NPTEL

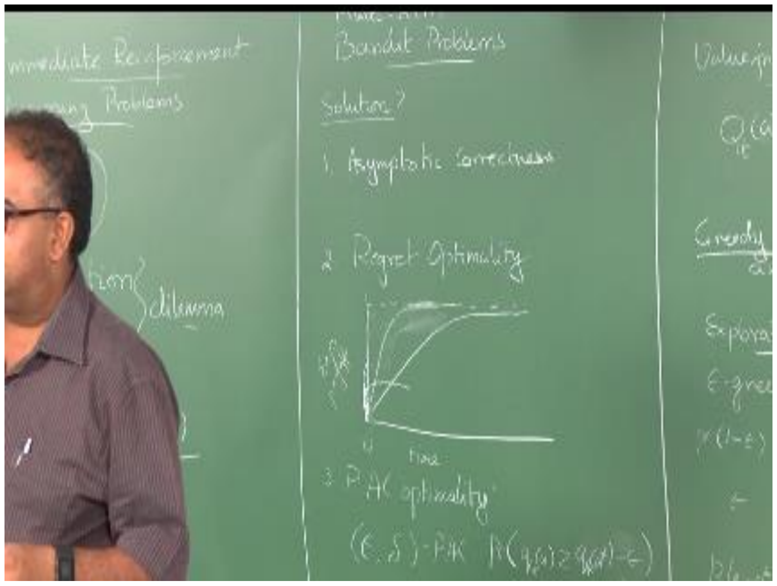
NPTEL ONLINE COURSE

REINFORCEMENT LEARNING

Value Function Based Methods

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

(Refer Slide Time: 00:15)



So the first thing I want to talk about this or a very simple, what the textbook calls value function based methods right, what the textbook calls value function based methods these are approaches where I do not know the Q* right what I am going to try and form a estimate of the Q*****. So how can I do that so what is Q* the expected payoff right.

So I play each action, I choose each action say 10 times I can take the average of that gives me an estimate for Q* right. So we will denote the estimates by QT. So whenever we attach a estimate to an action we will be using the notation Q take this list, so that you know so we use Q here, and whenever we esteem we will attach a estimate of the payoffs to a state we will denote it by V, V stands for value function okay that is easy to remember do not ask me what Q stands for.

So one of the problems with the RL is that because there are so many different influences that came in right. So you have at least three different sets of notations that people use in the literature okay. So you have to swap back and forth if you are reading papers in reinforcement learning right, so somebody will say that the value function is denoted by J right and Q denotes the cost function right.

So if they use disjoint set of symbols at least is easy for us right, but they will use the same symbol that one body of literature will call the value function another body will call that payoff function right which is essentially corresponds to the reward that you are getting at every time right. So that will be denoted by Q, so the all kinds of confusions are there. So this is one set of permutations that I am choosing essentially because it is the one that is adopted in the Satan Berger book.

In fact people who are reading edition one right there has been a overall of notation in addition to right. So to the extent possible I will try to follow the addition to notation, but addition one is ingrained in me right, so I mean, I have been using addition one notation for god knows, 15 years or something right.

So suddenly they decided to change the notation in addition to, so by force of habit I will slip into addition one notation occasionally so it is in your well-being to read ahead so that you can correct me whenever I do that right. So do not wait for me to teach and then go and are well do not wait for the quizzes and then read the book right.

But read ahead and come to class okay. This will certainly help you significantly in following the material better right. So I am deliberately going slow today warming you up okay. But then there is lot lot of material that we cover in fact usually I cover the material in the book in about one and a half months right in the rest of the time I will be doing material outside the book that one and a half months, two months okay.

So the book is big it is at some point I will switch modes and start going fast. So reading ahead helps okay, so we are going to have QT so essentially what you can do is you can just take whenever I took action A right the reward I got at that time and I can just keep adding those up and divide by the number of times I took action A that will give me my QT right.

So how will we write this, let us look at interview some notation so that is a notation that I will use whenever I want an indicator function right. So normally what is the notation that people use for indicator functions I or Δ right, so people typically used Δ or I so I will use one because we use I and Δ for other things later on okay.

So this essentially means that it is one whenever it is A or it is 0 right, so should I divide this way okay. So this gives me the average right great. So once I have this now I have made an estimate for my Q **star**ts right. Once I have an estimate for my Q* so how will I select an action apologies okay. So I use t on the left hand side so I cannot use T as the running variable as well okay. So now I have come to time T, I have this estimates for my Q*.

So how can I use that for selecting actions select the maximum right. So I could be what we call greedy right, so greedy would be I am going to select right. So I will take that action that gives me the max it is R max is the action that gives me the max right, so I will take that action that gives me the max and use that as the action to perform at that time right.

