NPTEL

NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

Function Approximation

Prof. Balaraman Ravindran

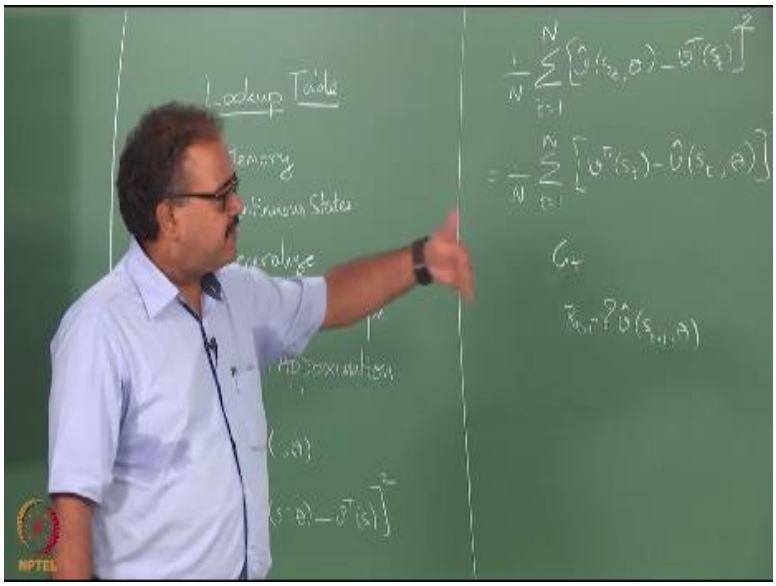
Department of Computer Science and Engineering

Indian Institute of Technology Madras

So far we have been talking about mostly value function based approaches for solving RL problems right, so remember that Monte Carlo or TD or whatever right so it is been value function based and there is a very small fraction that we looked at we looked at policy gradient methods and there was a reinforce algorithm we looked at, but we never talked about how would you use policy gradient for solving the full, full RL problem right we only talked about in the context of the immediate RL, right.

So I will come back to that at some point I will talk about policy gradient methods for the full RL problem right, and perhaps before that I will do a little bit of average reward reinforcement learning before we get onto that, right. But except for the policy gradient part right, and the linear bandits it is except in those two circumstances we have essentially been using some kind of a some kind of a

(Refer Slide Time: 01:17)



Some kind of a lookup table representation for our value functions right, whether it is a bandit case where we kept a q(q) right, or when we talk about V(s) or V of when rq(s,a) right we have always been assuming some kind of a tabular representation right, for every s I know what the value is right. So I have been looking at some kind of a tabular representation so what are the problems with this tabular representation.

So what are the drawbacks of using a lookup table or that any drawbacks are using a lookup table, memory right. Yeah, good for the time being let us assumed that we will come to the first question is memory, okay next problem is handling continuous states right, then anything else. You are not allowed for the remainder of this class you are not allowed to answer questions, then of people are wondering we just had this discussion yesterday in the lab meeting so in fact he is using a cheat sheet.

Let us suppose that memory is not an issue right, so we have huge RAM machines you know I mean so 1 terabyte RAM is not uncommon now, right let us say that I build a very, very large memory machine right, or you some external memory algorithm for something all you need is some kind of a large table data structure, you need have a data structure that can handle very

large tables and in fact that actually exists external table storage libraries right, that allow you to handle tables as large as your hard drive, right.

So let us say memory is not an issue right, and let us say convenience state is also not an issue and leading up to something else, right so I have discrete states right, and I have a lot of them fine but I also have lot of memory that I can store these things right, so what else could be an issue in having a lookup table representation. What is it, quite a three-time could be high okay, yeah, something non-implementational that maybe I should be say is non implementational but something non system level answer.

In case of no machine learning who have been in general class and things like that should have an answer for this even if not others, now come on okay, so what are we talking about here we are talking about functions right, we are talking about value functions right, we are talking about state value functions and action value functions of v and q. So we have learnt about functions right, in machine learning we have looked at functions right, so have you ever lo used to lookup table for learning a function from data what do we do.

Use some parameters and try to estimate this parameter so why did we do that, so how does learning work right I give you a set of data points right, and from those examples you are supposed to induce forms of functions that allow you to answer questions about points you have not seen already, right. So why does it become important in RL if I have lots and lots of states I have a huge table and if I am using the classical lookup table version of things then unless I have visited that state not.

