STOCHASTIC APPROXIMATION: THEORY AND APPLICATIONS

Dr. Gugan Thope

Department of Computer Science and Engineering

Indian Institute of Science, Bangalore

Week 13

Lecture 48

Q-Learning with Linear Function Approximation under \epsilon-Greedy Exploration

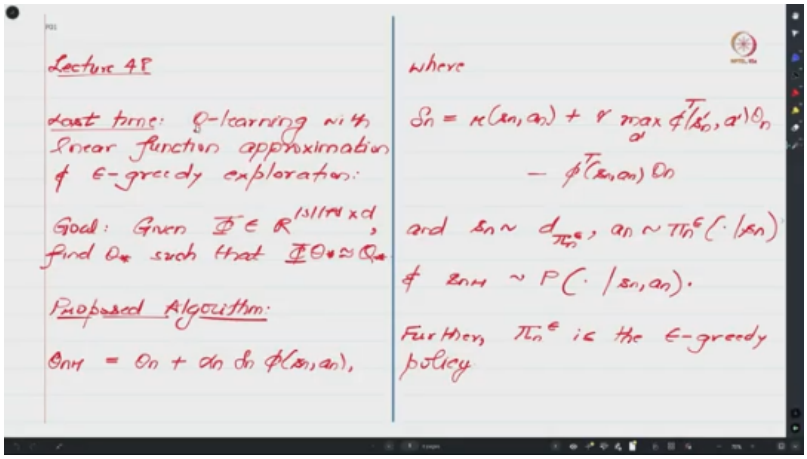
Hello and Namaste everyone. Welcome to lecture 48 of this NPTEL course on Stochastic Approximation. This week, we have been looking at some advanced topics related to applications of Stochastic Approximation in the context of reinforcement learning. In particular, we have been focusing on understanding the behavior of Q-learning with linear function approximation and epsilon-greedy exploration. In the case of policy evaluation, we saw that the TD(0) algorithm with linear function approximation is guaranteed to converge under some reasonable conditions, and we saw that the limit point was close to

you know, the projection of V pi onto the function approximation space, in particular, V pi onto the projection of V pi onto the column space of your phi matrix. So, then we used some of the ideas from there and we developed our Q-learning algorithm for the tabular setting, and in the tabular setting as well, we saw that the Q-learning algorithm, when the behavior policy is fixed, right, converges to Q star, which is the Q-value function of your optimal policy. And you know, things again, this convergence was guaranteed under reasonable conditions. And in the previous class, we saw or we considered the setup of Q-learning with linear function approximation, but instead of you know interacting with the environment using a fixed behavior policy, we followed the idea that is practiced in that is used in practice or applications, which is that of interacting with the environment using an epsilon-greedy policy. The idea is, whatever is your current estimate of Q star, which is formally denoted as phi theta n, you try to identify the greedy policy with respect to that estimate and then be epsilon-greedy according to that greedy policy, which

means that with 1 minus epsilon probability you act according to this greedy policy and with epsilon probability you act randomly.

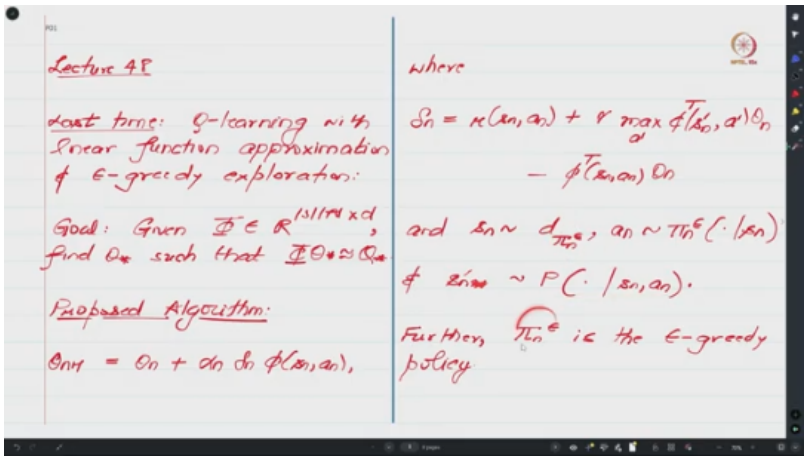
And we wanted to understand the behavior of this algorithm, and in the last few slides of the previous class, I showed you some pictures which showed some very interesting behavior. That is, if you start the algorithm from some initial point, on different runs of the experiment, you converge to different places. In particular, we saw that on one of these runs, the policy or the greedy policy associated with these iterates keeps oscillating without stopping. And we saw this very interesting behavior, and we wanted to understand why that happens. In this class, we will give a formal description of why that is happening; in particular, we will try to you know help understand or come up with a technique to understand the asymptotic behavior of such algorithms.

So with that, let us begin with the formal discussion. So, as I said, we are studying Q-learning with linear function approximation and epsilon-greedy exploration. So, the idea in function approximation, in particular linear function approximation, is that we have been given this matrix phi whose number of rows equals the cardinality of the state and action spaces and the number of columns equals d. And in this, I mean for this given phi, the goal is to find a theta star such that phi theta star is approximately equal to Q star, where recall Q star is the Q value of the optimal policy. And towards that, we propose this algorithm, which is theta n plus 1 is theta n plus some step size times delta n times phi of SnAn, where delta n is your TD.