Is there a good choice or alternatively what does this choice correspond to exploitation this choice corresponds to exploitation, but I cannot just exploit right I have to explore as well, if I cannot come up with an algorithm that only exploits right. So what do you think will happen let us say that I pulled an arm right the first time well I select and actually and then you get a reward of 1 let us say right.

So what will happen to my Q it will be one, what about other Qs well depends on what you initialize them with the finish line with 0 they will state zero, but I could have initialized them with the 100 right I could have initialize them with -100 I do not know whatever it is initialization with then stay there, but this one will become one right.

Now if I am doing greedy what will happen I will do this again let us assume that I get a 1 again right, and they keep getting one every time I pull this. So what will happen is I will never ever sample any of the other arms right. It could very well be that arm 3 has a reward of 50 right, just because I happen to pull arm one the first time right, I will never ever go and pull log 3 even though it has a much, much higher reward right.

So we do not want to be exploitative completely so we need to have some kind of a exploration policy okay. So the simple exploration policy okay, so I am going to confuse you a little bit is called ∑ greedy nothing to do with this ∑ okay. So this ∑ is completely different okay, it talks about a solution concept and just because I have something ∑ greedy does not mean I am going to give you some ∑ pack guarantee okay. ∑

This is a completely different ∑ in fact it was named ∑ greedy before people even started thinking about bringing all this pack optimality equations into RL okay. So what is ∑ greedy means any guesses sorry, within ∑ of the function okay good choice. But I said this has nothing to do with this ∑ it is got nothing to this ∑, what I do is with probability 1 minus ∑ I will be greedy okay, with probability ∑ I will just explore okay.

So ∑ greedy means with 1 minus ∑ I will choose probability 1 minus ∑ probability ∑ I will choose which is uniformly from the set of actions A okay. Si it is one thing which you should note here is that if I choose uniformly from a I still have a probability of hitting that R max A right. So in fact the probability of me taking this action is 1 minus ∑ plus ∑ by n right.

The probability of me exploring is actually n minus 1 ∑ by n okay, so this is something which is just a small thing net picking thing but just remember that. So the probability of exploration is not ∑ because I am picking uniformly random you could choose to say that I pick uniformly from a minus this R max action right does not make a difference, but I am just pointing out that there is a small difference just remember that okay.

So ∑ greedy by far is one of the most popular exploration is by far the most popular exploration strategy that people adopt okay, by far the most popular explorations. There is another one which ironically I could not find in addition two of the book which is also rather popular is called soft max or maybe I flip through addition to too quickly to notice it, but so the soft max exploration policy there is something slightly cleverer okay.

So if you think about the ∑ greedy what is it that you would notice is that I have the best action the best according to my current estimates right. So when I say, whenever I say greedy or best it is according to my current estimate it is not Q* okay, it is according to capital Q whenever I say greedy is according to capital Q okay.

So the greedy action gets the bulk of the probability right, and every other action as a equal probability of being picked in exploration correct. Suppose if some actions are clearly bad right if I figure out some actions are clearly bad, so I do not want to be picking them in during exploration, because I know that they are bad I really do not want to be picking them during exploration or at least I want to reduce the probability of picking them during exploration right.

And if two actions look equally likely to be good right maybe 0.0001 difference right some small Δ difference between the two actions in ∑ greedy what will happen the one that is that 0.0001 higher will get the highest probability okay the other one will have the same probability as the all the other arms of being picked right.

But then in reality I just want to be able to compare these two things more closely and figure out which one is better right, because the difference is so small I really want to compare them and figure out which one is better right. So what people did was they came up with this soft max idea right where instead of giving all the probability two one action right I make the probability of picking the actions proportional to the estimate current estimate.

So whatever is the current Q I will make the probability proportional to the queue right. So ideally it should be something like, so this should be the probability with which the TH step I will pick action A right that should be the probability right, what is the problem with the different orders yeah, different signs right.

So I mean the things could be also negative numbers inside you I could even end up with a negative probability right, if my QT(A) was -1 then I will end up with that minus something as the probability of picking the ration does not make sense right. So to encompass all kinds of options what do we do exponential right.

So why is this called soft max because it is not greedy just not written to you the one that has the highest Q value all the time, but if there is a big difference it will return that value to you most often. I suppose there are four actions right and action 1 has a value of say 19, action 2 has a value of 1.2, action 3 has a value of 0.7, action 4 has a value of -0.3 right.

So now this will return to you most often it will return action 1 to you right, so because that says a significantly higher value okay that is why it is called soft max right. Sometimes what people do is they throw in another parameter called the temperature parameter sometimes so β right. So now think about if β is very, very high.

So what will happen to your soft max function everything will get squished down right, so β is very, very high. So essentially you will be behaving at random, uniformly random right because all the regardless of what the Q values are right if the β is very, very let us say β is 1 million right and I say 17 and 1.8 no, no it does not make any difference right, all of them will look close to 0 right.

