Course Name: INTELLIGENT FEEDBACK AND CONTROL

Professor Name: Leena Vachhani

Department Name: Multi-disciplinary primarily for Mechanical, Electrical, Aerospace and Chemical engineering streams

Institute Name: Indian Institute of Technology Bombay

Week - 04

Lecture - 24

Hi, in this video we will look into an example of inverted pendulum. And we'll consider control using the reinforcement learning. In this case, we will consider that there is no control framework being set, as in we don't know what kind of control we are going to put up, as in there's no structure like PID exists here. So here we will set up an objective of developing a model-free reinforcement learning agent capable of stabilizing a nonlinear pendulum in a simulated environment. Let's understand what this particular statement talks about.

Let's first see what is the inverted pendulum we are talking about. The inverted pendulum, the pendulum is pivoted somewhere at top. And this in this particular pendulum should get stabilized to its vertically upright direction. So the force that is being applied is at this particular pivot place, which is the tau that is being that is shown here. And it comes into this particular vertical upright position.

We know that this particular equilibrium state, if I perturb very little because of the environment disturbances or anything that very small changes, then this pendulum will come back to its stable equilibrium, which is the downward direction itself. So we would like to consider this exercise as a model-free methodology. Of course, the physics of the inverted pendulum has been studied a lot. We do it. Okay, so the governing equations are ML squared theta double dot plus MGL sine theta plus B theta dot equals tau, right?

Which we can write it in the form of the pendulum dynamics in terms of the state-space representation where we will consider the two states as x1 equals theta and x2 equals

theta dot which is the angular velocity of this particular angle that is shown over here. Remember in the previous example in other videos we showed this theta as the angle between the vertical upright position which the pendulum arm makes. Whereas here we are showing it with respect to the vertically down axis over here. And there's a reason behind it. We'll talk about that.

All right. So now when we are considering this exercise of creating a simulated environment, in this case, since we want to consider making training data out of it, we will consider that this particular governing equation is sitting into the simulated environment. And of course, this particular environment is having some kind of disturbances. Some noise is acting on theta. Some noise is acting on tau is going to be present there.

Noises on theta and theta dot because finally when I am controlling this particular system which has the states theta and theta dot. So theta and theta dot are the system, my system states are theta and theta dot, input is tau and there is an output which is again I am considering theta as the output. Because my theta representation is with respect to the vertically down, my objective is to make theta equal to pi which is my the reference position here, angular position over here. All right. So now when I'm considering this as a system, the RL agent has an environment.

And in that environment, these governing equations are being coded. And I am considering the noise on tau and noise on theta. It is because the measurements are on theta and theta dot. So whenever there is a measurement somewhere in the actual environment, when there is an actual inverted pendulum, somebody is recording this theta and theta dot, whether it's a visual way of measurement or it is encoder-based measurements and what not. Okay, or IMU based measurements here.

Each and every measurement will be corrupted with noise and those noises have to be understood acting as disturbance, acting as noises into the system. This disturbance can also be because of the environmental wind conditions or any other forces acting on the inverted pendulum here. Any other means forces other than gravity. Okay, so this particular simulation environment is giving me the data where I will be able to add noise

and disturbances to the system. And of course, I'm not making any disturbance model or a noise model over here.

All right. The data will be collected based on the observation that I am creating and the action that because for the RL agent, what is the input? Input is action. Output is observation. All right.

That is something we'll have to quote. So, for the simulated environment, the codes will be provided and which considers these parameters, the gravity parameter, the length of the arm, the damping coefficient and the mass of this, right, which you would be able to change it. Let's see when we formulate this problem as an RL agent problem, when we want to solve this problem as a reinforcement learning problem, what all things we need is we need certain way by which this particular simulated environment is getting actions and the observations being collected from the environment. So we have the control challenge over here. We don't have the model.

The RL agent doesn't know the model of the system. So this physics is only to create the simulated environment. The agent must learn to generate effective control commands. All right. So these are the actions being given.

The actions to the environment may or may not be exactly same as the input to the given, right? Input to our system is just tau over here, all right? So that is something we have to be clear on the action to the simulated environment and input control input to the system. These are two different things we are talking about. But the action space may or may not take directly what the control input is.

And then our challenge is to maintain this particular pendulum in an upright direction, stable position, where we do not have the prior knowledge of the dynamics. So be clear on it. Prior knowledge of the dynamics is available to the environment, not to the RL agent. Now we are focusing on developing this particular RL agent. At the same time, we can pose certain requirements for the control objective to be achieved.

