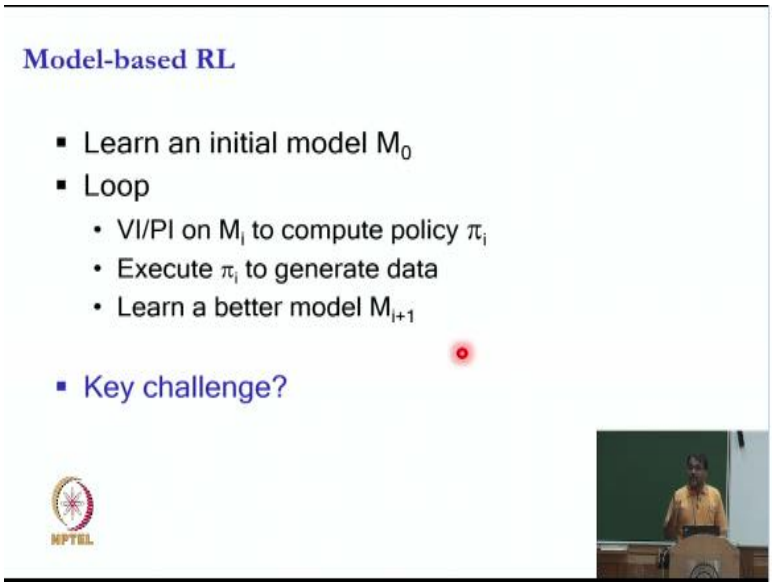
Artificial Intelligence

Prof. Mausam
Department of Computer Science and Engineering
Indian Institute of Technology-Delhi

Lecture - 81
Reinforcement Learning: Q Learning

So now model-based RL is going to be very simple, right? It is exactly how it was in model-based policy evaluation. The idea is that you have some initial model M naught.

(Refer Slide Time: 00:30)



Model-based RL

- Learn an initial model M_0
- Loop
  - VI/PI on M_i to compute policy π_i
  - Execute π_i to generate data
  - Learn a better model M_{i+1}
- Key challenge?

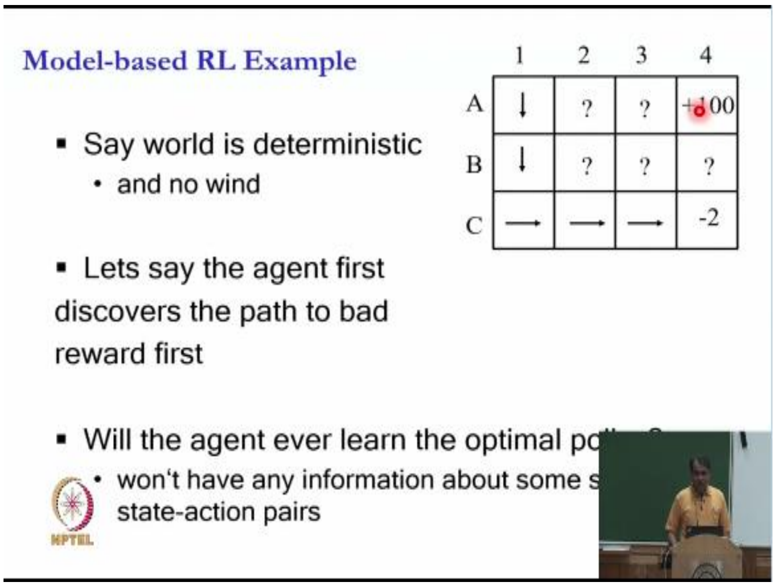
This is my MDP, right. So I have some transition function initially it could be uniform or whatever. I have some reward function and then I will be looping as follows. So I do value iteration or policy iteration on M i that gives me policy pi i. Then I execute pi i to generate more data. Now I generate more data so I learn a better model M i + 1 and I repeat. Everybody, okay with the formulation, the model-based RL version, right?

Because I need to estimate transition and reward. If I take all the random actions forever to estimate transition and reward, then I will never move to actually optimizing the reward, right. I also have to optimize the reward because I want to live my life and you know live it happily, not just computing transitions and rewards. So I will do it iteratively.

I will use the current knowledge to get the next better policy than the policy to optimize some reward and get more data and then repeat so that I can get a better model and then a better policy and this sort of goes on. Now can you figure out a problem with this setting? What is the technical problem? What is the technical problem with this way of defining the world?