TD error or the temporal difference error and is described as follows. It is the immediate reward plus the approximation of Q star or the current approximation of Q star at state SN prime and you take the max according to that. Minus the, you know, current approximation of Q star at state SN AN. So this is the approximation of Q star at SN prime A prime and this is the approximation of Q star at SN AN and one can think of this expression again as some approximation to Q star SN AN and this is another approximation of Q star SN AN and in this way one can see that we are taking the difference between two approximations of Q star SN AN and that is why this is known as the temporal difference error and unlike the TD0 setup

Right here Sn is sampled according to the stationary distribution that is associated with your epsilon-greedy policy and An is again chosen according to your epsilon-greedy policy and the next state again I should emphasize this is not Sn plus 1 rather it is Sn prime which is present over here. You can see that Sn prime is present. So this Sn prime is presumed to be sampled from your transition kernel. And recall that pi n epsilon is the epsilon-greedy policy. In other words, your pi n is a greedy policy.



With respect to phi theta n and pi n epsilon is 1 minus epsilon pi n plus epsilon pi n right.

$$\Phi \in R^{|s||\rho_d \times d|}$$

$$\Phi\,\theta \approx Q_*$$

$$\theta_{n+1} = \theta_n + \alpha_n \delta_n \phi\left(s_n, a_n\right)$$

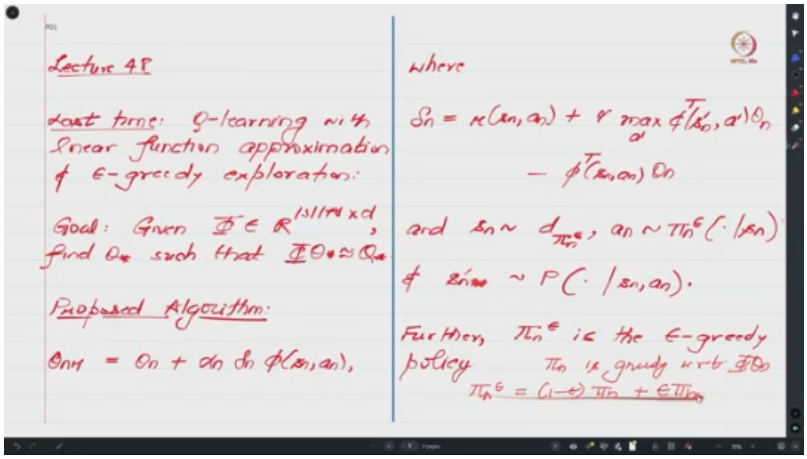
$$\delta_n = r(s_n, a_n) + r\phi^T(s_n', a')\theta_n - \phi^T(s_n, a_n)\theta_n$$

$$s_n \sim d_{\Pi_n}$$

$$a_n \sim \Pi_n^{\epsilon}(s_n)$$

$$s_n' \sim P(s_n, a_n)$$

So, this is the setup that we considered and as I told you, this algorithm had some strange behaviors. In particular, we had considered a two state two action environment right and we were using a two dimensional linear function approximation. And we saw that if you start even very close to the representation for the optimal Q-value function—in particular, we had chosen the phi matrix in this example to ensure that it is perfectly realizable, which means that the Q-star actually lies in the column space of phi. We had ensured that by keeping one of the columns of phi to be Q-star. Right. And so this is that point.



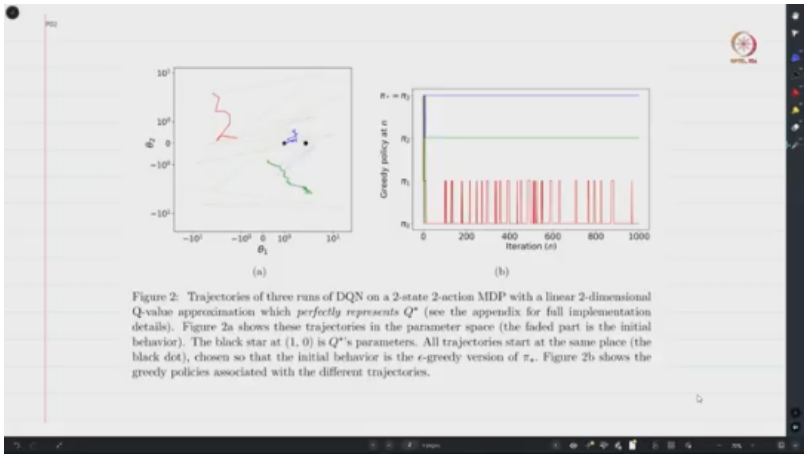
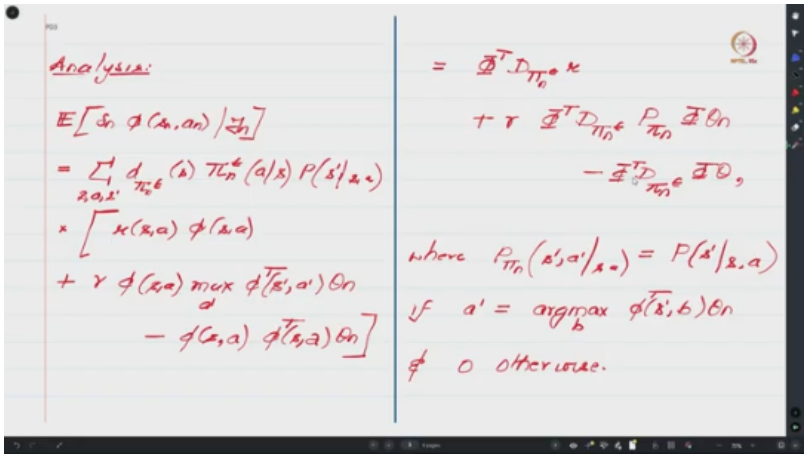


