Q-Learning with Linear Function Approximator

Exercise 6.1: Q-Learning Agent

In this exercise, you will implement a Q-Learning Agent with Linear Function Approximation.

Make sure that you have completed the setup requirements as described in the Set Up Lab Environments section.

Now, run jupyter notebook and open the "Ex6.1 Q-Learning Agent with Linear Function Approximation.ipynb" notebook under **Module** 6 folder.

1. Examine the notebook. We have given you boiler plate and helper code for the implementation of the Q-Learning agent with Linear Function Approximation.
2. Let's take a look at the QLearningFAAgent implementation.
3. Go to **##TODO1**, which is on the **init**() function. Here you need to initialize the agent.
   In the previous lab, you use a dictionary to store the state-action values. Using linear function approximation, we will instead store thetas, which are the weight parameters for the function approximators.
   How many thetas do you need? In this scenario you will need 64 thetas, more to that in a second, but right now initialize the value of the 64 thetas to zeros. Here you will also need to initialize the value of epsilon, alpha, and gamma. This part should be similar to what you have done with the previous labs.
4. Let's take a look at the featureExtractor() function.
   This function is used to create features from the combination of state and action. Recall the implementation of SimpleRoomsEnv environment. Each state is represented using arrays of one-hot encoding, that means each state is a vector of length 16. In this environment we have 4 different actions. Using dimensionality scaling, we can represent the features of state and actions with vector of length 4 x 16. That is why we initialized 64 thetas.

5. Go to **##TODO2**, which is on the **act()** function. Here you need to implement the epsilon-greedy policy for the agent to act according to the policy. Some boiler plate code has been given. What you need to do is to correctly compute the q values, which is Q(s, a) for a in all actions.
6. Go to **##TODO3**, which is on the **learn()** function. Here you need write the code to implement q-learning update as explained in this module. The pseudo code is given to you.
   What you need to do is to is to correctly compute maxqnew, which is the max Q(s',a) for a in all actions, compute oldv, which is the Q(s,a) where a is the action taken, and update thetas. Recall thetas update equation from the lecture.
7. Once you have implemented all the three #TODOs, you can test your implementation against the SimpleRoomsEnv.

QLearning with Function Approximation Agent in the SimpleRoomsEnv Environment (10 episodes)

Let's set up an experiment with your QLearningFAAgent and the SimpleRoomsEnv environment.

1. Use the default values for alpha, epsilon, and gamma for your QLearningFAAgent.
2. Once you've set up your experiment, run the the experiment for **10** episodes with the interactive set to **True**.
3. Run this experiment several times.

Lab Question

1/1 point (graded)

Based on your observation of the above experiments, on average, does the agent manage to reach the goal within 10 episodes?

- ○ No
- ◉ Yes ✔

Submit

You have used 2 of 2 attempts

Your agent should have managed to achieve the goal within 10 episodes, although it might not achieve the goal consistently.

When we say consistently, it means that the agent achieves the goal in consecutive episodes. In some environments, the agent might fail again even though it has achieve the goal consecutively in multiple episodes. This is due to the exploration factor of the agent and the design of the environment.

Now that we have verified that the agent "works" in small number of episodes, let's set up another experiment, this time with more number of episodes. Turn off the animation so that you don't need to wait for the animation to render.

QLearning with Function Approximation Agent in the SimpleRoomsEnv Environment (50 episodes)

1. Let's set up another experiment with the same parameters, use the default values for alpha, epsilon, and gamma for your QLearningFAAgent.
2. But now, set the interactive to **False**, and run the experiment for **50** episodes.
3. Run this experiment several times.

Lab Question

1/1 point (graded)

Based on your observation of the above experiments, on average, around how many episodes does the agent need to achieve the goal consistently?

- ○ After the first episode the agent already managed to reach the goal albeit with huge number of steps
- ○ All it takes is two episodes
- ◉ Ten to Fifteen episodes ✔
- ○ Around 25 episodes

- Only after 50 episodes

Submit You have used 1 of 2 attempts

Lab Question

1/1 point (graded)

Once the agent consistently achieve the goal, does it ever fail again?

- (•) No ✔
- Yes

Submit You have used 1 of 2 attempts