"Professor - student conversation starts" Exploration is very less. What is your name? Ashray. Ashray says exploration is very less. Now you are using a word that we have not yet defined. So can you say in simpler words what is exploration? Possible pi i is like I am not going through all the possible pi i's. **"Professor - student conversation ends".** I am not going through all the possible transitions, right? And that may hurt where? That may hurt in learning the better model, okay.

(Refer Slide Time: 02:28)



Model-based RL Example

- Say world is deterministic
  - and no wind
- Lets say the agent first discovers the path to bad reward first
- Will the agent ever learn the optimal po
  - won't have any information about some s state-action pairs

	1	2	3	4
A	↓	?	?	+100
B	↓	?	?	?
C	→	→	→	-2

So let us look at an example. Maybe that gets clearer there. Say the world is deterministic, say there is no wind whatsoever. Let us say there are two exit states 100 and -2. But the model does not know it. Let us say this is my initial policy. So let us say the agent for whatever reason, has figured out that there is a path and every action costs -1.

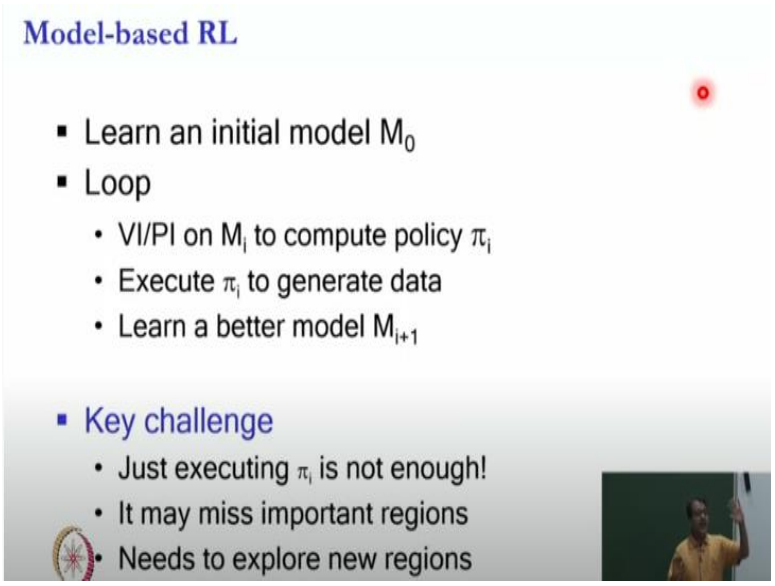
So let us say the agent has figured out a path that it should first go down and then go right and stop after receiving a reward of -2. And let us say this is the only thing that the agent has discovered so far, okay? So it feels that V of A 1 is -1 + -1 + -1 + -1 + -1

+ -2. So that is -7. So it feels like the world is composed of a long-term reward of -7 by if I go down, go right I will get this -2 I will stop, that is my life okay.

So let us say this is the policy I start with. And now I run it. Now when I run it, let us say the world is deterministic, what will happen? Will I ever even try the right action at A1. I am not even going to try the right action. I might not even discover any of these 5, 6 states. I would never discover this high reward of +100 because I never tried to go there. The algorithm says execute pi i.

And if my pi i is this, there is just no way I am going to get to these question mark states or this +100 reward. So therefore, the agent will never learn the optimal policy, because it would not even have information about some of the states or for that matter, some of the state action pairs.

(Refer Slide Time: 04:21)



