Homework 3:
Value function-based DRL

In this problem set you will study Deep RL for algorithms based on value function. Some exercises are inspired by the book of Sutton and Barto.

Instructions. You are to submit a PDF for your answers and a Python notebook for your code. For the PDF, you can type your answers for example using latex or use handwritten notes as long as they are **clearly and easily readable**. Do not forget to **fully justify** your answers. You notebook's name should follow the format: `RL-Fall23-HW3_LastName_FirstName_xxx` where `xxx` is to be replaced by the exercise and question numbers.

Exercise 1 (Programming)**.** This exercise is about comparing tabular RL and deep RL on a finite state space and finite action space MDP. Consider the cliff-walking problem (Sutton & Barto Example 6.6; see also sample code on Brightspace). You will use and modify the code provided in Brightspace in the file called `RL_CSCI_SHU_375_q_learning_fixed-gym.ipynb`.

Part 1: modification of the existing tabular code:

1. Add a function to evaluate the Q-table performance (using the greedy policy; not ϵ-greedy). For this, use the SARSA algorithm (for policy evaluation, without policy improvement step) and run it for `n_epi_eval` episodes (this parameter must be chosen, balancing the computational time and the noisiness of the evaluation result).

2. Modify the existing code for tabular Q-learning: Every `eval_every` episodes, call the above function to evaluate the Q-table and store the result. At the end of the Q-learning algorithm, plot the curve of the evaluated reward versus the number of iterations.

3. Run tabular Q-learning and SARSA and plot both types of curves:

   (1) the code provided already plots the "reward using behavior policy" vs iterations, and

   (2) with the modification, you can plot the "reward using target (greedy) policy" vs iterations (you will have fewer points in this curve if you choose `eval_every` greater than 1; try to take `eval_every` as small as possible while having a reasonable computational time).

Part 2: addition of deep RL code:

1. Implement DQN and run it on the same environment. Plot both types of curves:

   (1) reward using behavior policy vs iterations, and

   (2) "reward using target (greedy) policy" vs iterations.

2. Implement double-DQN and run it on the same environment. Plots both types of curves as above.

3. **(Bonus)** Implement dueling-DQN and run it on the same environment. Plots both types of curves as above.

4. Comment on the performance of the 5 algorithms (tabular Q-learning, tabular SARSA, DQN, double-DQN, dueling-DQN). Some aspects you might want to consider are: the number of steps, the number of samples, the number of gradient steps, the memory requirement etc.

Exercise 2 (Programming)**.** This exercise is about using deep RL on several continuous state space (and finite action space) MDPs. You will use and modify the code provided in Brightspace in the file called `RL_CSCI_SHU_375_dqn_lunar.ipynb`.

1. Create a copy of the code. What is the highest (training) reward that you manage to get by tuning the hyperparameters?

2. Create another copy. Modify it to run DQN on Gym's car racing example: `https://www.gymlibrary.dev/environments/box2d/car_racing/` Tune the hyperparameters to get the highest (training) reward you can.

3. **(Bonus)** Create another copy. Modify it to run DQN on Gym's bipedal walker example: `https://www.gymlibrary.dev/environments/box2d/bipedal_walker/` Tune the hyperparameters to get the highest (training) reward you can.

4. **(Bonus)** Add an evaluation function using MC samples and evaluate your DQN in each example.