NPTEL

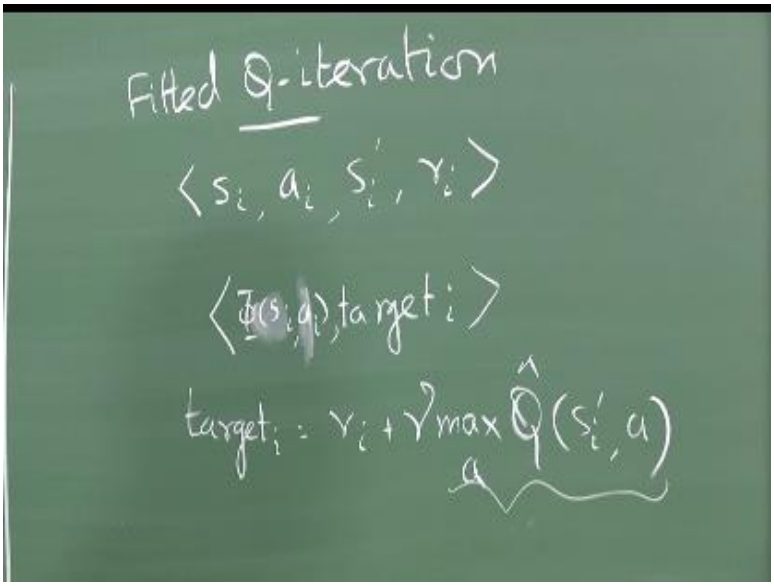
NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

DQN and Fitted Q-Iteration

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

(Refer Slide Time: 00:15)



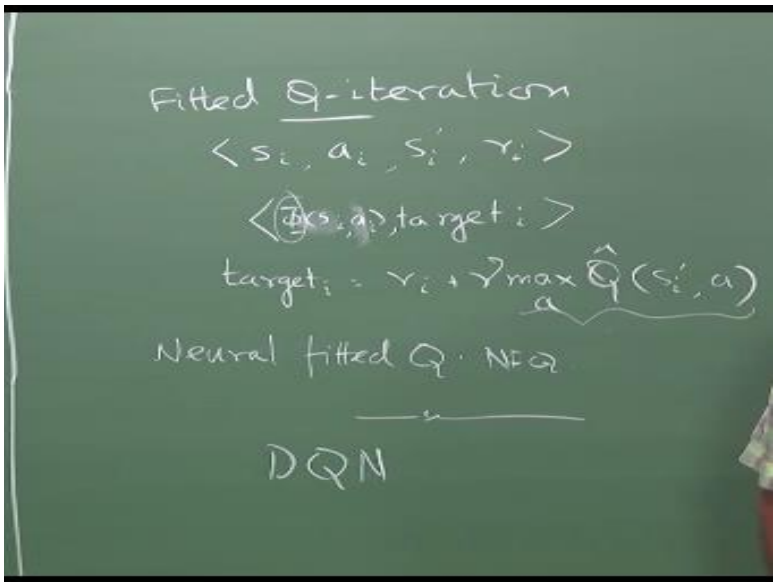
So the next thing I started talking little bit about fitted Q iteration so the idea behind fitted Q iteration is very simple right so essentially what you do is you gather data of the form right, so this is actually what you do in LA study right and what do you do in fitted q iteration, what yeah so essentially you learn the thing so I have my Si, target I this is not really this is not really BSA it will be right.

Is it correct there is nothing wrong right and then what is target I, right so this is the target so I form a set of topples like this feed it to the function approximate right and then it will train something it will spit out a new Q^ right and then I go back I do the same thing again right so what is the requirement that I should have that I should be able to run this max right so that essentially means that I should have taken that all state action pairs right.

I mean otherwise I really have no meaningful data on which I am estimating that max right or I have taken at least similar looking state action pair so that I can use my function approximator in order to generate the values for those right so the static training data that I work with let the static training data I work with should be broad enough that should be representative enough so that I will sample all kinds of state action combinations at least modulo my Ø .

