Reinforcement Learning
Exercise 3

1 Q-Learning

In this the exercise, you'll be applying grid-based Q-learning to the *Cartpole* and *LunarLander* environments.

1.1 Cartpole

Recall the Cartpole environment from Exercise 1.

Task 1.1 – 25 points Implement Q-learning for the Cartpole environment in the file `qlearning.py`. Compare using a constant value of $\epsilon = 0.2$ to reduce the value of ϵ over time (use the *greedy in limit with infinite exploration* formula from the lecture). Use the following hyperparameter values: $\alpha = 0.1$ and $\gamma = 0.98$. For GLIE, aim at reaching $\epsilon = 0.1$ after 20000 episodes and round the value of constant b to the nearest integer.

Hint: The states in Cartpole are continuous, while grid-based methods can only be directly applied to discrete state spaces. You can overcome this issue by discretizing Cartpole states to a finite grid (use the state space limits and grid dimensions from `qlearning.py`).

Task 1.2 – 10 points Use your Q-function values to calculate the optimal value function of each state. Plot the heatmap of the value function in terms of x and θ. For plotting, average the values over $\dot{x}$ and $\dot{\theta}$.

Hint: For plotting the heatmap you can use Matplotlib-pyplot: `pyplot.imshow(values_array)` or Seaborn: `seaborn.heatmap(values_array)`

Question 1 – 15 points What do you think the heatmap would have looked like:

(a) before the training?

(b) after a single episode?

(c) halfway through the training?

Justify why for all the cases. Attaching the plots is not required.

Task 1.3 – 5 points Set ϵ to zero, effectively making the policy greedy w.r.t. current Q-value estimates. Run the code again,

(a) keeping the initial estimates of the Q function at 0,

(b) setting the initial estimates of the Q function to 50 for all states and actions.

Observe the results.

Question 2 Based on the results you observed in Task 1.3, answer the following questions:

Question 2.1 – 5 points In which case does the model perform better?

Question 2.2 – 15 points Why is this the case, and how does the initialization of Q values affect exploration?

1.2 Lunar lander

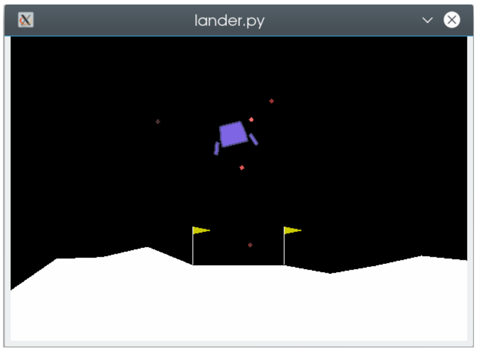

Figure 1: The Lunar lander environment.

The *Lunar lander* environment is shown in Figure 1. The goal is to make the lunar lander land on the ground between two flag poles. The agent receives a positive reward for moving towards the landing pad, landing, etc. A negative reward is given for firing the main engine (more fuel-efficient policies are better) and for crashing. Four actions are available: firing the left/right/main engines, or doing nothing (free fall). The observation vector consists of 6 continuous and 2 discrete values:

$$o = \begin{pmatrix} x & y & \dot{x} & \dot{y} & \theta & \dot{\theta} & c_l & c_r \end{pmatrix}^T, \tag{1}$$

where x and y are the coordinates of the lander, $\dot{x}$ and $\dot{y}$ its velocities, θ represents the rotation angle and $\dot{\theta}$ the angular velocity of the lander. Two discrete values c_l and c_r indicate whether the lander's legs are in contact with the ground (0 or 1).

Hint: For the Lunar Lander, change the environment to `gym.make(LunarLander-v2)`.

Task 2 – 10 points Modify your code for Task 1.1 and try to apply it in the Lunar lander environment. Run it for 20000 episodes (which was enough for the Cartpole to learn).

Hint: The last two elements of the state vector, c_l and c_r are binary (0 or 1), so, when discretizing the state space, it is enough to use a grid dimension of two for them. This will save a substantial amount of memory.

Hint: If you're getting errors about Box2D not being installed, you have to install the `box2d-py` package with `pip3`. You may also have to additionally install Swig from your distro repositories (`apt-get install`/`pacman -S`/`emerge`, etc.).

Question 3 – 15 points Is the lander able to learn any useful behaviour? Why/why not?

2 Submission

The deadline to submit the solutions through MyCourses is on Monday, 18.10 at 12:00. Example solutions will be presented during exercise sessions on that day (18.10).

The report *must* include:

1. **Answers to all questions** posed in the text.
2. The **training performance plots** for each of the tasks (Task 1.1 - fixed and GLIE, Task 1.3 - for both initializations, Task 2 - Lunar Lander).
3. The **heatmap** from the end of the training (Task 1.2).

In addition to the report, you must submit as separate files:

1. *NumPy* file `q_values.npy`, which **includes the learned Q-values for Task 1.1 for Cartpole with GLIE**, saved when the training has finished (don't attach the values for contant epsilon),
2. *NumPy* file `value_func.npy`, which **contains the value function for the same conditions as in the previous point**,
3. Python code used to solve the exercises.

Do not attach the Q values for Lunar Lander.

Please remember that not submitting a PDF report following the **Latex template** provided by us will lead to subtraction of points.

For more formatting guidelines and general tips please refer to the submission instructions file on mycourses.

If you need help or clarification solving the exercises, you are welcome to come to the exercise sessions in weeks 39/40/41. Good luck!