NPTEL

NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

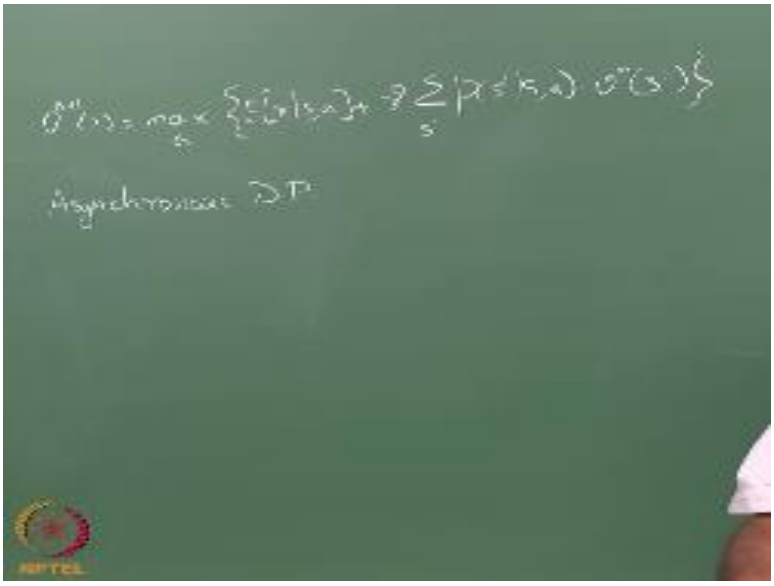
Dynamic Programming

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

We have been looking at the dynamic programming based methods for solving problems so far and solving empty piece right so why do we call them dynamic programming method so what is the problem are we solving here what is this a problem I actually give you the intuition in assuming that you have a like remember I talked about one step problems right as I said that they can assume that the value is one step problem.

Where for the first step you get the reward and then you have a cost for termination or a value for termination and I said you can think of it as R plus γ times V so the γ V is the value for termination right so if you think about what we did for value iteration.

(Refer Slide Time: 01:09)



In order to use it so was it find it I introduced the DARSI notation or there any resist explicitly right so we did this similar mother so suppose I am in a specific state right in which as soon as I take a step the problem is right the episode ends as soon as I take a step the episode ends really it means I am not asking you to hypothesis really the episode names okay if I do this once on that if I say let it start I start off with a zero value function right so this part will be 0 in fact this really does not play a role but anyway this part will be 0 and then I will just get the expected reward right so I take as I start from the state okay.

Let us call with some S1 I take one step and I end okay, some time will say I reach and then you get a reward will say plus 1 or something so the VN plus 1 I compute will be the value function for s1 right just one time I do this I will get the right value do you see what I am saying so there is an end point right if I look at the state just before the end point that is the very first iteration I will compute the right value for the state correct.

Let us say I have another state which is one step before the s-1 right that is all that is only transition available from S2 okay so I take there is an action I take as to and from which to I take the rational go test one okay so how many iterations should I run before I get the value for is 2 so one way of thinking about what this is doing is to say that this computes a solution for n step problem it computes the solution for an N step problem.

And assuming that I start from s prime I will end in n steps this gives me the value right and this concludes the value for n + 1 right so that is essentially where thee the dynamic programming aspect comes in so the smaller problem the sub problem is solved is the one for the end step prediction right and then the larger problem I solve is using this repeat the substructure so the value of n + one prediction will be R plus the value of n prediction so this using this sub problems and making it I am trying to solve the larger problem makes it a dynamic programming.

I is stochastic dynamic programming is typically called stochastic dynamic programming because you mean all this to cast city is there right and it is really this works are in respective of whether it is a finite horizon problem or not right so assuming you stop in one step what is the solution right here I am saying you actually get to the terminal state but you can just think about forget about terminal state any state can be terminal straight but.

I allow you to act for only one time step I will elevate act for two time steps three times for time steps and I keep going to allow you to act for an infinite number of time steps right so that is essentially what value iteration this so it is like a dynamic programming approach right that is why we see you solve for n and then you use that sub problem solution to solve for n plus 1 okay, so that is why it is called dynamic program.