So what do I mean model of a Ø, when I look at the Ø(S, a) I should have sampled enough of those representations that did not necessarily sample every state action pair right but I should have sampled it enough so that across the Ø I have sampled all possible state action combinations I will see some amount so that the generalization is useful okay so this is essentially fitted q iteration right.

(Refer Slide Time: 03:30)



One very popular version of fitted q iteration is essentially Neural fitted q or sometimes called NFQ so what is what do you think neural fitted q as, exactly right you keep it you make this make these data sets right you make this data set affinity a neural network right and then it in your network checks away and then it gives you back a new Q hat represented by the neural network and then you create another data set and you keep giving this to this until it converges okay.

So what are the problems that you might face when you are doing this one I already told you, what I mean I discuss one problem already with you what is a problem yes exactly so you should make sure that you have in our samples in the data right so that is one that is one problem right another problem is in general for any function approximation what is the other problem let us assume that I have a way of doing max on this yeah maths is a problem.

But suppose you have small number of discrete actions and I can train one network for each and so I can just evaluate each one of those neural networks and then figure out which is the maths rate, so if you have small number of discrete actions you can get away with that remember the very beginning I was telling you for representing q you have multiple options.

So one of them was to have different networks or different function approximate is for each action and in this case you will have different neural networks for each action but what is the other problem that could encounter this is not something peculiar to what I tell told today or yesterday it is a problem with all kinds of function approximation, which is what we have been talking about for the last two days.

Whatever it is your favorite training algorithm you use it right I am give you training data is a regression problem solve it using whatever you say favorite recursion right, why not it takes the state as an input gives QSA as the output you should revise from previous classes since this is one of the options I gave for doing control with function approximation right infact I in decay index the year the Ø with a as a subscript, right.

I said Ø a of s is something that you can use that is one different feature sets you could use for different actions as well and so on so forth and then I also had a θa and surf we looked at all of that for the problem knows the problem what is the problem what come on in yeah but I am not solving anon-stationary equation problem here right there it is a nice thing about all of this fitted cube business.

So every time I solve the regression problem solving a stationary problem LSU whether it be L STD or fitted Q so every time I am solving a regression problem I am solving a stationary regression problems it is not like the online updates that we were doing things when non-stationary, so it becomes a little Airy but now I am every time I do it is actually a fixed data set on which I am fitting.

Here we go is assuming that the cost is cheap, no the experience we are not generating again but this Ø target pair we are generating every time we go around right that is what he meant oh! My god exactly my god okay so long to get there now I told you what target is but who gives you the Ø right so we spoke about some very simple ways of generating free assuming things are going to be linear function approximated as and things like the right.

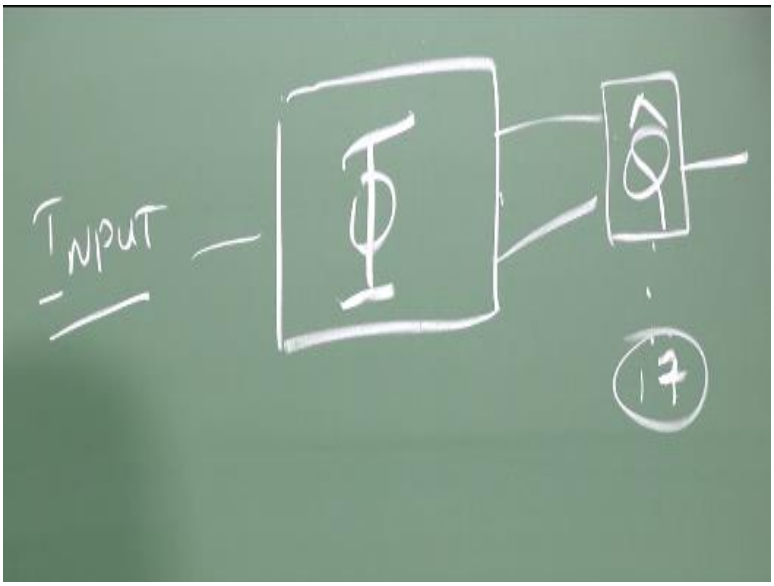
But then how do you come up with Ø, whatever nowadays people do not care about that right now a days you know what they do they throw a deep neural network at it they do not throw a normal neural network at it they throw a deep neural network at it right and then it learns learn stuff if you do not as you wish linearly independent have you seen any convertible for DQ and I do not ask you know people do not show convergence.