So essentially I will be picking all actions uniformly randomly right, this way it is called temperature. So if you are very, very hot right so things will move around at random right and then people typically cool it down say they make β smaller and smaller and smaller so that from a very high temperature where you are essentially behaving at random regardless of what the Q values are as you keep cooling it down it becomes more and more β.

So if beta becomes closer and closer to 0 what happens it becomes max right as β goes to zero this function goes to max is it become clean. So it gives an additional parameter for tuning right if you do not trust me you can note it out right way it is smart enough to work it out. So this gives you an additional parameter for tuning how sharp your probability needs to fall right.

So this is called soft max and so if we think about ∑ greedy then writing it in this notation. So this will be right, so probability AT= A* will be 1 minus ∑ plus ∑ by n where A* is the greedy action according to the current estimate right. The probability AT is not equal to A* or probability AD = A which is not equal to A* will be ∑ by n okay, and likewise here you have this probabilities here okay.

So these are the two rather popular ways of selecting actions in fact ∑ greedy and soft max are also used in the full reinforcement learning problem right, they are also used in the full reinforcement learning problem for selecting or for implementing the exploration policy right. Now you can convince yourself that ∑ will be very simply you can convince yourselves at ∑ greedy right. We will give you asymptotic correctness provided something but can you convince yourself that it will give you asymptotic correctness right.

You are always going to be taking all the actions right so eventually you will converge to the true to expectations right you will eventually converge to Q* because you are taking all the actions all the time right. So but what is a problem if I keep my ∑ fixed at some value let us say I keep ∑ fixed at 0.1 at 90% of the times I am behaving optimally 10% of the times I choose at random right.

So what will happen I will know the right Q* after a long time, but I will still be behaving randomly 10% of the times right what asymptotic correctness says is eventually I should be only giving you the behaving according to the right arm right. So I should be only doing the right choice eventually to assume that what should you do, you have to cool ∑ right, so ∑ has to keep decreasing.

But then you should not decrease it too fast if you decrease it too fast then you do not have the guarantee that you will get the correct Q* right. So there are ways in which you can balance this right and so in fact people show that if ∑ falls as 1 by T right that is usually a sufficient rate, but unfortunately that will actually slow down your convergence tremendously because it becomes truly an asymptotic convergence I mean you have one little infinity for you to get convergence.

So in practice what you can do is you can cool it in steps it set the high ∑ right, run it for a while okay, and then drop into a little bit, and then run it for a while, and then drop it a little bit, then run it for a while, and so on so forth right. So the figuring out how long the while should be is a tricky question depends on a variety of factors but you can do some kind of empirical estimates for the time being right.

So do we move on from here, so before I go on to regret optimality one small thing which I want to talk about right. So here I gave you a way of estimating the QTs right. So what do you notice about that I run an experiment still time T right, and then when I want to know what QT is I look back and see how many times I have taken action A and then I take the average right. So that essentially means that at time step T I need to know what are all the actions I picked earlier and what were all the rewards I got for those actions right.

So I need to remember this whole history okay. So is there a more efficient way of doing it I am sorry, yeah so I can essentially do this in a kind of doing the run I can do this running some right. So how do I do that right, so I want QT(A) right, so it is some numbers right. So it is intensely QT-1(A) right. So let me introduce a little bit of a notation make my little easier okay. So the number of times have taken action A I will denote by NA okay, right look at this okay great.

So assuming that I actually took action A at this time right, so what will happen okay, you want me to erase and rewrite both the numerator and the denominator only a denominator, numerator sign okay and assuming this is fine. So now I can do this in a incremental fashion right, so this is NA to NA+1 RT/NA+1 I can just keep adding this right.

So I do not need to remember all the rewards all I need to know is how how many times have taken action A so far and what is the previous estimate of QT right. If AT was not equal to A what happens this will vanish this will vanish the NA will get cancelled just the same thing okay so no change. I decided this is a different way we can write this, QT(A) okay is it fine right, this is N/NA+1.

So I can write it as 1-1/NA+1 right. Now I can rearrange this quantities and get a better looking expression right looks good. So this is the form of a general class of equations known as stochastic averaging equations and this really no stochastic that you need to worry about here but this is called stochastic averaging equations.

So one way to think about it this I have an old estimate right and to the old estimate I add a error term, so what is the error term this is my target right I should have predicted the reward is RT right so this is my target and that is my actual output right, QT - 1 is what I said will be the payoff RT was what the worthy actual payoff that occurred right. So RT- QT – 1A is some kind of a error term right.