Figure 2: Trajectories of three runs of DQN on a 2-state 2-action MDP with a linear 2-dimensional Q-value approximation which *perfectly represents* Q^* (see the appendix for full implementation details). Figure 2a shows these trajectories in the parameter space (the faded part is the initial behavior). The black star at (1, 0) is Q^*'s parameters. All trajectories start at the same place (the black dot), chosen so that the initial behavior is the ϵ-greedy version of π_*. Figure 2b shows the greedy policies associated with the different trajectories.

So this is 1, 0. Right. And this point corresponds to the I mean, if I multiply phi with 1, 0, you will get the first column of phi. And we had said that to be Q star. And one can see that we choose a theta 0, which is very close to this.

You know, the ideal place where we would like to go, but despite starting so close to this, one can see that the noise pushes you very far away often and along different runs of the experiment you sort of reach different points. The blue trajectory ended up reaching the place where we would have liked. The green trajectory, you know, sort of settled somewhere else and the red trajectory also sort of settled somewhere else. And the interesting part of the red trajectory is that if you look at the greedy policy that is associated with the phi theta and estimates along this red trajectory, the greedy policy keeps jumping between pi 0 and pi 1 and surprisingly none of them are optimal, right? So it keeps jumping between pi 0 and pi 1.

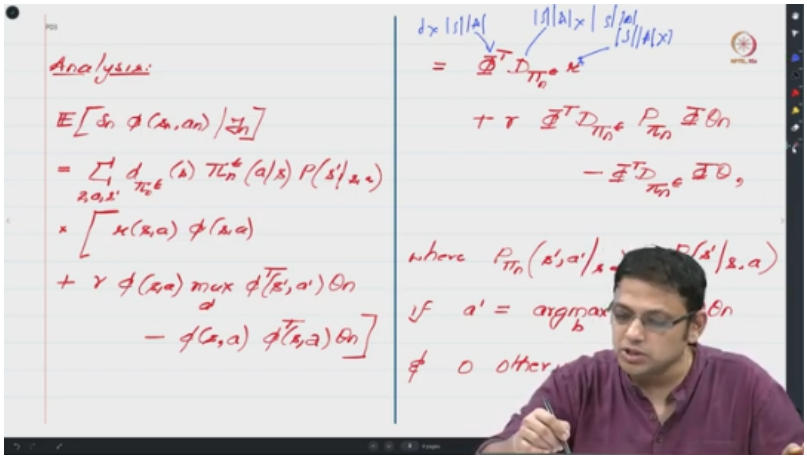
And you know if you are in an applied setup and if you see such a behavior you may want to know you know should I choose pi 0, should I choose pi 1 and so on. But here we observe that neither pi 0 nor pi 1 are optimal but you know the greedy value associated with your phi theta actually keeps oscillating and it does not stop. You can see that the oscillation continues throughout our experiment. So the question that we had is why does this happen? Can we provide a rigorous explanation of why this is happening?



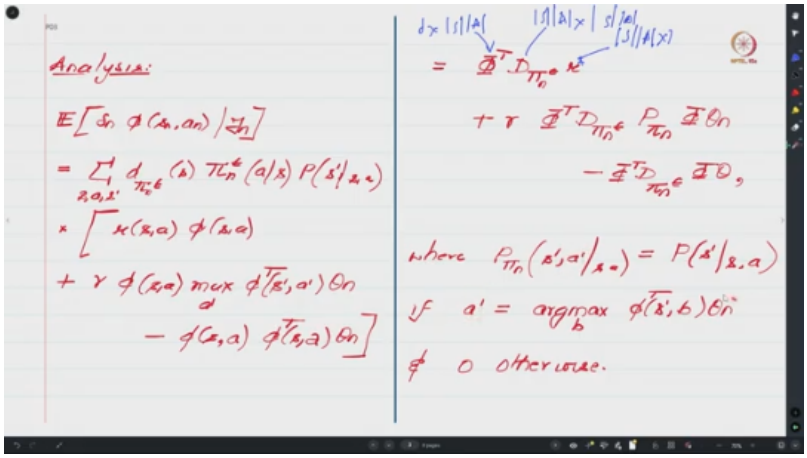
So towards that let us begin the formal analysis. So as we had done previously what we will do is we will take this expression that is present in your update rule for your Q learning with linear function approximation and take its conditional expectation. And making use of the formulas for delta n and phi s n a n and also recalling the way delta like s n a n and s n prime are sampled, one can see that this conditional expectation can be represented in the following way. So, we are sampling S according to the stationary distribution associated with the Markov chain induced by this policy pi and epsilon. We pick action according to our epsilon greedy policy pi and epsilon and the next state is chosen according to this transition kernel.