They just dazzle you by organizing a public match with the worlds champion go champion and fix like this you do not ask them oh by the way can you show me that alpha go converges right it just beats the world champion you got do not ask them for convergence rate so that is basically it so the proof of the pudding is in the eating in this cases right essentially it just works, so I did not mean to imitate him or anything just a subconscious right.

So DQN is a deep q network their address is a few of the problems that we spoke about the first one is it does not use a offline set of samples it actually samples online right so then that is in some sense allows you to not have to worry about the restrictivity of using off site offline set of samples right the second thing it helps you solve this the Ø equation, so I am not going to tell you how it solves the Ø Equations.

Because it requires me to tell you about deep networks requires me to tell you about convoluational networks and so on so forth right so let us ignore that for the time being let us say that I give you an input right.

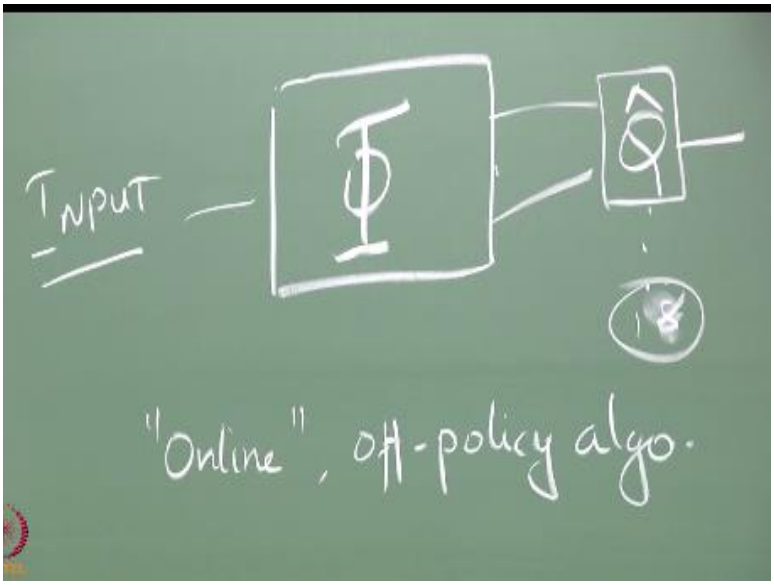
(Refer Slide Time: 11:00)



I give an input it goes through black box okay that computes a Ø and I get the Ø at the other side and then it goes through a Q^ network and I get this right, so in this case they basically have so the DQN is the is the network that was trained to play the Atari games wait remember I showed you Atari games in the very beginning so D cares of the network that strain to play Atari games.

So in Atari games you have 17 actions so have eight directions in which you can move your joystick and then there is a fire button right and you can move the joystick in 8 directions with the fire button pressed or without the fire button right so that gives you16 and the 17th action is so on right so do not do anything right so that is the sum detection so there are 17 actions so you basically you have something says 17 such networks here right.

(Refer Slide Time: 12:28)



18 is over the 18 section this fire and do not do anything I thought was seventy okay they say it is a team is 80 yeah I am quite sure then the eighteenth action must be do not move the joystick but just press the fire button okay that that has to be an action for that right so yeah so 18 actions so there are 18 6 networks and so we get 18 outputs at the end of it rain this of this give you all the Q values right.