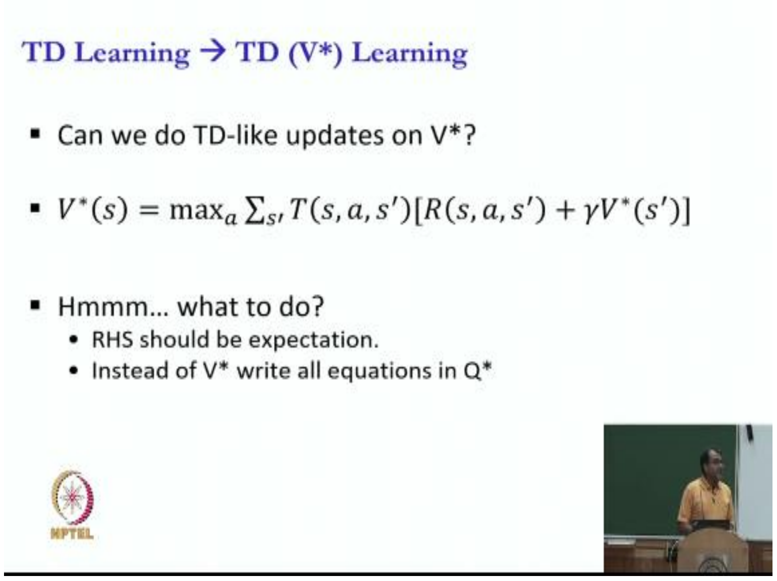
Model-based RL

- Learn an initial model M_0
- Loop
  - VI/PI on M_i to compute policy π_i
  - Execute π_i to generate data
  - Learn a better model M_{i+1}
- Key challenge
  - Just executing π_i is not enough!
  - It may miss important regions
  - Needs to explore new regions

So the key challenge is that just executing pi i is not enough. Because it may miss important regions where pi i does not take me. We need to as Ashray said, explore new regions. And the current way of defining it has not solved this problem, okay. We will come back to this question after we do the next algorithm. So we will talk about exploration versus exploitation in a few minutes, right.

So let us first take the model-free leap and say that we want to go from model-based to model-free. And like we had TD learning what is the equivalent of TD learning for this general version where I do not know a policy.

(Refer Slide Time: 05:10)



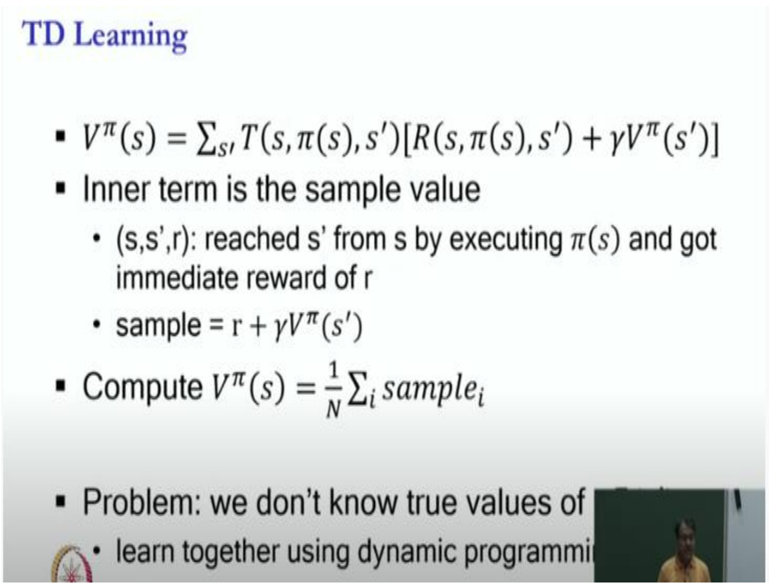
TD Learning → TD (V*) Learning

- Can we do TD-like updates on V*?
- $V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
- Hmmm… what to do?
  - RHS should be expectation.
  - Instead of V* write all equations in Q*

Now the first thing is to write down the set of Bellman equations. And the set of Bellman equations are very simple. V star of s is max over a sum over s prime T s a s prime of the long-term reward estimated as R s a s prime plus gamma V star s prime. These are the Bellman equations. Now is this Bellman equation easily converted to a convertible to a TD like update? Easily. What do we need for a TD like update?

We need the total right hand side to be n. Okay, this is something I want you to answer. We have been doing this for a while. What is what have we learned to compute by taking samples? Expectation. Why did TD learning work? Because the right hand side was an expectation. Come on, you can do this right. So let me remind you what TD learning was.