So, by taking the product of these probabilities and writing an expansion of the expression that we have over here in particular the first term in delta n is R S n A n. So, if you presume that you know S n and A n equal S comma A then we would end up with R S a over here and similarly this quantity will translate to phi S a. In the same way, the next quantity will translate to gamma times phi S A times the max of this expression and the last term in your delta n expression when it is multiplied by phi will translate to something like this. And one can now try to compactly write as shown over here. In particular, one can see that This expression times these things in particular by noting that there is no S prime over here one can see that the you know expression that is over here can be compactly written in the following way that is phi transpose times D subscript pi N epsilon times R right here.

This R is a cardinality SA times one vector. This is a SA cross SA matrix and this is a D cross SA matrix. So this is how it is and your D pi n epsilon is actually a diagonal matrix. whose diagonal entries are made up of, so it is a SA cross SA matrix and the SAth diagonal entry is made up of this times this expression. So this product will be present in the SAth diagonal entry or little s comma a diagonal entry of this matrix over here.

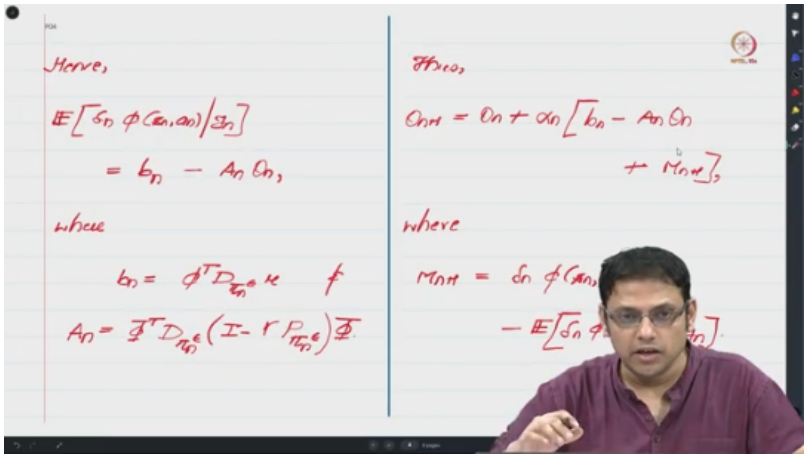


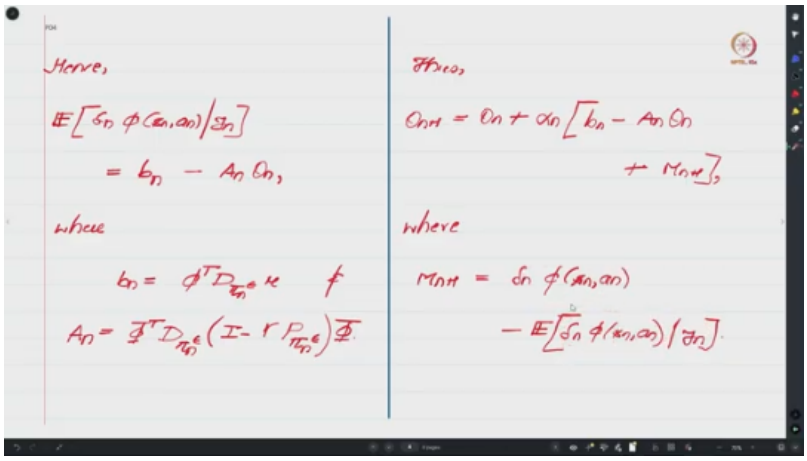
So if you take this expression and multiply it with this one can see that you can compactly write it in this way. Similarly this expression when you multiply it with this way and here you can see that there is an S prime present that leads to some expression like this where you additionally have this P pi n. So P pi n again is a S A cross S A matrix. whose s prime a prime given sa right this entry will equal p of s prime given sa if this a prime over here maximizes this quantity right or it is 0, right. So if you define P pi n in the following way, one can then see that because of the presence of this max expression, right, we can write this times this in the form that is given over here.



Finally, you can see that this expression again does not involve S prime and hence if you take this and multiply it with this, one can see that we would end up with an expression that is shown over here. And again I would like to highlight that because of the dimensions that are mentioned over here, this will be a D cross 1 vector. Similarly, this will be a D cross 1 vector and this will be a D cross 1 vector. In particular, this matrix that is present, this will be a D cross D matrix and this will be a D cross D matrix. Is this okay?

All right. And given this expression that we saw, it is not difficult to see that the conditional expectation of this expression can be written as Bn minus An theta n, where Bn is given by this expression and An is given by this matrix. And consequently, we can say that the theta n update rule can be succinctly written as theta n plus 1 is theta n plus some step size times b n minus a theta n plus m n plus 1, where m n plus 1 is the true update minus the conditional expectation. And in the previous slide, we saw that the conditional expectation is exactly equal to this. So one can see that this exactly equals the true update that we have been studying so far.