I said clear right so these classes of methods both value iteration and policy iteration So policy situation also has dynamic programming method which is a little tricky to see what the dynamic programming dynamic program there is but still it is also a dynamic programming approach and so I am going to talk about a general class of dynamic programming solution here but before that I want to introduce one other distinction actually a couple more distinction.

So there is something called a synchronous dynamic programming so if you think about this is one step in the value iteration that means I am assuming most people who in here have that in your notebook so I am not going to write out the entire value iteration procedure right so that is

one step right that is one iteration in think that is the main iteration in the value iteration algorithm right.

So what is the crucial thing to note here is that for all the VN + 1 computation right I use VN on the right hand side right so one way of thinking about it is okay I have this array right now I have this data structure some array that represents the value function at time n so I use the array and compute the value function at time n plus 1 so it is a different array right so I make it says I have n and then I make another array which is n + 1 and they compute it and then I copy that n plus 1 over 2n and then keep doing this right.

Instead of that you could think of a way of doing it where I do this in place I do not use to erase I do not use one array for n1 a referendum I can do it with a single array so what that would mean is when I suppose I have some ordering on the state still have to go through some loop on the state's say for s equal one to n or something like that say so when I do a s = 1 right I will be using the old values of V n so I will be using VN right.

So when I use s =2 here I will be using VN for 2 3 4 5 2 n but I will use VN plus 1 for 1 likewise when I use VN + 1 of 3 when I am computing it I will be computing VN + 1 of 2 and 1 and VN of all the other states okay. Now this is a little tricky if you think about it because it is not clear what I am doing right there it is VN + 1 or VN + 2 suppose I come to state 5 okay and all the transition from five is going to state 1 & 2 so when I am actually computing me n plus 1 of 5 I will be using VN + 1 of 1 and V n + 1 of 2 so is it I am a competing GN +1of five.

Or I am computing VN + 2 of everything the second iterate off I am using one further literate of one and two right. So in fact it is not even clear whether if you do it this way it will converge or not right turns out it will converge right and it turns out that I do not even have to do it in this kind of a you know sequential order and I can do it I can pick any state right and then update its value and then pick some other state right now then update its value I can keep randomly picking states and keep updating their value using whatever is the current stored value of its neighbors right.

So instead of using that right so I can do this I mean from an array point of view right so whenever I update some value this is all the Eigen side also will change I guess keep picking states and I can do this right so this is called as synchronous mode for running dynamic programming so the only condition I need now to make sure this dynamic programming converges is I have to say I love our from the other things in it an additional condition I have to assume I have to ensure is that I sample all the states at least infinite number of times so that I can ensure convergence okay. I am just waiting I give you a rock Q as I think it is equally guys will do, but I have to apply the update in every state infinite number of times for it to converge right.

So that is the convergence criterion we saw earlier and only as this the iterates tend to infinity we have looked at all the the fixed point theorems only as a traits tend to infinity do we converge right. So in that case so we have to make sure that all the states are sampled infinitely often right so then it will converge. So it turns out it does not matter in what order example may say actually very powerful reason.

And it allows us to do a whole bunch of interesting stuff right in fact it allows us to do something called real-time dynamic programming which is a very interesting algorithm which is almost like reinforcement learning but not quite there okay. So one of the main problems in doing value iteration and policy creation and things like that is when you are working with really large state spaces right when you are working with really large state spaces you will be every time you do this iteration right.

Whether do it in the synchronous on a thickness fashion so every time you do this iteration right you basically run through all the states. So in fact if you think about the amount of effort you spend in computing the solution you spend the same effort in computing the value function for every state right, but that is not often needed if you think about any practical problem there are many, many states which are very seldom visited right.

So you want to spend more effort in trying to discover the true value function for states that you visit often right, and states that you very rarely visit you do not mind right. So if people

remember the the expected error formulation that we used in machine learning right so we always try to vary the error by the probability of actually seeing the data point. So like that here I would like to if I want to try to do an optimization I would like to weigh the error with the probability of me actually visiting that state.

So now I have some estimate some BBB hat of S right so when I am evaluating my performance I look at the true V*-V hat right times the probability of going to yes right, this is a, so why do I want to do that why do not you, common sense things yeah, you do not have infinite computational power I mean, we just say nicely run none thing still infinity you really do not have infinite computational power.