Once unless I visit that stayed often enough I do not have an estimate for that value function, right that means if I have like a million states or million small right, so in fact there is one problem which I implemented some algorithms on during my PhD., which had 10^{25} states right, and if I have to experience every state in that multiple times I would still not have graduated, right.

So we need something else right, so we need to be able to generalize feel like a dentist extracting answers is like extracting a tooth from you guys I mean anyway fine, let us go back so we want to generalize right, that is very crucial so to achieve all of these what do we do instead of representing the value function as a lookup table we try to represent it as some kind of parameterized function, right. So the most popular choice for the last couple of years has been is going to use a neural network, right use a neural network an artificial network as your parameterized representation.

So what are the parameters here, there will be the weights of the neural network right, so speaking of which so pretty momentous thing happened a few minutes ago so how many of you did you follow the alpha go think it one yeah, so alpha go one against the highest ever rated human master in the game go, so remember I told you about alpha go is RL and deep learning based go playing agent just so the matches are started in soul, right so this guy is a Korean who is the highest ranked ever human in go and the first game I will go for our one.

And people are actually stand even the deep mine guys did not expect to win the company who made this thing they just wanted to see how close they can run the human right and they won maybe he was sick or something I do not know we will see so it is a five match series million dollar prize money they probably win it and then donate it to some charity or something, but anyway so that is amazing right.

So and that uses a neural network for representing the value function okay, so in the parameters we talked about here or your the weights of the neural network right, could be any kind of weights, thresholds right and other parameters slopes of the transfer function whatever right, however complicated you want to make the parameterization right, so that is typically what we do, right. So this kind of parameterize representation obviously, comes with a cost, so what is the cost is it there will be some kind of underlying bias and what does the bias deal to and it leads to some kind of approximation right,

So as soon as we move into this kind of a parameterize representation unless I make the parameters parameterization so powerful that any value function that I want to represent I am

able to put it in here which will be kind of defeating the purpose right, I will be essentially end up with as larger state space as I originally had it as a I mean so if you think about it the look up table itself is a parameterized representation, what are your parameters here each entry a value of each state itself is a parameter, right and you can just think of that as a parameter is representation right.

So if I make my parameterization presentation complex enough that it can represent any value function without any loss right, then the number of parameters I will need will essentially be the same number of entries as the lookup table. So I will not have saved much right, so you essentially you want a simpler representation you do not want something that has that many parameters as the lookup table you want something that has a lot fewer parameters than the number of states and perforce you have to accept an approximation, right.

Because I will not be able to model every value function defined on some state so i will have to essentially accept some kind of an approximation, so this is also known as right, so people call this function approximation RL function approximation and so on so for the chapter itself is called in the new book affected on policy evaluation with function approximation or something like that so that is the title of the chapter right.

And fortunately since this chapter is not edited it also talks about control in the same chapter and the second chapter is called on policy control with function approximation and it is empty the tenth chapter is called on policy control with function approximation is empty and the on policy evaluation with function approximation talks about both right now, okay so it is called function approximation essentially you can use this as synonyms, right.

You can say you I mean a parameter is representation or function approximation typically are used synonymously it great. So what is going to happen so I have my true value function say v^{π} right, so I am going to approximate that by some function $\hat{v}$ which is defined by a set of parameters Θ, so we looked at this notation earlier right, in reinforce when you looked at reinforce I looked at this kind of an notation this when I put a semicolon and write something to

the right of the semicolon it is a set of parameters to the left of the semicolon is the argument to the function okay, right.

So what should be the criteria for deciding on what $\hat{v}$ should be, so there are multiple things I need to do here right, so for the benefit of people who have not done ml right there are at least two stages that I need to proceed here. The first stage is deciding that is the second stage the first stage is deciding how the output is going to depend on Θ, I am saying that some set of parameters, right there must be some functional form associated with the set of parameters and I said neural networks that is one functional form, right.

The parameters are the weights right, I can very simply write $\hat{v}(s:w)$ but how do those w determine what v(s) is right, so that is some kind of a connected neural network I can talk about three layers, four layers or 150 layers I mean if you work for Google or Microsoft you can start talking about another 30 layers of a neural network and things like that right, so how do I this is called deciding the functional class right, or deciding on the class of models that you are going to use for representing this, right is a first thing.

