INTELLIGENT CONTROL OF ROBOTIC SYSTEMS

Prof. M. Felix Orlando

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Lecture 16: Introduction to Reinforcement Learning

Good morning, everyone. Today, we are going to see an introduction to reinforcement learning and its application in robotics. First, we will see the definition of reinforcement learning. It is a type of machine learning approach where an agent learns to make decisions by performing actions and receiving rewards or penalties based on the proposed action. Unlike supervised learning,

which requires a teacher, reinforcement learning does not rely on labeled data but instead learns from the consequences of its actions. Now, consider the case of a chess game, where the problem statement is to learn to play chess without explicit guidance on every move by the agent or the player. The challenges associated with this case of chess play are the cost of a teacher to guide every move and the fact that the goodness of a move depends on future moves, not just the immediate action. So the objective here is to achieve a sequence of moves that leads to victory rather than

optimizing individual moves. The objective is to obtain final victory through a sequence of moves. Here, the feedback mechanism only provides feedback—winning or losing—at the end of the game. There is also another example, which is a maze-solving robotic system. The robotic system has to navigate a maze to reach the exit.

Navigating the maze may take a lot of time to reach the exit or may take less time to reach the exit. How smartly the robot can navigate the maze to reach the exit. And that smartness or the intellectual property of the robotic system is through decision making. Now the challenges are no immediate feedback during the journey and the goal here is to find the shortest path introducing a time preference.

Here the objective is to minimize the steps to reach the exit competing against the time. And here the feedback mechanism is rewarded by Only upon reaching the exit. Now coming to the core concepts of reinforcement learning. They are agent, environment, actions and rewards.

Agent is nothing but the learner or the decision maker. Whereas in the case of chess game. The chess player is the agent. Whereas in the case of a robot navigating a maze. In order to reach the exit.

Is an agent. Now coming to the environment. The world. The agent interacts with. Is called the environment.

Now coming back to the examples. Chessboard is the environment. In the case of chess play. And maze is the environment. In the case of.

Robot navigating a maze. To find the exit. And the rewards. Or the actions are. The possible moves.

the robotic system can take towards the directional steps to reach the exit. And the rewards could be appreciating reward or penalties, depending on the feedback from the environment. The feedback from the environment is the reward, which could be winning or losing rewards. Okay, in the case is the maze exit report, you can say in the case of a robotic system to reach the exit through the maze. And also, we can see about policy, which is nothing but the strategy that the agent uses to decide actions based on the current state.

Policy is nothing but the strategy that the agent or the robotic system uses to decide actions, whether to move forward, backward, left side, right side. thereby avoiding obstacles or taking optimal path to reach the target. The value function is nothing but it is a measure of the long term benefit of states under certain strategy or policy. Now coming to the important aspect in reinforcement learning which is nothing but the K-armed bandit problem. So here

The definition of this, first of all let us see what is K-armed bandit problem. K-armed bandit is a hypothetical slot machine with K levers each offering different rewards. Sometimes their rewards could be deterministic, sometimes they are stochastic. So the objective here is to decide which lever to pull to maximize the total reward. Because the K-armed bandit is nothing but a hypothetical slot machine with K levers, different levers.

So the objective again is to decide which lever to pull to maximize the total reward. Now, comparing this approach with supervised learning. Supervised learning is nothing but, in a classification problem, the teacher provides the correct class or classes. The optimal lever for maximizing the earning process—there is a teacher in supervised learning so that the error can be found. The desired target and the actual outcome or output of the

approach discrepancy is the error between the desired and the actual values. So, in supervised learning, we do have a teacher who is there to correct or provide the optimal lever for maximizing the earnings.

In reinforcement learning, instead of being told which lever is the best, the agent must try different levers and learn from the outcomes of pulling each lever. What is the outcome? Poor or good? Based on that, the agent himself learns through experience, you can say. Now, coming to the characteristics of the K-armed bandit problem.

