1. What is Q-learning:

Q-learning is an off policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It’s considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy isn’t needed. More specifically, q-learning seeks to learn a policy that maximizes the total reward. Which Q stands for quality.

1. What you learned in the ICP:

In this ICP I learned how to design Deep Q Learning Network for cartpole game in python using Keras & OpenAI Gym and visualize the result. The goal of the game is to ensure the cart is moving left and right while the pole is balancing.

Please note that in this ICP I used Jupyter notebook because of the “env.render()” did not work on Google Colab.

1. Screen shots that shows the successful execution of each required step of your code:

Importing the required libraires visualize the CartPole-v1 environment. For the policy I added 1 to push cart to the right. (change it to 0 means to push it to the left)

Graphical user interface, text, application, email

Description automatically generated

Look at the docstring to get more information about the environment.

Graphical user interface, text

Description automatically generated

I start creating a policy function to manage the tip speed/velocity of the cart. After that I divided the state space into number of buckets. In addition, I used KBinsDiscretizer to look at the angle and pole speed and initialize the Q value table.

Graphical user interface, text, application, email

Description automatically generated

New\_Q\_Value method: this step I update the function and calculate as the reward of action taken plus of the max of expected reward for the new state.

Learning rate method: to evaluate the old q value with the new one.

Exploration rate method: to explore the state space by random proportion actions.

Graphical user interface, text, application

Description automatically generated

Time to train 10000 simulations and define the q value policy to determine the action and the exploration rare to insert random action. Last, I have update the q value as a weighted of old q value and learned value.

Graphical user interface, text, application, email

Description automatically generated

Result:

Chart, box and whisker chart

Description automatically generated

1. Conclusion:

I have Successfully executed the code and design Deep Q Learning Network for cartpole game. As shown in the screenshot above, I tried to minimize the number of episodes as much as I can but that did not help me to keep the pole balanced.

1. Video link:

<https://drive.google.com/file/d/1ih5s8_v8etLx08yKWUSIurafkFGXVB9M/view?usp=sharing>