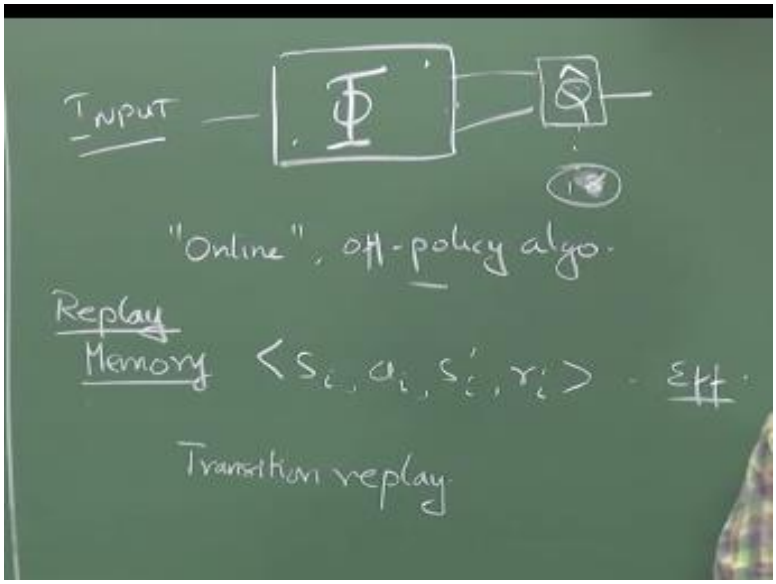
Now you can just go pick your best Q right the best action according to the Q function perform that right and then see what happens to the state it will go to the next state automatically and then what you do you feed that next state into the Ø box right and then you find the corresponding action right and then you set for your cue update I just do it in one shot right so you do not actually do this kind of target generation.

And then have a whole set of things and then do this you are actually doing online updates as and when you see a transition you do this updates right but there are problems in doing this right so you could potentially run this as a right potentially then this is online but of policy algorithm because q-learning is off policy right you are learning about optimal policy while you are doing some behavioral policy you could also run it as an online algorithm.

Because we are going to be updating along the trajectory that you are see right I am NOT writing down n equations all of you know what the Q learning equation is right so essentially you do the Q learning that is possible there are some problems in doing this online updates for this so what do you think of the problems, problems in doing an online q learning variants yes can you elaborate on that bunch of things that can go wrong so the first thing is well if you go by if you go by what the what the paper the DQ and paper tells you the first disadvantage they point out is that you are not using the sample sufficiently right.

I am just taking and drawing one sample right and then I am just using it for one update and then I throw it away right great is there some way of fixing that much more essentially what you do is.

(Refer Slide Time: 15:13)



You basically have a memory right and you add your to the memory it is as and when I get an interaction here in the world I keep adding it to the memory right and then I keep pulling these things out of memory once in a while and then I replay them rights I do not necessarily need to have actually seen the state again but then I will just pull it out of my memory and I will replay them.

So therefore this memory is sometimes called replay memory okay so replay memories are not something which the DQ n guys introduced it actually goes all the way back to 93 this guy called Longi Len right who did bunch of it actually interesting stuff with RL so one of the things that Lynn did was this experience replay, so what Lynn's experience replay was is that you remembered entire trajectories like this right.

And you just pull these trajectories out of your memory periodically and then reapply those trajectories right so from the start to the beginning right it is just like you are running over the same thing again and again in your head you know it' is like reliving your life's experience so you just did something yesterday and then you just think over it in your head and then you say okay that is what I did.

So let me let me memorize it or something like that you kind of replay your experience so I am not being facetious here and it is actually seriously as one of the things you think about it humans actually do a lot of replay in your head when you think about what you did right so this is why people play a short again after they have played it and think like the kind of replay what you are thinking right so this is the kind of took inspiration from that and then came up with this kind of replay memory right so but in the lean version of replay memory you replay and I trajectories the difference.

So the way at least so which certain talks about planning is that you use these experiences that you have right and convert so go from SA, SA' R topples like this you estimate your probability of s' given S, A estimate your expected reward given SSA' right given SA S' exactly so this you are not this is like this is like exactly like having the same experience all over again right if you run a simulation that is different.