What is it that I am going to do I can use decision trees and I can use regression trees in fact people have used regression trees for representing value functions right, so that is one option and I can use some kind of complex aggregation state aggregation I will talk about that little later because it is a very popularly used the function approximated, right and we can use neural networks another very popular function approximation just to use some kind of a linear parameterization.

So what I mean by that so I will have $\Theta 1$, $\Theta 2$, $\Theta 3$ and then it will just be the weight and they will assume that my state is represented by some features, right if it is a grid well it can be the x coordinate and the y coordinate right, or if it is some kind of robot it can be the x coordinate the y coordinate the angle the velocity which the robot is moving right and the angular velocity and all of those things x velocity y velocity there could be a vector of features right, let us say that my state is represented by some vector right ϕ so $\phi 1(s)$ will be the x coordinate $\phi 2(s)$ will be the

y coordinate and so on so forth, and my value of a state will be $\phi^T \Theta$ right, $\Theta 1\phi 1 + \Theta 2\phi 2 + \Theta 3\phi 3$ and so on so forth that will be my value function this is called a linear parameterization, right.

So what we will do later right is explored linear parameterization more detail but this is a very popular kind of parameterization I can basically the first thing I have to do is decide on what is the class of function approximation I am going to look at, right and what is the second thing I have to do so what is the second thing I have to do right now I need a error measure right, to figure out what is the performance measure I am going to use right and in some sense this can be common across different choices of parameterization right.

It need not depend on the choice of parameterization what error measured I use need not depend on the choice of parameterization but what would potentially depend on the choice of parameterization is how do I find that Θ that minimizes this error measure, okay. So there are three things now right, so the first thing is find out the family of functions that I am considering whether it is a neural network or machine three or whatever.

The second thing is well figure out what is the error measure how I am going to decide that I have a good approximation to the function and how am I going to find that Θ how I am going to find the that Θ gives me this good approximation, okay so that is basically the setup for us these are the questions that will have to ask, right and so what do you think is a good measure Lee squares this square is good enough.

Well it is situation what else you want, right so this squares are pretty good enough right so how LE^2 target that I want right, the performance measure itself would be mean squared error, mean squared error right, so essentially what I will be looking at this continuous I am going to have a integral is this fine, yeah so I left a significant gap here so that should mean something other than 1/n within the $\sum$, right it is 1/n fine.

So what is the problem is 1/n so we said mean square right, so mean so you have to take the mean of the square the square of terrors of obviously right, so if I do it by 1/n what does it mean it is a different mean yeah, so what does it mean I am waving each state uniformly I want to get a

good approximation everywhere that is what I am trying to do right when I do 1/n and trying to get a good approximation everywhere right. If you remember the expected error computation that we did in how many of you where in ml good number, right okay yeah.

If you remember the expected error computation in ml this you have to cast your mind back to like the third class or fourth class in ml right, how did we write down the expected error computation that weighted by the probability of that particular pair occurring right so likewise and then what we what we said we assume that the data is coming to us actually from the probability distribution and therefore we can do the empirical mean you can just divide by 1/n/

Because if our data point is more likely we will assume that it appears more often in the sample that is given to you right that is how we ended up justifying using 1/n right, likewise what would be the probability here. So we are lucky in one way because we have a π here right, and therefore I am going to write some new quantity here any guesses what there is, how many times no, close elect guess that is a technical term for it.

Yeah that is what that is what it is but it is called the let us I have an MDP I fix a policy π right, and then I am looking at the evolution of the system what is going to what is it going to be mathematically it will be a Markova chain okay, it is no longer an MDP right so what do you think deep I of s is the steady state distribution under the policy π right. So it is essentially it becomes a Markov chain if I keep running the Markov chain for a long time right, very, very, very long time it is going to settle into some visitation patterns right.

So the steady state distribution essentially tells me if I said state probabilities essentially tells me if as T tends to infinity I sample a stay I stop the Markov chain running and a sample a state where it stopped at right, the probability that I will find it in that state that is essentially what Dπs is. That I do not have any decisions to make any more right, my π is fixed so when I come to a state I just toss a coin pick the π action according to the π and I go on right, my π is not changing so if you look at the overall transition dynamics right it will just look like this look like a Markov chain, right. In fact we did this when we did that convergence proofs right for like Bellman equation so we defined the transition matrix itself I took the cut action out if you remember I

wrote pπ(j,i) or j,s so when we did that we took out the effect of the policy that we essentially wrote the Markov chain dynamics back there, so essentially that is what we are doing here again so dπ(s) is the steady state probabilities of state s under the policy π.