Now this is some kind of a step size 1/NA+1 is some kind of a step size. So what I am doing is I am taking my old estimate and adding some α times an error right. So many, many iterative algorithms that we have looked at even if you have taken ml that some of the algorithms we looked at in ml right and many iterative algorithms that we will look at later in the course will all have this form, old estimate times I mean plus some α times an error term right.

So what is the thing that you should note here why is it called stochastic averaging rule, what I am really trying to find out, what my real target is the what is my real target, the expected value of RT the real target is not RT right. The real target is the expected value of RT right, so my actual error should have been expected value of RT - QT - 1 right, that would have been my actual error.

But since I do not know the expected value of RT so what I am I using here is a some kind of a measurement of the expected value right and the unbiased measurement is a sample drawn from the distribution according to which you are taking the expectation, when I say expected value of RT let so RT is a sample right, and this is I think I am going to take the average of all these samples that is going to give me the expected value.

Since they do not know the two expectations I can operate with these samples right. So I am going to form the estimate of the target or estimate of the error right by using a sample as supposed to using the true expectation that is why we call these a stochastic averaging rules. And so this is a very standard form that will keep seeing again and again in all the algorithms that we look at right.

So current estimate plus target- current estimate right, so this is thing. And this particular choice for my α here 1/NA+1 gives me the the exact average I mean this is exactly I did no approximations anywhere this gives me exactly the same estimate as I would have obtained by storing all the values and taking the average at every time okay, great.

So we will stop here today or we can just do one more thing and then we can move on to the other things in the next class. So far we have assumed that your coins are not changed right, so we are assuming that the distributions from which the rewards are sampled or the payoffs or sample is stationary okay constant has a different meaning stationary, stationery means it does not change with time okay.

So if it is, so when they say something does not change with time what does it mean, it means the coin does not change with time, it does not mean it always comes up heads okay, when you say something is stationary okay it does not mean that its outcome is always the same okay. It means that the distribution from which I am sampling the outcomes is always the same okay, but sometimes you might have non stationary problems right.

So it is basically keep tossing the coin the coin once I starts wearing off a little bit right, so after a while it might start falling tails more often than heads or what will happen if the head wears off, will start falling heads more often or tails more often. Start falling heads more often right, because there will be more metal on the head side, so it will be heavier so it will tend to come at the bottom more often.

So if head **star**ts wearing off okay it will tend to come relatively more frequently on the top and it used to earlier anyway fine. So things will start changing our time right and so what do you do in such cases. So you need a way and there are technical ways of handling it I am just giving you a practical way of handling, you need away to be able to track the changes overtime right.

So if I am going to use this kind of an update so what will be the problem after I have pulled arm say 1000 times write this number will become very small it will be 1/1000+1 right it will be 1/1000+1 right. So the impact of the new samples I am drawing will become smaller and smaller and smaller right. Suppose I pulled an arm 1000 times okay 1000 and first time I pulled on right the impact will be very very small.

And if you think about it, I am averaging like a 1000 numbers any individual number is going to have a very small impact on the average right, but this is all fine if I am sampling from a stationary distribution, because I already done 1000 samples what is the 1000 and first sample going to tell me right.

But if I am drawing from its changing distribution so the more recent samples are more important to me than the older samples things are changing my coin is varying out or time when more recent samples are more important to me than the older samples right. So what do I do in such cases is, but $\alpha < 1$ so you can do a little bit of expanding out on this and you can convince yourself that the as time passes by the older rewards that I get will get exponentially lesser weight right.

The first time I do this right the new reward will get in a weight of α right, but second time I get another reward what will happen I will get a weight of α^2 going in there for the older reward, the new reward will still get an α the older reward will get an α^2 right. So like that so I will be paying more and more attention to the newer rewards right, and the older rewards will keep getting decay.

And already it was very simple change instead of having 1/n constant alpha, so making it a constant α actually allows me to pay more attention to recent rewards and less attention to the older rewards right. And depending on how quickly your rewards are changing you can tune the value of α right. So problem with keeping α fixed is what, if it is stationary that you not converge if it is stationary you still not converge because you are still paying attention to the more recent rewards right.

And we will keep oscillating around the true reward right, so whenever you get a 1 you will go up, whenever you get a 0 will come down up down, up down we keep going like that right because you are paying more attention to the recent rewards. But the advantage of keeping α constant is that if it happens to be non stationary right, you take it to some extent there are other more more robust ways of tracking non-stationary problems but this is very simple trick that you can use right.

And more often than not we will actually end up using a constant α in many of the algorithms that we look at and but whenever we talk about theoretical convergence will always talk about α decay right whenever it look at practical implementations of this algorithms will always have a constant α right. So the problem is decaying α is 1/n is again the same thing which I told you where decaying ∑ right the changes will become small very fast right. So that is always a trade-off that you have to right.

IIT Madras Production

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

www.nptel.ac.in

Copyrights Reserved