It is a single-state one. And it is simplified because there is only one state. Because it is a one-slot machine. So, no need to consider transitions between the states. And there is an immediate reward.

The reward is received immediately after pulling a lever. So the effect of the action is instantly known. Immediately known. Now let us talk about the difference between exploitation and exploration. Exploration.

Here we have exploration and exploitation. Exploring. Trying different levers to discover which one is the highest reward. That is exploration. Exploitation is continuing to pull the lever that has historically given the highest reward.

Only that is getting used again and again. That is why it is getting exploited. Now introduction to action value Q of A. As the name indicates we get a value for performing an action. That quantitative measure is Q of A. As for the definition Q of A is the value of action A representing the expected reward from taking action A. Initially $Q(a) = 0$ for all the actions. Now there are two types of rewards.

One is deterministic, and another one is stochastic. Let us talk about deterministic rewards. If rewards are deterministic, R A is constant. For each pull of a lever.

So set Q of A, the value of action. A, that is $Q(a) = r_a$ after observing the reward. Once $Q(a) > 0$

for an action, continue exploiting it. To consistently receive the same reward R A. Which is greater than 0. Now there is a need for exploration. Because even if Q of A is greater than 0, There might be another action with a higher reward.

That means the reward value RA. Which may be greater than the RA that we prescribed earlier. So, we need to explore another action to get a better reward compared to RA. Choosing different actions to potentially find a better reward and update Q of A accordingly. These are the two needs for exploration.

Now, coming to the action selection strategy. There is a rule called the exploitation rule: choose A* if Q of A* equals the maximum Q of A for all A. This ensures selection of the highest Q(A) to maximize the reward. Now, let us talk about stochastic rewards. When rewards are stochastic, R of A value varies for each pull of the same lever.

In this case, the reward is defined by a probability distribution. That is, the probability of R given an action A. Now, let us talk about estimating action value over time. So, there is an online update rule for Q of A. The value of action Q of A for an action A is updated online by an approach similar to the delta rule used in learning algorithms, given by

$$Q_{t+1}(a) = Q_t(a) + \alpha(r_{t+1} - Q_t(a))$$

where Q_t(A) is the current estimate of the action value, and r_{t+1} is the reward received after action A at time t+1.

And alpha here is a learning rate that gradually decrease to over time for convergence. And the value is in the interval 0 to 1 for the learning rate alpha. Now as time t increases Q of A converges to the mean of probability of R given A. Now let us talk about the generalization to full reinforcement learning. Earlier it was single state.

Now let us generalize to multiple states so that it extends to scenarios with multiple states each having its own reward probabilities given by P of R given S_i, A_j where A_j is the action In state S j. Or S i. Let us talk that. Let us talk about the. State action value. Q of S i comma.

A j. That represents the value of taking action. A_j in state S_i. There. Due to multiple states. There exist. State transitions.

Which are nothing but the actions. Influence not only rewards but also the transition of the agent from one state to another. And there is delayed reward that introduces the challenge of estimating immediate values from rewards that are delayed. Now again we come back to the important reinforcement elements agent rewards. which is nothing but the learner or the decision maker in a reinforcement learning setup.

And this agent is responsible for making decisions based on the information it gathers from the environment. The environment is nothing but the external world with which the agent interacts. The environment includes everything outside the agent that can affect the outcome of its actions, leading to the rewards. State and action is another important element.

The agent perceives its current situation known as the state ST and takes an action AT to change its state. The set of all possible states is denoted by capital S. And the actions is available. And the actions that are available in a state are denoted by A of S t. Now let us discuss about the interaction with the environment. Time and state transition.