(Refer Slide Time: 06:19)


TD Learning

- $V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^{\pi}(s')]$
- Inner term is the sample value
  - (s,s',r): reached s' from s by executing $\pi(s)$ and got immediate reward of r
  - sample = $r + \gamma V^{\pi}(s')$
- Compute $V^{\pi}(s) = \frac{1}{N}\sum_i sample_i$
- Problem: we don't know true values of
  - learn together using dynamic programmi

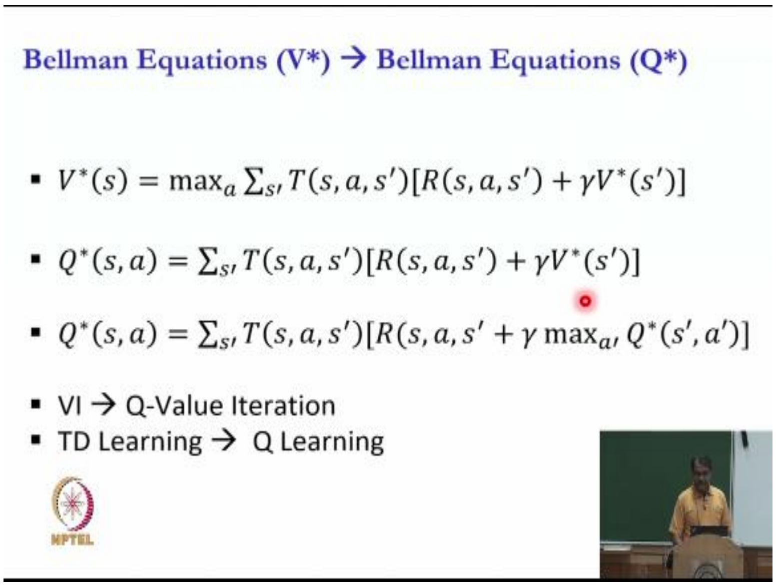
The reason TD learning worked is because my equation the right-hand side was an expectation see sum over s prime, T s a s prime times something. This is expectation with the probability distribution T. Does it make sense? Right, this is an expectation. Unfortunately, in this equation V star, is this an expectation? This is not an expectation. We want right hand-side to be an expectation.

But what we are getting is max over a something. So we cannot convert it into a sampling based algorithm easily. What do we do? This is a big question. Any guesses? Now this will require some insightful thinking. We want expectation on the right at the same time we want to learn the true value of every state. What can we do? Yes Divanshu. Sorry. So you said do some local no, but what are we maintaining?

We are maintaining V star. We cannot take that as an average of a sample. We want to compute expectation as an average of a sample. That is the only thing you want to do. There is nothing more to it. I told you that is the rule in this lecture. We have to somehow convert this equation into an expectation.

"Professor - student conversation starts" Yes Poorva. Sir if we maintain, instead of maintaining V star s you maintain Q star s comma a. **"Professor - student conversation ends".** So Poorva says something very interesting. Instead of maintaining V star of s, let us maintain Q star s comma a. Let us see if that is going to work for us.

(Refer Slide Time: 08:32)



Bellman Equations (V*) → Bellman Equations (Q*)

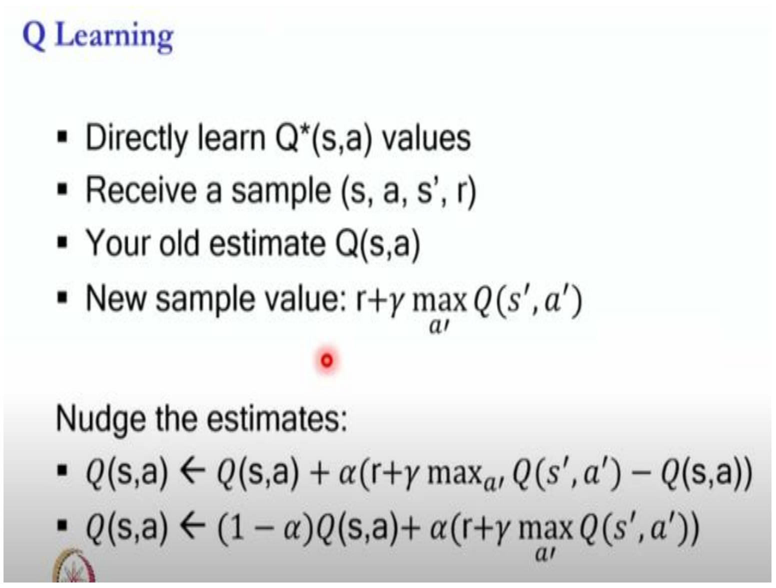
- $V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
- $Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$
- $Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s' + \gamma \max_{a'} Q^*(s',a')]$
- VI → Q-Value Iteration
- TD Learning → Q Learning

So this is V star of s max over a sum over s prime. If we have to write the equation for Q star s comma a, what would that be? Everything without the max, everything without the max that is it. So first of all the first observation that Poorva has is that it is okay let us not compute V star let us compute Q star. Q star is an expectation by definition. However, there is still one challenge, what is the challenge?