So, this is what the Q-learning with linear function approximation and epsilon-greedy exploration looks like. In a formal sense, this is what it looks like. In particular, the thing that drives it has the nature that is given over here.

$$E\left[F_n\right] = b_n - A_n \theta_n$$

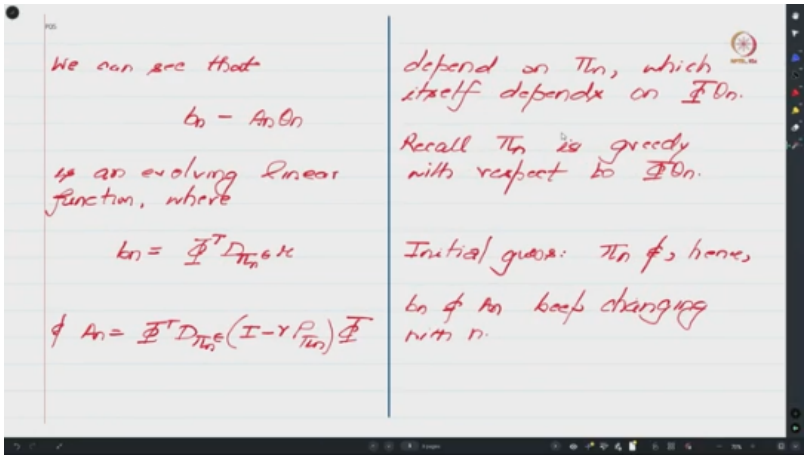
$$b_n = \Phi^T D_{\Pi_n^{\epsilon} \mu}$$

$$A_n = \Phi^T D_{\Pi_n^{\epsilon}\left(I - \gamma P_{\Pi_n^{\epsilon}}\right)\Phi}$$

$$\theta_{n+1} = \theta_n + \alpha_n\left[b_n - A_n \theta_n + M_{n+1}\right]$$

$$M_{n+1} = \delta_n \phi\left(s_n, a_n\right) - E\left[F_n\right]$$

So, at first glance, one can see that this Bn and An depend on n. By depend on n, I mean that, as your theta n changes, your phi theta n changes.



And because your phi theta n changes, your pi n also changes. And because your pi n is changing, pi n epsilon will also change. And consequently, your b n and a n actually depend on the value of phi theta n. So, while this may appear to be linear in nature, the challenge in the analysis stems from the fact that this Bn and An actually depend on phi theta n, which means that as your theta n changes, Bn and An also change with them. And this is what sort of complicates the analysis.

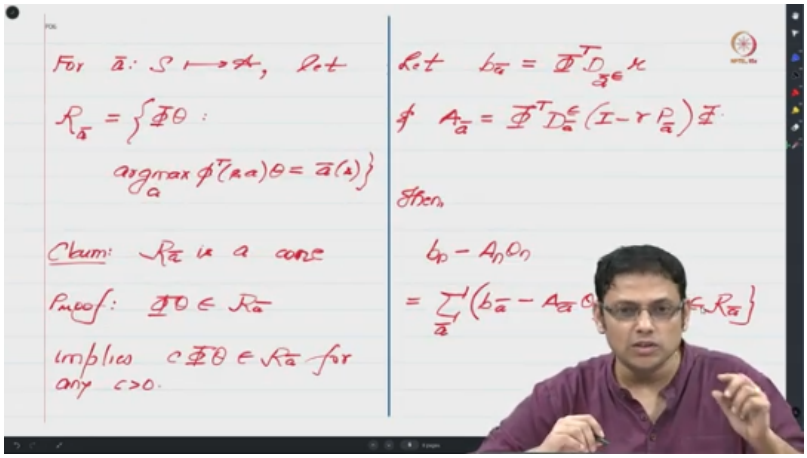
And I would like to again highlight that this pi n is greedy with respect to phi theta n. So, as I said, the summary of this whole discussion is that Because of these reasons, one can guess that because pi n and hence b n and a n are changing, the analysis of this algorithm will be very, very difficult.

$$b_n - A_n \theta_n$$

$$b_n = \Phi^T D_{\Pi_n^{\epsilon} r}$$

$$A_n = \Phi^T D\_\Pi_n^{\epsilon}\left(I - \gamma P_{\Pi_n}\right)\Phi$$

And now we will see how to tackle this. So towards that, what we will do is we will introduce a few notations. The first of these notations is this a bar notation.



So, when I write A bar, I am going to imagine this to be a function from the state space to the action space. So, this function takes as input a state and spits out one deterministic action. So, when I say A bar, it is a function which takes as input a state and spits out an action. And what I will do is for any such function a bar, one can also view this a bar as a deterministic policy. Deterministic policy meaning whenever you see s you act according to a bar of little s. Whenever you see state s you act according to a bar of little s. So in that sense, this policy whenever it sees a state only picks a particular action and hence I am referring to this a bar as a deterministic policy.

And corresponding to this deterministic policy A bar, one can define a region in the following way. So what is this region? It consists of all those vectors of the form phi theta. So theta is d dimensional, phi A is S A cross d, hence phi theta is will be a vector that lives in the SA dimensional space where S and A you know are capital S and capital A and hence when I say SA dimensional space it lives in the space that corresponds to the product of the state and action spaces.

So this lives in a very high dimensional space. So it is the collection of all those vectors in this high dimensional space whose greedy policy is actually A bar. So, I should perhaps emphasize this should be true for all S in capital S. So, what I am saying over here is R A bar is the collection of all phi thetas whose greedy policy is A bar.