So this essentially tells me that the states that either I am more likely to visit under this policy if I keep running this policy often those are the states for which I want the approximation to be good, okay. Now let us go back to our sampling case, if I am trying to instead of knowing this dπs if I am trying to solve this using sampling, right so what I am I asking for here what kind of sampling I am asking for here you are allowed to answer this question on policy, right.

Because I expect on policy samples I do not want you to give me arbitrary states it has to be sampled according to the policy π because only then it will approximate dπs okay, so in this building up when we have talked about all of these things individually but you can see that it all hangs together, even when I am doing function approximation right, I am doing this is the thing I want to make optimize so I can do it like this take that as my error measure right provided these are all sample s1 to sn were sampled along a trajectory generated by using π, okay it makes sense.

So now we have an optimization you have a least-squares set up right, when I have the objective function so can I go an optimized set and find Θ I mean let us forget about what is a functional form of Θ now is there are still some questions I can ask in the general settings so can I just take this and go optimize the Θ other way I should write it slightly differently so that it helps me later, only and I go, so this makes it easier right, so this target right, minus my current estimate so this error, so target minus current estimate is error so i take the square and so instead I mean writing it like this does not fit with our target minus estimate formulation that we had earlier so I just looped it around.

Since the squared error is okay, right so is there any problem in trying to optimize this now there is a problem that is very, very serious problem what is the problem. Even before that even before we talk about solution techniques exactly, I do not know what we vπ(s) if I know vπ(s) I have no

problem I do not even have to learn anything right, I put $v\pi$ as my target and I do not have the target what will they learn about, correct okay.

So we have to find out way of specifying the target all right, so what should be out to be a good target we will be spoke about this the same thing we did in the tabular case so Gt will be, what will make you Monte Carlo right, so what else can you do here you, where, no, we do not know $v\pi$ that is what I told you right we do not know $v\pi$ see classically if for regression problems you would assume that you have data of the following form some x and f(x), x f(x) x f(x) and then you go and learn F attics correct, so that is the normal assumption that you make.

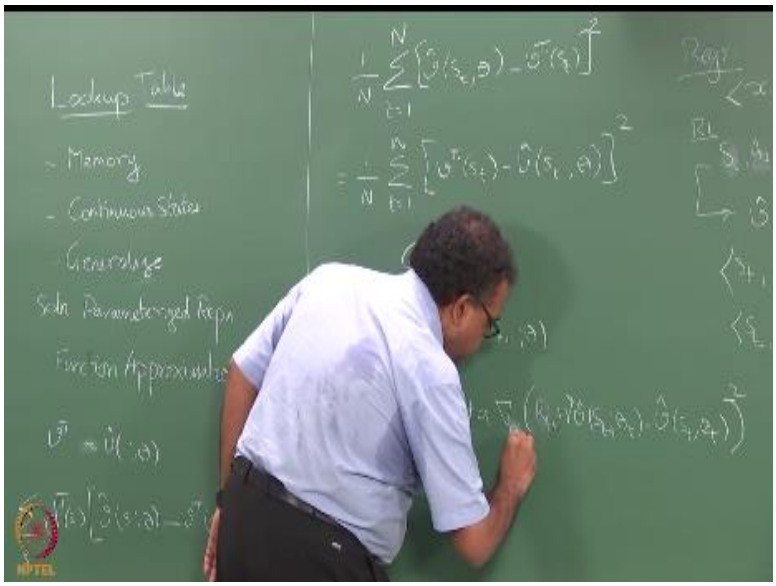
But here the training data will not be of that form the training data is going to be of what form, will be trajectories the training data will be trajectory like St, At, Rt+1, St+1 80+1 Rt+2, St+2 yeah, 80+2 Rt+3 and so on so forth that is how my training data is going to look like nowhere I am going to tell you okay, here is St here is $v\pi$(St) right, so this is the difference so in classically in regression the training data will be x,f(x) pairs and from here you will learn some kind of $\hat{f}(x)$ right.