So that is why we gave you some epsilon criterion so you can sadly satisfy the epsilon optimal policies and so on so forth it so you do not have to run till infinity but if I give you a mdp which has say 100 million states so I do not even want to run into finite number of times on all the states right that is a still a significant amount of computation usually a limited amount of computation.

So now we have to figure out how am I going to assign this limited amount of computation and how am I going to spend this limited amount of computation. So what you would really like to do is spend a lot of time on those states which you are likely to visit under an optimal policy. So you do not want to miss out on states which are good with which you need to execute an optimal policy right.

So one way that people do it is the following right, they start from some state, they execute some trajectory right then they look at states that occur along those trajectory right and do the dynamic programming updates only on those states. Then what you do is now you have a value function right you have made some changes to the value function and this implicitly gives you a policy, so we all know that right.

So you know how to recover the policy given a value function, so I looked at it we looked at it from V look at it from Q given a V function you know how to recover their policy, given a Q

function you know how to recover a policy. So now recover the policy from this value function okay. Now run another trajectory just by running this policy so it is like a sample you are taking from the world.

So it starts from the starting state execute this policy see which are the states that you visit right and then run your updates on those states, but the crucial thing to remember here is that every state you are running your dynamic programming update like this we are assuming that you know their expected value of the reward you are assuming you know your transition probabilities and then with these you are changing the value right.

So I am not worried about what was the sample I am getting here if you think of the Bandit case whenever I draw a sample I get just one reward right. So in this case what we are using the sampling for just to determine which are the states that which we have to do the update right. So not in the reward and all that we assume that the reward distribution everything is given to us we are only doing the sampling to figure out which are the states we have to do the updates.

So it is called real-time dynamic programming I mean the real time is within coats here that is because you are generating these trajectories using the current policy. So the idea is that as your policy as your value function gets closer and closer to the optimal value function the policies become closer and closer to the optimal policies and therefore the set of states that you sample will be those that are highly likely to appear under those trajectories right.

For example, if I have to start here and go here right states at all to the corners might not get some sample that often you know I will probably start trying to go in a straight line from here to here so we will start going like this after sometime. So you can see that there will be some kind of like an envelope of states that will get sampled a lot right and I will be applying the the the DP updates on those states more than DP updates on the side other things.

Yeah sure, so I did not talk about exploitation here so, you never behave getting you with respect to the value function we always have some kind of exploration you could ∑ greedy. So you know about ∑ greedy right. So at every point so I look at the values of the actions or values of

the next state and be greedy only with the probability 1 - ∑ and then the probability of ∑ I do something right yeah you should do something ∑ right yeah.

So this is called real-time dynamic program it is actually a pretty powerful method provided you have a model available to you this allows you to focus your computation only on a small number of states okay. But if you want to ensure convergence in real time dynamic programming right what is the condition you need, you have to visit every state infinitely often I mean, so does not matter that you are focusing your computation.

So what essentially dynamic programming allows you to do is that given a finite number of updates that you can do okay it allows you to focus those finite number of updates in things are in states which are more likely to be useful to you right. But given an infinite amount of computation it will eventually converge assuming that you have an ∑ said never going to zero okay.

So crucial thing that you should note here is we will have the DP we use the sampling only to figure out which states to update okay nothing more than that. If you move on when we actually look at true reinforcement learning algorithms you will be using the sampling to determine not only the states to update right but also what is the reward that you have to update it with because I do not have this reward right.

And also what is the the terminal cost you will use here what is the VS' you will use here because this also I do not have this is why I said, this takes you some ways towards a full reinforcement learning algorithm but it is still a dynamic programming solution because it uses all the expectations okay good any questions on this.

So how many of you have been reading ahead on this is what chapter is am I on in the book okay good how many of you have read chapter four you are my only hope man come on disappointed only gate has been reading so far yeah anyway yeah so I do not think our TDP is that in Chapter four so whatever I told you is the thing if you want to read more and probably have to pick up

the rtd be paper it actually is very well written the version lottery if you captain of course I am a big fan of Andy's writing.