If you run a simulation that means you are sampling you are building the model and then generating a sample from the model right so in the lint case you take an existing sample already which is sample from the real world right you do not rebuild the model right you take the sampled world and then you just rerun the same sample again and again it is like over sampling the same thing again and again right.

So instead of trying to do a Cauchy conscious estimation of the model parameters it could very well be the system is very complex right and you do not really know what are the possible outcomes right so you just in some way if you think about it if you if you go through the samples often enough right you can argue that what you are converging to will be something similar to the certainty equivalence estimate for this data that you have gathered right.

But then you need to have some little bit more constraints on to ensure them right so what replay memory is a lot lighter than actually trying to estimate the full model you do not try to do a model for example in the Atari case right your states or continuous right so you are really not fitting a model right so you are really not trying to fit a model you are just replaying the episodes at your place just like.

You play one game and then you just play it again and again and again hundred times yeah, Lynn's word low does not sample uniformly he samples according to the trajectories that are the according to the current policy, Oh from the memory he samples trajectories uniformly he does not sample transitions any samples trajectories uniformed yeah and is that is what I remember the 93version of the paper.

That he had a follow-up version in 96 I do not know if he made any changes to that but 93 version of replay memory that is what it did is a sample uniformly from whatever is that you just keep throwing them into memory and then the sample trajectory is uniformly from good point these are all questions to ask right these are all different variants that you can explore so the original DQN thing what they did was they just basically had a memory of the last hundred transitions or so at the original DQ in paper.

But then since then there have been many variants that people are played around with where they throw away some of the trajectories so preferentially right earlier it was just based on history more than 100 time steps whole they throw it away right so now you can start playing around with preferentially retaining more rare transitions in the in memory and things like that yeah cut at some point all kinds of patients on the path yeah.

So the lots of issues that will have tracking it right ok so the replay memory allows us to do efficiently use rain so the efficient reuse of samples is taken care of, so what is the next problem that we need to take care of this comes from the fact that we are doing something that is online right so it gives rise to two artifacts one if you think about it my states are changing slowly typically right when I am looking at these games mistakes are changing slowly right.

And typically my reward also would change slowly right so if you look at successive inputs that I am going to see they will all look very similar because my States would have changed very little you know maybe one enemy moved here and I moved here and the bullet has moved up two pixels right, so the states will look all very similar right there for the changes the updates that I am going to be making will be very strongly correlated right from one update to the next that will all be very strongly correlated so that can lead to all kinds of problems in the convergence right.

So what they wanted to do was a way of breaking this correlation in fact they had problems with convergence when they are actually doing the updates only along the trajectories right they have problems with convergence because every update was very similar teacher so it is like taking the same update and then applying it multiple time's right so it is kind of messes up with E with the gradient following that you need to do right.

So what they did was they said okay we will put everything into replay memory so every time we get this transition we put it into replay memory but I might not necessarily update for that transition right I put it into replay memory and pick one transition at random from the replay memory and I will update it right, since I am picking things at random from replay memory I do not have this correlation.

So now I may be updating for some state here right next time I may not be updating for some other state there because it might be picked from a different trajectory and so on so forthright so I am I do this kind of random sampling from replay memory which is this is what where it is different from Lynn's replay Lynn's replay was on trajectories here the replay is on samples right and why are we able to do this because of Q learning.

And because it is off policies q-learning and I amusing the max over the next state right because of this I am able to do this right because I don't remember I reiterated many times when we did q-learning that you do not have to update along trajectories for Q learning and here is a case where you actually put it into practical use that because you are going to get this from replay memory so you do not have to.

So there is a second artifact that is introduced by doing the online updates so what is that it is kind of a bias sampling of the input space right because it is going to depend on the actions you take suppose I decide to go left at some point right so the kind of features I will see will all have the player to the left of the screen right so that again increases the correlation between the successive inputs I am going to see.