More formally, if you look at phi transpose SA theta, you look at this inner product and look at the action that maximizes this inner product for a fixed state S. So, you are collecting all those phi thetas where when you try to do this maximization, you end up

with A bar S. So and if this happens for all S you put that phi theta in this region and RA bar is made up of such phi thetas. We will refer to this RA bar because of this nature as the greedy region that is associated with your A bar policy. So given a policy deterministic policy A bar we will refer to RA bar as the greedy region associated with your policy A bar. Now our claim is that this region is a cone, right?

That is our claim. So what do we mean by a cone? By a cone we mean that if RA bar contains a vector, then if you take any scalar multiple of that vector, in particular a positive scalar multiple, not arbitrary scalar. So if you take a positive scalar multiple of this vector, then that resultant vector is also lying in that region, okay? That is the interpretation of cone that I am going to use.

So, the claim is that for any deterministic policy A bar, your R A bar is actually a cone. So, let us just verify that. So, suppose phi theta lies in R A bar. Now, phi theta lies in R A bar implies that A bar is greedy with respect to phi theta. In other words, this condition holds true for all little s in capital S.

And now if I multiply this vector phi theta by this positive constant c, then one can see that the ordering of the different coordinates of phi theta, that ordering will not change. Of course the value will change, but the ordering will not change. By ordering I mean that if there was some coordinate of phi theta, that was larger than some other coordinate, then that same coordinate will continue to be larger than the previous coordinate even after multiplying by c. This is because since c is positive, so I mean the broad idea I am saying is that if you have two real numbers x and y and if x is bigger than y and if I multiply both sides by c, then cx will continue to be bigger than cy. That is the idea that I am using to conclude that

If you have phi theta in R A bar, then C phi theta is not, since you are multiplying each coordinate of phi theta by C and C is positive, right, the ordering of the coordinates of phi theta is not going to change. Consequently, If phi theta is in R A bar which implies A bar is greedy with respect to phi theta, one can conclude that A bar continues to be greedy even with respect to C phi theta which implies in turn that C phi theta will belong to R A bar for any C bigger than or equal to 0, strictly bigger than 0. And from this, in fact one

can show it is greater than or equal to 0 as well. And from this one can see that your phi theta lying in R bar implies C phi theta lies in R A bar for every C bigger than or equal to 0 and hence one can conclude that R A bar is a cone.

So the summary is that if you define a greedy policy in this way, And if you look at the, sorry, I should say it again. If you define a deterministic policy in this way and you define the greedy region associated with this deterministic policy in this way, then this greedy region one can immediately show is actually a cone. So now define BA bar to be this vector over here. So this vector is very similar to your BN but the difference is that

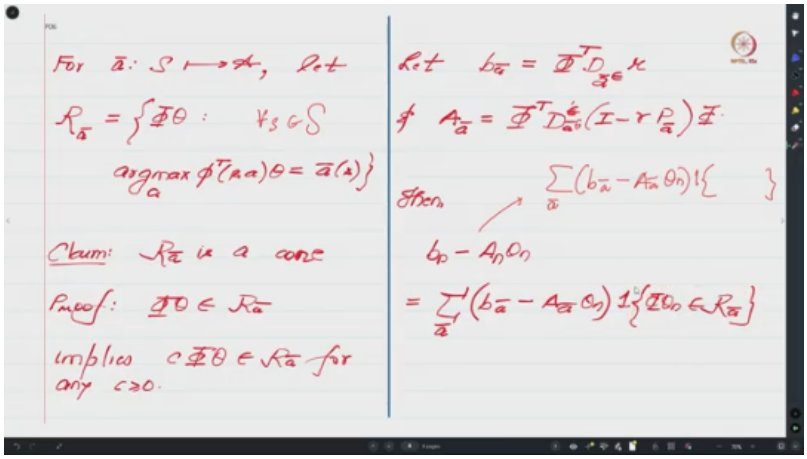
Here instead of having pi n epsilon we have a bar epsilon. So what is a bar epsilon? It is the epsilon randomization of a bar which means A bar is a deterministic policy with 1 minus epsilon probability you act according to A bar and with epsilon probability you pick a random action. That is what A bar epsilon is.

So this is again a stochastic policy. You look at the stationary distribution that is associated with this stochastic policy. And then you know define this matrix over here. Again this is a SA cross SA matrix whose little SAth entry is basically the product of the stationary distribution at state S associated with this policy and the probability of the action under this stochastic policy. That is what this matrix would be and accordingly one can see that BA bar is defined in this way.

And similarly, capital A bar is defined in this way. Again, this has a very close structure or similarity to the A N matrix that we saw before, except that there, you know, this quantity and this quantity were pi N epsilon and pi N respectively. Here instead, we work with, I should perhaps clarify that this is A bar epsilon. So, here instead of having pi N epsilon and pi N, we have A bar epsilon and A bar. So, in this way you define this B A bar and A A bar.

