that it is only a forward computation I do not typically need the reverse transformation okay.

So what is other advantage, yeah what does advantage there is a computational advantage no such conditions. Exactly so what does it mean fewer parameters estimate right so for queries memory is a non-issue a few parameters estimate memory is a non-issue because I mean memory is cheap now anyway you are not going to be that much worse than what we had earlier so that much better off right.

So it is fine so fewer parameters estimates see the more the number of θ is that you have to estimate right the more data that you need right so more number of trials and so on so forth so computationally it becomes easier if this is a few θ vector is much smaller okay. So these are the advantages so we have a large number so I put everything in quotes right because we do not know what large means right.

So how large is large for could be large right I mean even with four actions you can think of using after states it might work right or it could be continuous in which case it is an English question I said large could be 4 right as ϕ was four and after states might be useful thing or my action space could be continuous in which case number of is not an appropriate term to use okay.

So that is why I put everything in quotes just this tell you that I put things in quotes for some reason so so if it is a continuous state I cannot enumerate it right so saying large number is not does not make sense anymore right. So if it is a continuous state space well can you use after states yes, possibly yeah okay yeah yeah that is a variant of it depends so I like it go ahead why why possibly.

Exactly as long as you have a forward model right that tells you okay here is a deterministic component to this right so it has to be a deterministic component for it to depend after state which a stochastic component then it becomes a little tricky right. So if you have as long as have a deterministic component to a dynamics and you have a forward model that describes it you can use after states does not matter with the discontinuous the state space can be continuous, your action space can be continuous or discrete does not matter you can always use is provided we have this deterministic forward model okay.

So remember what distinguishes after states from you know state right what does even after state from state is that you are assuming that there are deterministic move and natures moves right the stochasticity in the world can be split up into or the transition dynamics in the world can be split up into a deterministic component plus a nature component is something happens like the cows right.

So you selling cows is a deterministic component right and the cows, the cows breeding or going older is a stochastic component right. So you can split it up into two components like so suppose you are looking at any kind of a control system right, suppose I want to move this by 3° okay I can move it by 3° that is a deterministic component and then there is a 0.5° noise that gets applied to it after the deterministic component goes right.

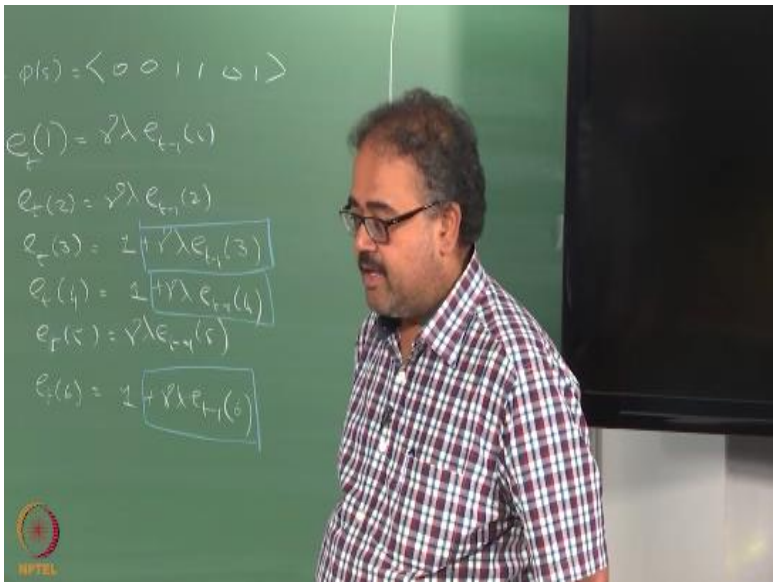
You could think of it that way whether that is a valid model of your system is something you have to think about or in games it is very clear so you make a move deterministically then your opponent makes a move stochastically. So this opponent move is stochastic because you cannot read his mind right.

But if you have a good enough opponent model that stochastic anymore what if your opponent is inherently a deterministic player and then you have this model when it will be easy okay for you to introduce that you add the state of the opponent into your state representation right if you say that okay I know what this person is thinking now okay.

Then that becomes everything becomes determined so whenever I make a move I know what the opponents response will be also right. So I can actually plan far ahead into the future and then then do it expect okay that is in a sight nothing to do with this. So when you have lots of actions so when you do control with the function approximation you can either do it like this for small number of actions.

So this means I am going to maintain a different function approximator where A is the size of A right for each actually different or they could do this right. So anything else you can think of any other way you can handle this have you used something like this before we come across this kind of joint feature encoding of states and actions before contextual bandits right. In contextual bandits we have actually looked at joined encoding of states and actions right. So it is not something new so we already seen it before okay.

(Refer Slide Time: 20:27)



So now move on to the next thing that we should be looking at which is anticipate mean right that is it, nothing more fancy than that okay. So all of these are accumulating versions if I want replacing versions of the traces what will I do, and stuff I will not do that is it just add the gradient to that, whenever the gradient is nonzero when do I do that, this is not on ST right this eligibility trace is defined on each θ right.

Oh! By the way we certainly do not want the θ to be the size of S it will only be the case when you are using a lookup table to presentation right, the indicator variable representations if I am going to using anything more compact the size of the eligibility trace will be the size of the θ vector not of the number of states okay.

I have kind of abuse the size notation there, but I fiduciary guess and understand what they mean right the first two size notations are about the length of the vector the last one this is the size of a set, the cardinality of a set okay. Yeah number of conference this is what I said by abusing the notation ET is number of components in ET will be the number of components in θ not the size of the state space right.

So what will be the replacing version of this, it turns out that in general you cannot define a replacing version of this okay. So you can do this for very specific cases where your feature encoding assist that you will have ones and zeros only it not necessarily be the lookup table representation right it could be some kind of a indicator function any kind of indicator function, it could be tile coding right it may not be a one of any encoder it could be some K of N encoders right.

But then it is course coding or tile coding or anything, but the point is my ϕ will either be ones or zeroes right. So whenever my ϕ is one I will replace this to one right, whenever my ϕ is zero I will just decay this does it make sense. If my representation this is that for every state right take some state S okay and my $\phi(S)$ something like this okay.

So now what will happen is my E(1) will be this is for replacing traces if it had been accumulating traces this would have been everything outside the blue box is for replacing traces everything including the blue boxes are accumulating traces right, this is fine assuming that my ϕ binary representations like that right.

If it is not a binary representation say it is 0.3 somewhere 0.2 somewhere thing you could think of saying that whenever the thing is 0 right just do the discounting, whenever it is nonzero initialize it to that value right. But that becomes a little tricky, because you are even if there is a very small sensitivity say 0.03 or something like that you will essentially erase the entire hold at trace and replace it with 0.03 this is a bad thing to do right.

So we do not want this so if it is like this it is fine right and in fact we looked at a lot of examples in the last class where the representation was binary right, whether it is course coding, whether it is tile coding right all of those things had binary representation. So we could essentially or state aggregation we looked at state aggregation, course coding, tile coding all of those they anyway give you binary representation in such cases you can use replacing traces.

And otherwise you use accumulating traces okay, so any questions good. So I am actually going to stop here and I am going to get back to function approximation next class I am going to talk

about a bunch of other ways in which people have been using function approximation for RL right. No I am not going to talk about deep learning it, but I will eventually get there right.

So maybe next class of the class after I will do that right. So next class I will talk about D squares methods which people have been used using quite widely and more and more it is becoming a standard tool in your reinforcement learning repertoire right. So I would like to talk a little bit more about D squares methods and then we will go on from that talking about other ways of doing function approximation okay.

IIT Madras Production

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

www.nptel.ac.in