Now, time is divided into discrete steps. Example: t equals 0, 1, 2, and continues. At each time step, the agent is in a state S_t and takes action A_t. This causes a transition to a new state, denoted by S_{t+1}. Now, the reward is given after the agent takes an action. The environment provides feedback in the form of a reward, denoted as R_{t+1}. The reward helps the agent to

evaluate the quality of its action. Now, the Markov decision process. So now, the significant part in Aurel. Now, the Markov property is that the system follows the Markov property. Meaning that the future state S_{t+1} depends on the current state,

and the current action, Aurel. Represented as S_T and A_T, but not on the past states or actions. Now, the probability distributions. The reward and the next state are drawn from probability distributions. That is, the probability of R_{T+1}.

Given S_T and A_T for the reward and distribution. P of S_T+1 given S_T is the probability of state T+1 given S_T and A_T for the state transition. Now, let us talk about the deterministic case where, in some scenarios, taking a particular action in a specific state always leads to the same next state and the reverse. This is termed as the deterministic environment. Now, let us discuss the important terminologies involved in RL.

Episode and terminal states. When we say terminal state, there is something called the initial state. Which is nothing but the state where the agent begins. Terminal state is nothing but a special state where the episode or the task ends. Once in the terminal state, no further actions lead to a new state, and accordingly, no rewards are given.

An episode is nothing but the sequence of actions taken by the agent from the initial state to the terminal state. Each episode represents one trial or one complete attempt at the

task. Now, let us talk about policy and value function. The policy, represented by π, is a strategy or rule that the agent follows to decide what action to take in any given state. Formally, it is a mapping from a state to actions.

Thus, Policy pi is defined as a mapping from state S to A actions. A value of policy is represented by V pi of S T. It is nothing but the expected total reward the agent can collect starting from state S T and following policy pi. It is a measure of how good it is for the agent to be in a particular state given that it follows the policy now let us talk about two cases finite horizon model and infinite horizon model the first one here is finite horizon model here the objective is in task with a finite number of steps say horizon t the goal is to maximize the total reward over the next t steps thus the cumulative reward is given by the value function for this case is $V_\pi(S_t) = \mathrm{E}[r_{t+1} + r_{t+2} + \cdots + r_{t+T}] = \mathrm{E}\left[\sum_{i=1}^{T} r_{t+i}\right]$

So this equation sums up the rewards over the next t steps starting from the state S t. Now let us talk about the infinite horizon model. Here the discounted rewards in task without a predefined endpoint that is infinite horizon. Future rewards are discounted by a factor gamma which is nothing but which is termed as discounted rate to keep the total reward finite. And hence a value function in this infinite horizon model is given by

$$V_\pi(S_t) = \mathrm{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots] = \mathrm{E}\left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}\right]$$

So this equation sums up the rewards over the next T steps starting from state S T. So here there is something called the discount rate which is represented as gamma which value lies in the interval 0 to 1. It determines how much future rewards are valued compared to immediate rewards. Precisely a smaller gamma makes the agent short-sighted And the 1, which is the larger 1, which is closer to 1, number 1, makes it more foresighted.

Now, let us talk about the optimal policy. So, the optimal policy represented by pi star is nothing but the best possible strategy that maximizes the value function for all states. It is a policy that, if followed, yields the highest cumulative reward. So, the objective is to obtain the maximum value for the policy, which is represented by $V^*(S_t) = \max_\pi V_\pi(S_t)$

Now, let us talk about the state-action value function, that is, the Q function. So, state-action state S and action A. The Q function, represented for the state-action, is denoted as

$$V^*(S_t) = \max_{a_t} Q^*(S_t, a_t)$$

which, instead of just considering the value of state, evaluates the value of taking a specific action A t in state S t and then following the optimal policy afterward. state, the Q function evaluates the value of taking a specific action A t in the state S t and then following the optimal policy afterward. Now, let us see the relation to the value function, which is given by $V^*(S_t) = \max_{a_t} Q^*(S_t, a_t)$

