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 - 25

Hi, in this video we will look into one method for reinforcement learning called policy gradient method. So let's have a quick review of what the RL agent is and what it requires to formulate a problem. So in case of the reinforcement learning, we look into doing the dynamic programming with rewards and penalties, as we have said this. The importantly, we are exploring the policy based on the observations that we have. And then based on the observations, state of the environment, we look into giving the actions.

And of course, once the action is given, what is the reward and what is the whether the action was good or bad? Depending upon that, we come up with a reward function to to say that, OK, this action was good or bad. So we will have to consider what is the environment. We will have to give the answer what my RL agent is doing. And of course, since this is policy gradient based method, we will look into having some kind of an policy update method over here.

And this is definitely, as the name suggests, it is going to be a gradient based method. We will have to say what is the state of the environment and what are our actions based on which we'll say, OK, since these were the observations, you have taken this action, whether it's a good or bad action that is described by my reward function. All right. And then we looked into understanding the training phase, which had two terms, actor and critic. The actor responsibility was to take actions based on the observations and the state, mainly the state of the environment and keep refining the policy.

Whereas the critic will evaluate the actions. All right. So when we are doing the policy refinement, that time we'll say, OK, we had this particular trajectory which went from which took certain states and certain actions, state and action pairs. Now, when the state and action pairs were considered, we had followed certain policy. But so that is what my theta parameter is based on which we'll consider, OK, out of all these theta values, which policy is giving me the good answer or good set of rewards.

And that's what is going to be giving me the optimized policy answer. Whereas critic will look into saying that, OK, now I have the cumulative rewards starting from a particular state to the final state. What is the value that it creates while taking that particular trajectory or that particular transition of states, which had happened because of the actions. And it will evaluate and give a particular value to it based on which it will say, okay, this policy needs to be updated or this policy needs to be having some kind of, what kind of changes it will require in order to reach to the optimal policy levels. And so this critic network has a parameter W that we will consider.

All right. So the policy gradient method, what it has is, of course, we are looking into the environment. And this policy gradient method is to certain extent at a particular training phase, it will give a best action to be given to the environment. Now, how is it doing? It has a policy network which takes the state as an input and calculates the action probabilities.

Out of the multiple action probabilities that it has, it will select the best action and ask the environment to act on. Now, when I have to update the policy, what we will do is we will take the log of the probabilities just to make sure our probabilities are normalized to certain extent. And then we will consider the environment to give me the reward since we have defined the reward function and the environment is going to give me the new state. And there comes the reward function or the calculations of the value functions which the critic is doing. This particular reward and the log probabilities that we have is going to give me the idea where should I look forward for updating the policy.

So it is to certain extent giving me the gradient value, whether I should go in this particular direction to consider the policy update or that one in order to reach to an

optimal value. So in summary, The policy network will serve as the direct representation of a policy. So these are all networks, deep Q networks and whatnot, neural networks. These neural networks will output action probabilities, as we said, given a state.

And this particular algorithm, this reinforcement algorithm, will use this back propagation or the gradient ascent for the policy updates. All right. This is what is more or less high level way of understanding the policy gradient methods. There are many, many different variants of it, which talks about how to calculate the action probabilities and how to do this particular black propagation. And of course, how do I have different kind of critic networks and so on.

So the underlying challenge only in the policy gradient method is the high variance from returns. So when we do this particular back propagation, so this is based on individual updates that we are taking care of. Now these updates, this particular update, whether this action was good or bad, based on which the reward has been coming up, This particular value that is being fed back to the policy network can have a bias. Bias as in depending upon what was my previous value and this new value that it is receiving as an input should have some normalization.

All right. So they should have some common basis. And this particular return will have a very high variance if I'm not normalizing it and I'm considering a very large range of values.. So how do we mitigate this challenge? We can design a baseline network to get an estimate of the expected actual returns.

Finally, there's going to be some idea on how these particular input values to this particular feedback that we are calling it given to the policy network for the updates are going to vary. And this is going to be my values. And this network will basically make sure that it will not introduce any bias into the policy gradient. OK. Quick idea about how we implemented it, and we are giving you a set of code, which will talk about, okay, we have this actor network given by these many layers and mostly relu layers.

This is how the network has been created. Similarly, critic network is a very straight network. The parameters to be given here is 0.05. The way we have created and got the outputs which were shown in the previous video is sample time. We fixed sample time to

0.05, discount factor to something, critic learning rate, and the actor optimizer learning rate.

Here a small correction. this is minus four and this is minus four. These are very small values by which the gradients are being moved towards the optimal value and that sets the learning rate. The idea here is to give you a gist of what all parameters are required to set this particular RL based methodology to work and so on. So these are more or less like the parameters which play around, but the values that vary are of this particular order is what the idea behind giving these numbers. Interestingly, now, when we look into this policy gradient-based training,

We will continue this particular training phase and to certain extent, what we have to consider is what is the average reward that we are getting over the multiple episodes. So we will keep creating the training data and this particular training data will be run over multiple episodes. And this episode, these number of episodes, individual episode is making sure that my average reward is to certain extent is getting saturated. Now, even if I go beyond more number of episodes, this training data, this average reward is not changing that much. And that's where we will say, OK, this is how my network is trained now.

And let's look into applying it into the actual system. We have applied into the same simulation environment and showed the results last time. Our testing results showed that the vertical stabilization of the inverted pendulum from different initial states, even with noise, is able to create this. Because when we generated the training data, we had already considered some sources of noise and some different initial conditions for which this network was trained. We will quickly look into what this particular code structure is.

You can consider that reading this particular readme.txt. At the same time, there are three files, pendulum environment file, this test file, and the training environment file, all right? So this readme file gives you the idea of what each particular file is about, and then how do we use this particular, use these files. First of all, we'll have to consider training the environment. So we'll run this train underscore environment underscore pg.m, and then we'll do the testing part by running this test underscore pg file.

All right, wish you all the best and please do try running these files, these particular MATLAB files and change a little bit the way we did, made changes into removing the torque term into the reward function. And we could see that the maximum torque is increasing by value from plus minus 10 with the range of the torques have changed from plus minus 10 units to plus minus 50 units. All right. Thank you so much.