Here for RL the training data will be right, so somehow from this we need to learn $\hat{v}$ correct, so the training data should start looking something like this right, from here I can create it is one set of training data get I will create right, whatever is the state St right, but it is a first data, second data point will be like that, right so i will create data points like this and I can use it for training right, so alternatively what can you do right, so this is how I will create training data for my if I am going to use my regression as a black box right.

My training data for the regulation look like this right, so this seems to be a little acceptable even though there are problems here but this seems to be a little acceptable than that y what is the problem is that, so what do you do you go back and do it again right, that is where you get into the problem because what you are now setting up is a non-stationary target, so i will make up a set of training data like this right, or let us say I will do one training data like this right and then I go back I will update the Θ right, so i will get a new estimate for my parameters Θ and then I

come back and try to do the same sample again it will be a different value, right because I updated my Θ right, so this will essentially this will be Θ old right.

(Refer Slide Time: 30:59)



So I will be using the old Θ i will make a prediction i will make a target right, for my s1, so vs1 vst should become this where this value was computed using the old Θ now I updated the Θ, right. Once they do this I update the Θ I go back and look at the target for St it will be RT+1+γv̂Θη right, it will be different target, so I am essentially trying to shoot a moving target right, so my targets keep changing after every iteration of updation so it is not like I can actually converge to that right, so lot of people thought this is al lhukum, right I mean how can I use RL to train a neural network because I am shooting at a moving target.

So people say RL would not work blah, blah and things like but then of course we always have this existence proof so luckily Jerry SRO never heard of all of this convergence results and stuff like that and so he fired up a neural network that learned to play back, not a simple single neuron or anything it was a very complex neural network so he trained a neural network you played backgammon so people know that it works, right but seems to be pretty crazy that it actually works and later on the loss of results that show that with the linear parameterization, right.

We can show convergence even with this kind of an update one condition you need is that it should be what is the important condition I said on policy, sampling on trajectory sampling on policy so as long as you are being on policy you can converse that was shown right, so now we even have results for off policy updation right very recent things people are shown some kind of stability for off policy updates as well. So, lot of work is being done in this space okay, because this is such a counterintuitive thing for people doing regression, right.

So every time I look at the data right, as soon as I use the data once the data is going to change right, so I cannot say that I have a stable conversion point but people have shown that this actually happens because of the way the dynamics of the system is set up in fact the way you show that is that with the function approximation also the TD updates give you a contraction, right so we showed contraction long time battery so you have to show that eventually this even with the approximation it is a contraction under certain conditions.

On the approximation so that is essentially how people show this is also works right, so any questions so far on this, okay. So how many of you are familiar with using gradient ascent or gradient descent to learn to optimize functions I expect every hand to go because we talked about in policy gradient yes, can assume that everybody knows this stuff right, otherwise flip back to reinforce so we talked about all of this in the policy gradient thing, right.

So in the policy gradient we did ascent right, why did we do ascent gradient ascent because the function they were dealing with there was the performance right and we want to maximize the performance with respect to the parameters Θ, right. But the function I am dealing with here is the error I want to minimize the error with respect to the parameters Θ so i will go in the opposite direction of the gradient right.

In policy gradient we went in the direction of the gradient here I will go in the opposite direction of the gradient till I try to get to the lowest point one of the popular ways of trying to minimize mean square error right, and very popular variant of gradient descent is stochastic gradient

descent so that I do this on a point wise basis so as soon as I get a single point estimate I go in the opposite direction of the gradient estimate that, right.

So essentially what I can do now is have a truly online TD with function approximation implementation so every time I go to a state right every time I do not do the summation business here so every time you go to a state I form this estimate right give it to my STD optimizer it takes a step that now I have a new function so now if I am going to do that at every time step I can start calling this as ϴT, right.

So essentially what I am going to start doing now is I am going to say my ϴT+1 will be ϴT plus some alpha times gradient of the error right, where the error is computed as whatever so what you want to use, right so this is gradient of the error function evaluated at ϴT exit this will be my update rule, so one thing to notice there is that all the computation there is not $\hat{v}$(St+1ϴT+1) is that St+1ϴT, so depending on the actual form we choose for $\hat{v}$ right, so you will have different ways of computing this gradient and you can compute this once you compute the gradient you can actually make the update, so any questions on this so far.

How many of you are following it how many of our completely lost anyone completely lost is it any questions, good. So let us move on so there are many ways in which you can choose this parameterization like so the ϴ.

IIT Madras Production

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

www.nptel.ac.in