So, this equation shows that the value of a state is the maximum value obtained from any action in that state. So, there is Bellman's recursive equation, which is defined by

$$V^*(S_t) = \max_{a_t}(\mathrm{E}[r_{t+1} + \gamma V^*(S_{t+1})])$$

This is Bellman's recursive equation, which expresses the value of a state as the best possible immediate reward plus the discounted value of the next state. So, accordingly, the Q state or the Q function formulation is given by $Q^*(S_t, a_t) = \mathrm{E}\left[r_{t+1} + \gamma \sum_{S_{t+1}} p(S_{t+1}|S_t, a_t) \max_{a_{t+1}} Q^*(S_{t+1}, a_{t+1})\right]$

So, this equation represents or expresses the value of taking action A_t in state S_t and then following the optimal policy. The policy derivation is towards a greedy policy approach, which means that it always picks the action that seems to be the best possible. At each step, which is given by the expression $\pi^*(S_t) = \arg\max_{a_t} Q^*(S_t, a_t)$

Thus, once the Q-function is known, the optimal policy can be derived by always choosing the action that maximizes the Q-value. Now, the value iteration algorithm, which is nothing but the last portion of this lecture. So, that is given by initialization as the first step: start by assigning arbitrary values to V(s) for all the states.

Then the iteration begins: repeatedly update the value of each state using the following steps for each state s, for each action a under the state s. Find Q(s, a), which is given by

$$Q(S, a) = \mathrm{E}[r|S, a] + \gamma \sum_{S'} p(S'|S, a)V(S')$$

Then update the value of S using the expression $V(S) = \max_{a} Q(S, a)$

The third step is the convergence. Continue updating until the values converge—that is, the change in the values is negligible between the current iteration and the previous iteration. Finally, the outcome is the algorithm converges to the optimal value function, from which the optimal policy can be derived.

Now, let us talk about the connection to robotics in path planning. Imagine a mobile robot navigating an unknown environment, such as a warehouse, where it needs to reach a specific target location while avoiding obstacles. The robot here must learn the optimal path over time through a trial-and-error approach rather than being explicitly programmed with all possible paths. Because all possible paths may not be so easy for a specific or generalized case. That is why we go for the RL approach, so that it can learn through its trial-and-error approach over time.

Or from its experience it can learn because through each action it gets a reward or penalty based on good move or worst move. So the environment here is the robot is in a grid like environment where each cell represents a state S. The robot can move in four directions up, down, right and left each representing an action A. So the agent here is a robotic system which decides the next action to take based on its current state. The rewards or the robot receives a positive reward when it reaches the target location. A small negative reward is given to the robot for each move to encourage shorter paths.

And a large negative reward is given if the robot collides with an obstacle. Now, let us correlate with the K-armed bandit problem. At any given state, the robot has four possible actions. Move up, down, right or left. This is similar to the K-armed bandit problem where the robot must choose the action that maximizes the reward.

Here, the robot uses an exploration-exploitation strategy to balance between trying new actions and using the known action. best action. So, here the Q learning approach is that the robot estimates the value of each action Q of S comma A that is the action the value of the action at every state based on the rewards received. So, initially Q of S comma A is 0 for all actions as a robot moves and receives reward it is getting updated by the expression and $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

So, the Markov decision process is continued here.

The robot's navigation is modeled as an MDP, which is a macro decision process where the next state and reward depend only on the current state and action. Over time, the robot learns the optimal policy π*(s) to choose the action that maximizes the cumulative reward, enabling the robotic system to navigate efficiently in the warehouse without colliding with obstacles and reaching the shortest path. Now, the Bellman equation for the robotic system here is: the value of the optimal policy is updated in such a way that

$$V^*(s) = \max_a \left[r + \gamma \sum_{s'} P(s'|s,a) V^*(s') \right]$$

The robot iteratively improves its policy until it converges to the optimal path. Hence, the result is that through reinforcement learning, the robot eventually learns the shortest and safest path without colliding with obstacles to reach the target, and it has also spent the minimum time reaching this optimal path. By this,

we have seen the introduction to reinforcement learning and Markov decision processes, as well as the learning update rule for this process, applied in robotics for path planning by avoiding obstacles to reach the target. Thank you so much for your time.