It also skews the kind of samples i am going to get I am not going to look at parts of the state space which are to the where the agent is to the right of the screen right there will only look at parts of the state space where the entrance on left of the screen if you remember what is this input going to be for Atari, the pixels on the screen right this just looking at what I saw whatever is coming on the screen.

So I am going to see only some parts of the state space in a single trajectory right so essentially what will happen is only some portion of the state space gets their value functions updated the other portions of the state space do not so this can lead to some kind of unbalanced updates in the weights face right so too and this kind of a transition replay takes care of that also so these are the three main difficulties which the paper Lists out.

One is inefficient use and the other one is this correlated updates and the skewed sampling and all of this three can be addressed by using a replay memory and picking things from the picking things at the transition level yeah that is what the last problem right, so see suppose I decided to move to the left at some point right so essentially then for the next few stand steps right I will be only seeing the agent to the left of the screen right.

All the states the agent to the right of the screen will all get ignored so the sampling I am going to make is strongly driven by the policy that I have taken we capture the whole screen is not it yeah but the whole screen is there but that is just one state right the screen with the agent here is one state the screen with agent here is a different state but what I will be seeing is only some subset of the states right so that subset is determined by their policies I am taking actions I am taking.

Therefore things can get really skewed up right so only one side of the screen or only when one part of the state space will get updates the other parts of the state space does not get update so this can lead to some kind of a you know because this side is all incorrect values right and this side is all getting correct values but then they might get updated by the incorrect values here so things can get completely messed up.

So you really want the whole state space to get some kind of uniform updates so that you do not mess up the overall backup process, there another online this thing's the other online methods also had this problem that is good point it did not really did not creep up in the empirical setting until we moved on to such a large state spaces, right so only when you have went on to such very large state spaces where you have a significant difference it became a problem.

With your little suppose it is just the problem of doing online updates I did not see it as a problem with digital networks it is a problem with new letters to the extent that the function approximate has now become so complex right so what you do in one part of the state space can very non deterministically affect what is happening in other part of the state space earlier when you are looking at linear function approximated things are a little bit more comprehensible right.

So now people just take stabs in the dark saying oh this might be what is wrong and then they try fixing it and if it works then they conclude there was what is wrong okay so that is essentially how we do work with this really multi-layer complex neural architecture you really do not know what is going wrong so you make hypotheses you try fixing this you really do science know right so you make a hypothesis you try fixing it and if it works great if it does not you go back and change your hypothesis and try something else right.

So that is essentially okay so go to any other questions on this sorry take advantage if your Ø function does then great right you know all about convolutional neural networks do they take advantage of symmetry, depends on the kind of symmetry they do not take advantage of all the symmetries are out there right so depends on what the Ø network thus so if you can actually start injecting ways of taking care of symmetry then great right.

So sorry no that is all that black box there that Ø is deep learning okay, any other is it just as a register normal two or three layer neural network so that you try yeah cube times since I remember Q prime is a one-layer you will occur right I take the D prime and then we have a completely connected layer that is your Q prime right multi-layer is it only one layer is a one-layer neural network.

So this is the original DQ n is 2 layer and the current version of DQ n is 3 layers but it is a convolutional neural network so it is not layers as you normally understand it and this is a one layer network that takes the output of this and chugs off and it is not one networks it is 18 different one layer networks, I mean you are not you are not restricted to use a needle network there right EQ prime could come from anything.

As long as you can compute a gradient through it, well considering that Q prime is actually a neural network so if you think of it is like a four-layer neural network right I split it up as Ø and Q prime but if you think of it, it is actually a single box from the input to Q prime so it is like a four-layer nuclear attack the first three layers or of one type and the fourth layer is of a different type.

So I mean it you are just talking semantics and you can apply that deep to anything your thing so if you want to say oh we are only doing deep representation learning on top of that I am doing a one-layer neural network for regression yes sure you are welcome to say that right, there is no I mean there is no standard nomenclature here good.

IIT Madras Production

Funded by

Department of Higher Education

Ministry of Human Resource Development

Government of India

www.nptel.ac.in