So I always say that but so the next thing I want to talk about this very very powerful technique that we use called the generalized policy iteration people remember policy iteration right so they say essentially has two steps right so you have policy evaluation and then you have policy improvement so what is the synchronous just checking if people are listening you can look we are so you do not have to do it in last step right.

All the states do not have to get updated in one iteration right so there is no actual definition iteration itself is gone here as synchronous state, but anyway so people know me that is policy iteration right. So I have a policy evaluation step right and then have a policy improvement step the PI stands for policy improvement not policy iteration right.

Policy evaluation policy improvement I keep doing this until my policy does not change anymore right and so we did do the convergence of policy iteration then I can kind of convince you that policy iteration would converge at least on finite in degrees. So we have not done the convergence on its infinite state spaces I am not going to do that because most of the results I will be showing you will be only on finite state spaces.

Mainly because I need to develop a lot more mechanism for showing you this on infinite state paces and it really belongs to a course on one mdp not on RL right. So it really belongs to a course on mdp so if interest rate I do not know Sir Rachal Murata used to teach a course in dorms on MVPs I do not think he does that anymore he actually used to teach out of cysts.

So whatever book I have been using to get all this material for you it's called Markov decision processes by put in on so what do we have a copies of the chapters from put minus on this earth I do not tell me if there is an online version of put man but maybe you should tell them not me we do not have a version of Phutra Marissa did they give one last year I think you can pick it up from the last years material or something whatever yeah do not try to read it right.

So what is essentially did was took three different chapters from the book and cherry picked results that I wanted to present here and then kind of you know simplified the theorems and other proofs and so on so forth and presented it to you so if you actually read put Herman directly it in fact some of you might being confused as to where the heck of those theorems he did in class in the book right but I assure you that the place to go if you want to understand this better.

And that is where I took this material from anyway so going back to policy iteration right so there is this technique called generalized policy iteration somewhat like a synchronous DP right in the sense that it does the following so I have a policy I have evaluated the policy right I do not try to be greedy on all the states in the value function I just pick some states right.

And then I say okay I am going to change my policy by being greedy only here right then I go back and do the evaluation of this new policy but then we have to evaluate this new policy in every state only which are updated not really I have to I learned it a little bit more because states that are close by which moving here their values will also change because this values change.

So I have to do some amount of evaluation around the states where I change it right in fact it turns out that this works both ways so I start off with a policy I do not evaluate it on every state okay evaluated only on some states just like we did in our theory p it actually DPI did value iteration right guidance of doing value iteration right I would have taken off this max here right I could have put a summation over π SI, so I start off with some arbitrary π it run this update only on trajectories right.

Now I will sample some states I will update the value function therefore the current π that I am following right and then what I do now I go try to be greedy with respect to this value function now here it makes sense only to be greedy with respect to those states in which no need to be greedy in those states in which you change the value function because other states it would not change okay.

I am only looking at the value function and I try to be greedy right so it will not change but in those states where I change the value function I have gone do the clarification I'll change the

policy now now what I do have a new policy I can go back and sample some more states according to this new policy and into the policy evaluation only on those states at a sample so I am doing both a partial policy evaluation and a partial policy improvement this is called generalized policy improvement right.

So one way of thinking about it is that instead of that's a picture in the book if you do not like so instead of doing a complete evaluation or a complete improvement you are doing this partially bit bits and pieces right so why do you want to do this again you can save computation yeah you have to remember thee in which you did the policy evaluation in policy or even you do not really have to the most effect you will get is if you do the policy improvement only on the states where we evaluated the policy right.

But you don't have to so I can always say that I will pick some random number of states and then I will go I will pick some random set of states I will do my policy evaluation there I will go pick another random set of states and term a policy improvement that if the intersection is empty then you are probably not change my policy at all but that should not be your shopping criterion so you should have a different stopping criterion we are doing generalized Paula situation right.

But the feed intersection is empty you wouldn't stop and then when you would not change the policy go back and sample some more states so that way if you do you will be doing a lot of wasteful computation when the ball the policy improvement it it'd be wasted but you do not need to remember it is more useful if you do it on trajectories so why is that I mean technically I mean the rtdp style of convergence rate I mean if I do not do it on trajectories it becomes more or less like a synchronous DP right but I am claiming that are TDP will have a greater effect than a synchronous tpy I am sorry.