And over here, we can consider that the control input is between minus 20 and 20. The system should be robust in initial conditions. From wherever my initial position of theta

is, right, could be 10 degrees, could be 30 degrees, 90 degree, I can start, I can hold this particular theta and from there I am giving, the forces are given. So, the initial condition could be anything, not necessarily only theta equal to 0, which will make the pendulum swing to and maintain a stationary upright position. All right.

So this with given this particular requirement, since this is learning based method, we should be making sure that it is in initial condition independent. It means the simulated environment should take care of generating the data from various different initial conditions so that I am the RL agent is capable of learning it. For the simulation environment, as we spoke about, it incorporates system dynamics with additive noise and simulating real-world unpredictabilities. For example, there are certain forces acting onto the system, which are shown as disturbances, could be a wind coming in, there's a fan running around or many different ways by which the forces are. We have to be considering a realistic environment so that we are not making sure that it's a very large wind and we are making an ambitious attempt to make the inverted pendulum into the vertical direction.

So certain realistic disturbances, which we can say that, okay, we are able to add them, or there are sensor measurements, theta and theta dot, and we are making certain additive noise, adding some additive noise to it, so that when we are doing it in the actual environment and there are certain measurements appearing there, we are able to take care of those because the agent has already learned to handle those noises, noise levels and the disturbance levels. Now comes the design of the RL agent. For designing the RL agent, we have to first consider what are the observations and actions taken or given to the environment. So here we will consider the observation space as just not theta and theta dot, but we will consider theta as being given as (x,y) position, which is cos theta and sine theta. The thing is, the important point over here is that we have to consider these theta, these observation variables within certain range.

And it's always a good idea to consider some normalized range like given here. So, this normalization is something that we would like to do, all right. So, in my previous video, you have seen that my theta representation was with respect to the vertical axis. This was

theta. So, for if I am considering this as theta representation, then my control objective is to make theta equal to 0.

And my theta can remain say between minus pi by 2 to plus pi by 2. I'm just trying to make this situation a little constrained that my theta remains between minus pi by two to plus pi by two. And let's say it doesn't go down as a hypothetical case. To understand what we say, how do we convert this range into the normalized range? So then if my theta is between minus pi by two to pi by 2 and there is a 0 coming in here.

So, this is what is my stable position that is what we want, and this stable position should reflect within the range minus 1 to plus 1 with this 0 mapping to 0. So, this map that we are creating between minus pi by 2, to plus pi by 2 within the range minus 1 to plus 1 can then be done in a very methodological way by saying that okay now since the range is between minus pi by 2 to plus pi by 2, the entire range is of pi then let us shift this to minus pi by 2 plus pi Everywhere we will add this plus pi so 0 plus pi and this becomes plus pi by 2 to plus pi which makes it this as pi by 2 plus pi by 2 this as pi and this as 3 pi by 2 so as soon as we do that what we have now is the theta representation given by this position so this becomes my theta equal to pi position this is theta equal to 0 pi by 2 and 3 pi by 2 and that's the reason we considered theta with respect to the vertically down position, so now my mapping is from minus 1 to plus 1 by dividing the angle Of course, this theta is right now we are saying this as pi by 2.

Now, we have swapped this as well. So, this becomes pi by 2 and this becomes 3 pi by 2. I can consider that as well as I can consider because our convention is always anti-clockwise direction conventions. So, we would like to make sure that this is also swapped. So what is important here is a representation of the state variable.

Again, now I'm not talking about the system state, I'm talking about environment state. So this state representation when we speak about, Here, instead of theta, I'm again making cos theta and sine theta representation because when we play around with angles, angles are always notorious. As we have said, as we could see here in this particular example as well, the proper theta representation where we have the anticlockwise and clockwise representation being clearly taken care of in a very robust way And as soon as

I convert this into the Cartesian coordinate vector or, or a vector form, this becomes easy that as soon as I say theta is this particular angle and I have this plus and minus signs for x and y given, I can understand which particular Cartesian plane I am, and would be able to make the maneuvers or the computations very clearly done here.