In the right-hand side, we do not have Q star. We have V star. This is not a well-defined set of equations. Can we do something about it? No, this is a simpler step. Can we somehow write V star in terms of Q star? Summation over all actions, max over all actions Q star of s prime a prime. So now let us look at this equation. Is this an expectation? Also does it solve our problem? What was our problem?

We need to know what is the right action to do in which state. What is the value of a state? Can we compute the value of state if we are given Q values? Yes. So this is good. So instead of value iteration we can do what we can call as a Q value iteration. Instead of maintaining the V star function, we maintain the Q star function and similarly, instead of doing TD learning, we can do Q learning okay.

(Refer Slide Time: 10:17)


Q Learning

- Directly learn Q*(s,a) values
- Receive a sample (s, a, s', r)
- Your old estimate Q(s,a)
- New sample value: $r+\gamma \max_{a'} Q(s',a')$

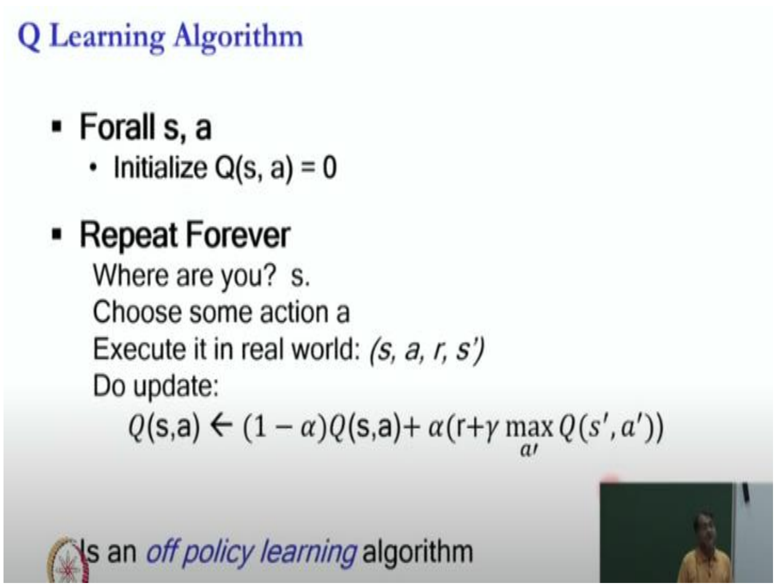
Nudge the estimates:

- $Q(s,a) \leftarrow Q(s,a) + \alpha(r+\gamma \max_{a'} Q(s',a') - Q(s,a))$
- $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r+\gamma \max_{a'} Q(s',a'))$

So that is directly learn Q star values. Receiver a sample. So what is my sample? I am in state s, I take an action a, I get to state s prime, I get an immediate reward r. This is my sample. My old estimate was Q s a. My new estimate value will be r plus gamma times max a prime Q s prime a prime. See here, this becomes my new estimate.

And now I nudge the estimates the old estimates in the direction of the new estimate, which basically means the new value of Q s a is the old value of Q s a plus the learning rate times the new sample minus the old value and the new sample is r plus gamma max a prime Q of s prime a prime or equivalent a 1 minus alpha and alpha versions. And that is the equal thing. This is it, this is Q learning. It is very simple. Now having built up the technical knowhow.

(Refer Slide Time: 11:21)


Q Learning Algorithm

- Forall s, a
  - Initialize Q(s, a) = 0
- Repeat Forever

  Where are you? s.