Not just happen more often there is a slightly subtle point here but I am looking along the trajectory so I am actually not only sampling a state I am also sampling a successor I am sampling a successor and sampling a successor so that will be changes you know when I am going to update VN + 1 off remember I put in some space like so other day of sample guess three

right I will re compute the value of S3 λλ that also affect the value of s 2 right so now because I'm sampling along the trajectory I will actually get this sequence here.

So when it when it update is three my stool also get updated my phone will also get updated so because of this connectedness right so I actually have a greater effect for that sample I drew that if had sample some S1 somewhere and this to somewhere else and they are not connected there is no way they are going to win appear in each other's update equations right then I will be using the same world values that I had and I will be trying to update this so that is why so sampling along a trajectory can be more effective.

So you want to sample opposite race again you do not have to right so what we are trying to do is so even how to in the in the sense of time to show convergence right all you need to do to show convergence is you sample it infinitely enough right so you don't have to show that the samples are long trajectories right it is possible that the few wits a little know with a lot of work you can show that by your rate of convergence improves if you are sampling along trajectories.

But I am not aware of any search results that people have actually worked out there maybe just because it is too hard to show and it's not that much of an interesting result anyway but the people get the idea began generalized policy iteration right. In fact you can take generalized policy iteration to an extreme how, I can sample a state okay, taken one action according to the policy we have right and then be greedy with respect to the updated value function.

So sample state take one action right, update the value of the state now be greedy disrespect to that and produce a new policy only for that state I will change the action right, sounds too extreme, right so this is policy evaluation policy improvement we talked about it I just take some trajectories I do the policy evaluation on that and then I do policy improvement and I keep iterating this and I am claiming it will converge, okay so I am not going to show you proofs of generalized policy iteration converging.

But I am claiming that will converge and trust me right, now the question I am asking you is I am doing this one day on one state at a time pick a state, I pick an action according to the policy

right and then I update the value function and then I change the policy for the state to be greedy with respect to that value function. I can do this by randomly sampling states over the thing or I can do this way sweeping through states I start with this s1 I will do this and change the policy they I will start with S2 is change the policy there and keep doing this, as it sound like a reasonable thing to do.

Why I just look at the value function and then pick that one action it gives me the maximum value right, so that is what I mean by greedy is that I am not saying that I have only one choice to make choose from I am saying I have to take only one action I have multiple choices among those i picked the one that is greedy with respect to the very function. No we have already looked at an algorithm that does that that is exactly what value iteration does if you think about it, right.

So I pick the best action right so being greedy I pick one step and then I find out what the next thing is and use that I update the value function, so the next time I come to the state I will pick the best action according to the new iterate of the value function, right so this is essentially what value iteration does is one way of thinking about value iteration is an extreme form of generalized policy iteration and you pick a state right, pick a greedy action, right evaluate the policy then change the policy right.

So in fact we are learning the policy evaluation in the policy improvement step has been rolled into one equation there can a crazy right, so that is what value iteration is in think of it as generalized extreme power of 10 based policy inflation so that does not appeal to you do not worry about it right but what we will do later is that many of the reinforcement learning algorithms that we will look at or some form or the other of generalized policy iteration.

So many, many, many cases that we look at is some form or the other of generalized policy iteration and so one of the powers of power of generalized policy iteration is the following so as soon as I come up with a mechanism by which I can evaluate a policy, if I give you some policy I can evaluate the policy hey, that is all you know right I can use that and to enter it into mechanism for learning to solve the MDP right.

If I give a mechanism for evaluating a policy I can turn that into mechanism for solving the MDP just saying that I am going to repeatedly evaluate policies and then improve between the evaluations, let us as soon as I define that policy evaluation scheme I can convert that into a solution for the MDP itself, so this is essentially the framework that we will see in many of the reinforcement learning algorithms we look at, so we will first look at the evaluation set up and then we look at the optimal solution setup, okay.

IIT Madras Production

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

www.nptel.ac.in