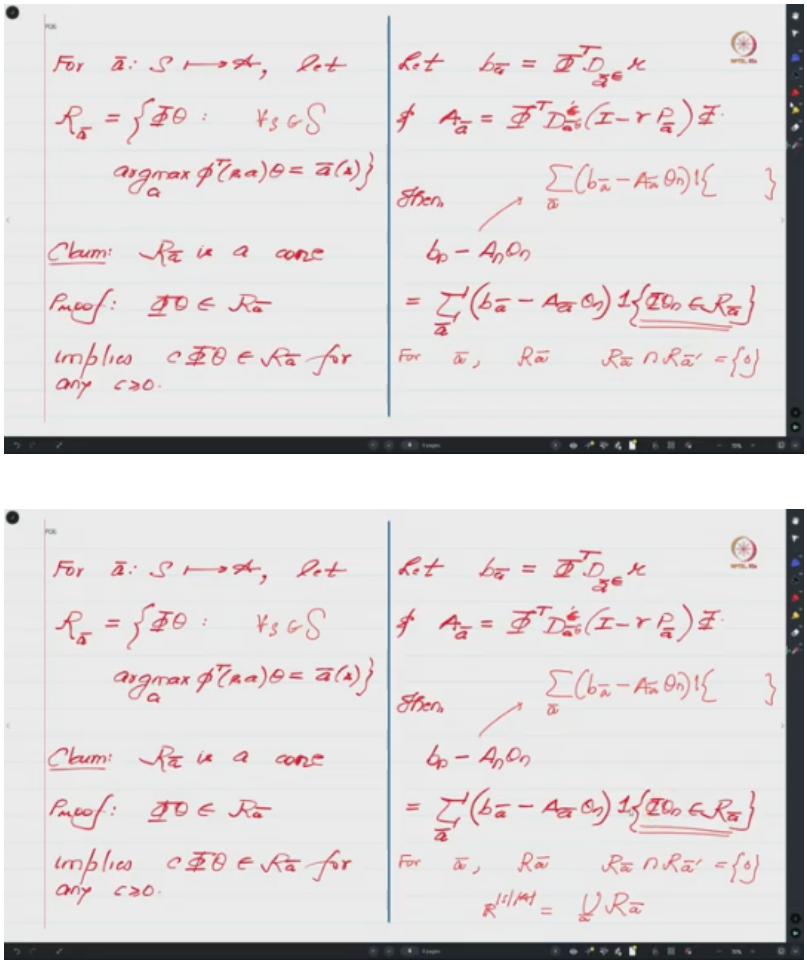
Then one can see that B N minus A N theta N can actually be written in the form that is given over here. In particular, one can see that B N minus A N theta N can be expressed as a sum of B A bar minus A A bar theta N. So, B N minus A N theta N can be expressed as sum over A bar. B A bar minus A A bar theta N along with an indicator, so what is this indicator? The indicator over here is phi theta N belongs to R A bar, in other words.

Whenever your theta N is such that it belongs to R A bar, right? Then one can see that the B N minus A N theta N will actually be B A bar minus A A bar theta N, okay?



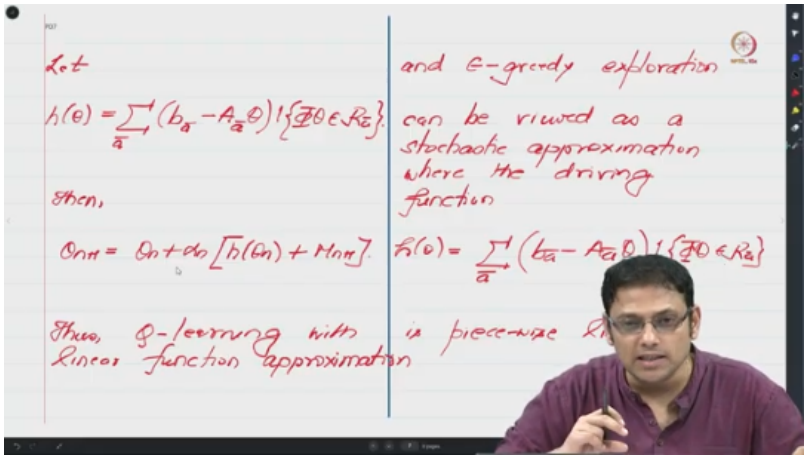
So, I would like to first emphasize a bit why we can write it in this following way, right? So, for every deterministic policy A bar, we can come up with a greedy region associated with A bar. And one can show that if you have two different greedy regions except for the origin. If you sort of decide some rule to, you know, break ties and so on, one can see that except for the origin, okay, these two regions will actually be disjoint, right? So, except for the origin, these two regions will actually be disjoint, right?

So, in other words, the greedy regions will be pairwise disjoint and one can show that the whole RSA space, okay, the whole RSA space can be written as a union of these greedy regions, right? So, whenever you have a phi theta n which is not equal to the origin, one can show that phi theta n must lie in R A bar for some A bar, right? So, accordingly, and in fact, one can show that when phi theta n is not the origin, right? one can show that there is exactly one A bar, right? So there is one A bar and that A bar is unique such that phi theta n belongs to R A bar. Again, I am presuming there is some way to break ties and, you know, if you sort of think a bit closely, you will be able to see why I require this breaking of ties and so on, right? But once you have a rule to break ties, one can conclude that phi theta n