  Choose some action a

  Execute it in real world: *(s, a, r, s')*

  Do update:

  $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r+\gamma \max_{a'} Q(s',a'))$

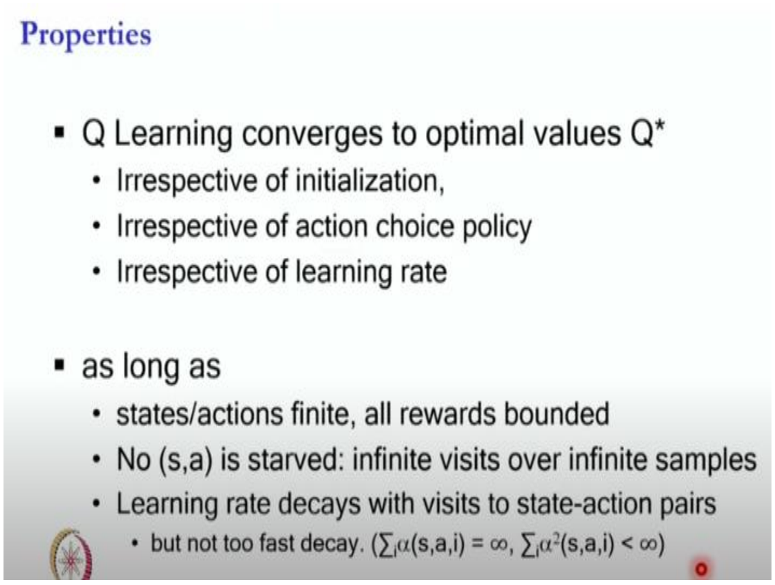
Is an *off policy learning* algorithm

So in practice, what happens is that you initialize all the Q values somehow let us say zero or whatever, and then you repeat forever. So you check where are you. You are in this state s. You choose some action a. You choose some action a you execute it in the real world. So what does the real world give you? The real world tells you what is the state s prime you reached, what is the immediate reward you achieved?

And then you make one update. You update the value of Q s a using this particular equation. And this is called an off policy learning algorithm. Off policy learning algorithm means that the action a that I take here. Instead s prime has nothing to do with how I change the value of s comma a. I change the value using the max over a prime but I do not have to take a prime instead s prime, I can take any action.

The learning happens off policy. Policy says take a specific a prime instead s prime. The learning does max over a prime anyways, so it does not care, okay.

(Refer Slide Time: 12:26)



Properties

- Q Learning converges to optimal values Q*
  - Irrespective of initialization,
  - Irrespective of action choice policy
  - Irrespective of learning rate
- as long as
  - states/actions finite, all rewards bounded
  - No (s,a) is starved: infinite visits over infinite samples
  - Learning rate decays with visits to state-action pairs
    - but not too fast decay. ($\sum_i \alpha(s,a,i) = \infty$, $\sum_i \alpha^2(s,a,i) < \infty$)

And the algorithm converges to optimal values irrespective of how you initialize, irrespective of what action you choose in a state. This is very interesting, irrespective of your learning rate, as long as some constraints are satisfied. So this is where I am going to tell you what kind of alphas are allowed. So first of all, state actions need to be finite, rewards need to be bounded just to make sure that we are not working with infiniteness somewhere.

Second, no s, a is starved. This is the kind of thing that we have been talking about in other algorithms as well. If you do not execute s, a often enough then you would not know their value. And if you do not know its value, it could be the optimal the algorithm will not converge. So basically we say that in infinite steps, each s, a pair will be backed up infinite times, will be visited infinite times, okay.

And secondly, you do not only need 1 by n, you can do any alpha as long as it follows the following principle. So alpha s, a, i means that ith time you visit you take action a in state s, the learning rate, sum over i it should sum to infinity. So 1 plus half plus 1 by 3 plus 1 by up to infinity is infinity, right. So therefore, it should be unbounded. The sum over learning rates over the various time steps should be unbounded.

But the sum of square of learning rates over infinite times should be bounded like 1 plus 1 by 2 square plus 1 by 3 square plus 1 by 4 square, that is bounded. So if some learning rate alpha i satisfies these two principles, then this particular Q learning algorithm is guaranteed to converge to the true values, okay. So I am going to stop here for today.

But just to tell you where we are going, I have just discussed the Q learning algorithm. In the next class, I will quickly remind you what is the Q learning algorithm, but then we will talk about the one thing that we have not talked about. What is the thing that we have not talked about in this algorithm? What is the missing step here? Guess. Why, why is max not known? You are maintaining the values of Q.

You are maintaining the Q table. So you can just do max over a prime Q s prime a prime. What is not given to us? How to choose the actions. See this is, this is the step which we have not described so far, how to choose? We do not know. Choose some action, which action? This is what we are going to talk about first.

Then we will talk about generalization, which will lead us towards function approximation, which will lead us towards deep learning, which will be topic of the next week. So I will stop here.