That is one reason that we converted this into the observation space variables in terms of X and Y. So in nutshell, when you are dealing with angles, you have to make sure that this angle representation is correct. At the same time, angle representation is in each quadrant is reflecting correctly when you are doing the manipulations or computations in terms of the even in the environment phase or in the RL agent case. All right. So making computations easy as well as making sure that your signs are not making any difference.

It is a good idea to convert into X and Y representation. The third variable is angular velocity theta dot, which is again made sure that it is normalized. So it is between minus and plus one minus one to plus one. So it is divided by maximum value of theta dot. The action space, same way the action space is just torque in this case.

So this tau is normalized to minus 1 by 2 plus 1 by dividing it by maximum of tau. Now comes the reward design. The reward design is now, if we see this particular term, which is theta upon pi minus 1. Again, what we are making sure that if theta is away from its equilibrium position or the stable position that or the desired position, in our case, theta equal to pi is our desired position. So this becomes theta upon pi, which is 1, 1 minus 1, which is equal to 0.

So this particular negative sign is making sure that if I am going away from theta, then it is a penalizing effect. So, reward is all negative terms you could see. What is my desired is giving me zero value. But if I am, as soon as I am going away from zero, this is penalizing it by making a negative sign over here, right? So, this becomes a penalty on a particular action being taken.

So before moving to the second term, we could see that writing it in the theta equal to pi case also made it a easier way of doing it. If theta equal to zero would have been a control objective in our previous objective, then we had to divide theta with zero, which is not a correct way of doing it. There are other ways to represent if theta, if the control objective

is to make a particular variable to go to origin. But in this case, anyway, theta equal to pi. So we could make this kind of penalty calculations done in the reward function.

The second term now. The second term is penalizing for giving higher values of theta dot, means higher values of angular velocity. So aggressive actions are being penalized. Similarly, the third term is about giving very high tau values. Very high tau values are penalized and this is what is a good idea to consider, because even if theta is away, and you are giving a very high torque, there is a chance that you will overshoot and then you will have to come back.

So giving a very small torque. TAUs and small angular velocities would be a good idea or a good control strategy to look at instead of making a very swinging action to stabilize it. And that's why these high torque values and high angular velocities are being penalized in the reward function. Okay, now let's see what comes up. During the training phase, if I consider this kind, the same reward function, which is having three terms and their weights are being summed over to one.

This is something we'll have to consider. During the training phase, let us say if my vertical stabilization from the initial condition of minus 0.74 being considered, then theta gets stabilized to some plus 180 degree plus pi condition here. And you could see that omega is initially high, but then gets stabilized to some value. Similarly, the torque values once the theta gets stabilized to its desired position is within plus minus 10 degrees. Let us take the other example where during the training phase, if I consider a reward which has only two terms, okay?

Two terms, what was the third term? The third term was having the tau by max tau term. All right. So this particular term I am not considering in this example. And in that case, what is the correction that I have to consider?

I have to make this particular weight to 0.5 since there are only two terms, these two weights are adding to one. I can always consider making some other weights, but I have given equal weights in this case. So in that case, if I start from some other initial condition, say, because we have to make the learning method agnostic to the initial conditions, but this particular graph shows some other initial condition, which is minus

2.23. So, if I start from minus 2.23 degrees, where am I settling down to is theta is equal to minus 180 degree, which is minus 5 position, which is equal. Whether I consider plus pi or minus pi, that is something is in a vertically upright condition.

Now, we could see that omega is still within certain values, but torque is now when it is in the stable condition, the torque values are between plus minus 50 degrees. So if I'm not penalizing for the high torque values in the reward function over here, we could see that torque values are limited to a higher values, which are plus minus 50 newton meter, as compared to the plus minus 10 Newton meter in the previous example when I was penalizing for the higher torque values. So in all, as a summary, if I have to consider making the problem formulated in the reinforcement learning language, we'll have to see what the simulation environment is to be. Where can we add noises and disturbances in order to make sure that this fits into the realistic environment case? And then describe what is my observation space and describe what is my action space.

By doing that, we are more or less understanding, okay, this is how my actual environment is to be, actual inverted pendulum in this case is going to function and creating the input-output dataset. So this training phase is making sure that my RL agent learns, and the same RL agent now I can put it to the actual inverted pendulum, and apply this control inputs that is being learned through the simulated environment. In the next video, we'll look into some more idea about how we have designed the RL agent in MATLAB. Those who already know the RL method terminology and have already programmed the RL agents, they can skip the next video. Thank you.

Thank you.