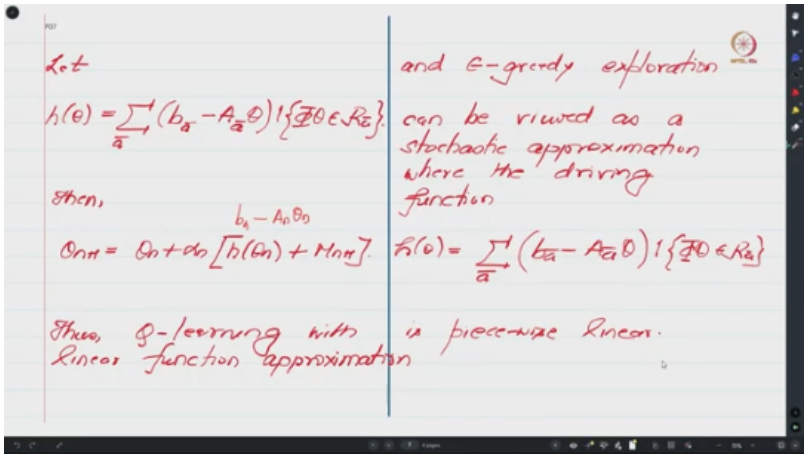
So for every theta, or every theta n, there should be a unique A bar such that phi theta n lies in that R A bar. In other words, this indicator, whenever phi theta n is not 0, will be 1 for some A bar, and you know there will be a unique A bar for which this happens. In other words, we can write bn minus An theta n in the following way. Now, this is very interesting because when you have finite state and action spaces, one can show that the number of deterministic policies is finite, which means the number of greedy regions is also finite, and one can think of this as a partition of this space into the different greedy regions. And what this says is that when you are, you know, phi theta n or theta n lies in a particular greedy region, right, or phi theta n lies in some particular greedy region, right, then the dynamics Bn minus An theta n can be expressed in the way that is shown over here.

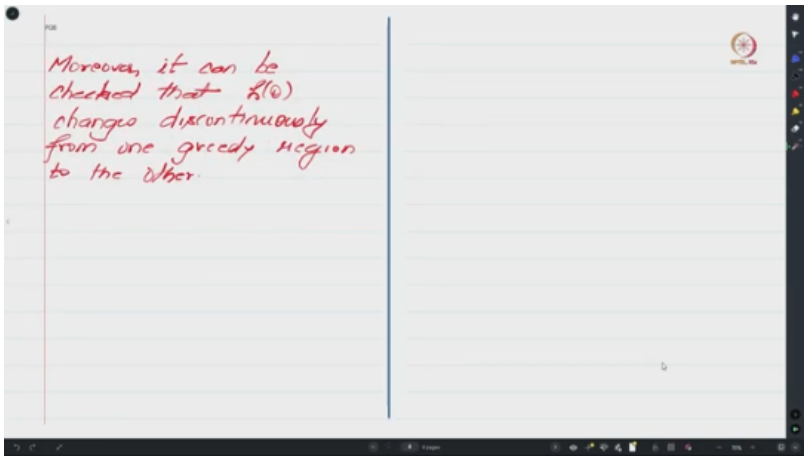
So accordingly, what we are going to do is we are going to define h of theta in the following way. This is sum over A bar of b A bar minus A A bar theta indicator phi theta belongs to R A bar. Then one can see that the update rule for Q-learning with linear function approximation can be written as theta n plus 1 equals theta n plus alpha n times h of theta n plus m n plus 1. So before we had this expression bn minus A theta n, right?



So, when we looked at that expression, it appeared that your bn and an keep changing with respect to every n, right? However, what this description of s and this update rule that we have written over here suggests that you know, when your phi theta n lies in a particular greedy region, then you follow this linear dynamics. And when your phi theta n moves from one greedy region to the other, then the linear dynamics also changes. Is this okay?

So this is the key observation that we need to make. In other words, one can note that or formally one can see that this h of theta that I have defined over here is actually piecewise linear. That is, in one greedy region it has some linear nature, and when you move from one greedy region to the other, then the dynamics also changes accordingly. And then, the surprising or the challenging aspect of this definition of H is that this H actually changes discontinuously when we move from one greedy region to the other. So, this brings us to the end of this class.





Let me quickly summarize what we have done over here. So, in this class, we tried reading Q-learning with linear function approximation and epsilon-greedy exploration. And we wanted to analyze the behavior of this algorithm. In particular, in the simulations, we saw some strange behavior, right? And we wanted to understand: can we say something about, you know, why this asymptotic behavior is happening? Can we explain something towards that?

So, in that direction, what we did was we tried, you know, taking the conditional expectation, which is the standard thing we do, of the update and, you know, adding and subtracting them, and we saw that the update rule is of the form b_n minus a_n theta_n, right? And from the definition of b_n and a_n, we saw that To begin with, it appeared that b_n and a_n both change with your iteration n. In particular, they depend on phi theta_n. So, as your phi theta_n changes, your b_n and a_n also change. But then we

observe that if we define these greedy regions associated with these different deterministic policies, then we observe that this b_n a_n theta_n, this dynamics that governs this Q-learning with linear function approximation, does not change arbitrarily. The behavior is that whenever you are in a particular greedy region, it is the linear dynamics that is B_A bar minus A_A bar theta associated with that greedy region that governs your dynamics. And when you move from one greedy region to the other, there is a different dynamics that governs your behavior.

And in the next class, we will see how this changing dynamics from one greedy region to the other affects the behavior of the algorithm, and we will see how this discontinuously changing, piecewise linear function can be used to describe the different behaviors that we saw in those empirical simulations. So, until then, goodbye and namaste, and thank you.