Exercise 3

Jari Mattila - 35260T
ELEC-E8125 - Reinforcement Learning

October 10, 2021

Task 1.1

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).

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)

NumPy file value_func.npy, which contains the value function for the same conditions as in the previous point

Source files: qlearning.py, q_values.npy, value_func.npy

Task 1.2

The heatmap from the end of the training (Task 1.2).

Plot the heatmap of the value function in terms of x and . For plotting, average the values over x and θ.

Question 1

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.

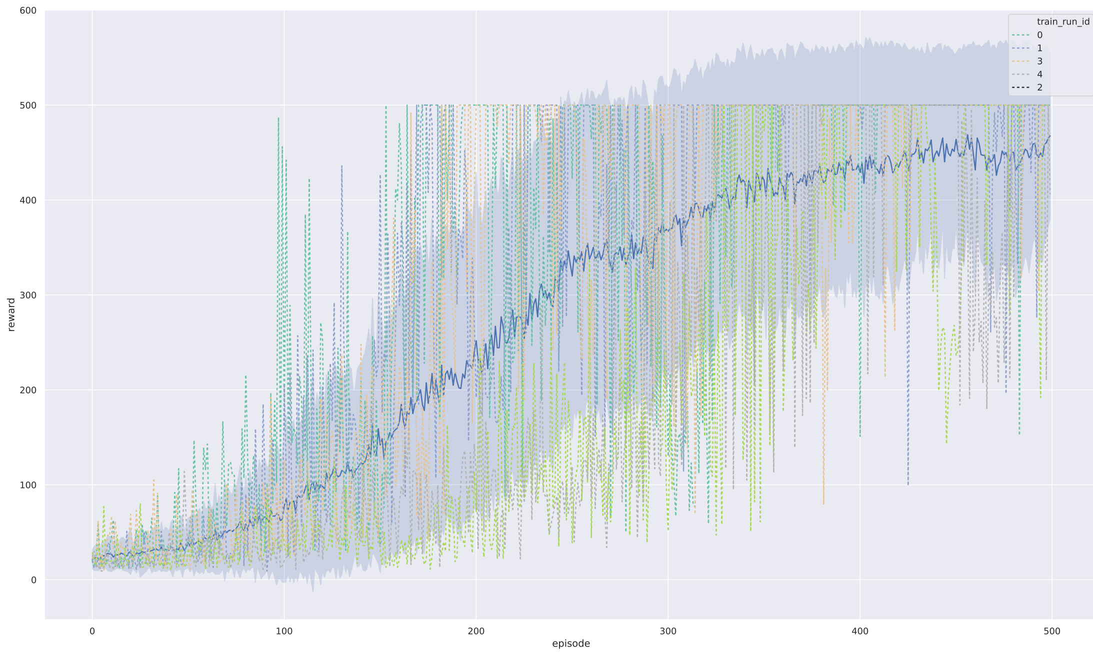

Figure 1: This is a sample figure.

Task 1.3

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).

Source files: qlearning.py

Question 2

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

Question 2.1

In which case does the model perform better?

Question 2.2

Why is this the case, and how does the initialization of Q values affect exploration?

Task 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).

Source files: qlearning.py

Question 3

Is the lander able to learn any useful behaviour? Why/why not?

1 Task 1090

If you add a figure, you can refer to it using Figure. 2.

To cite works, put them in the template.bib file and use [?].

```
for episode_number in range(train_episodes):
    reward_sum, timesteps = 0, 0
    done = False
    # Reset the environment and observe the initial state
    observation = env.reset()

    # Loop until the episode is over
    while not done:
        # Get action from the agent
        action, action_probabilities = agent.get_action(
           observation)
        previous_observation = observation

        # Perform the action on the environment, get new state and
            reward
        observation, reward, done, info = env.step(action)

        # Store action's outcome (so that the agent can improve
           its policy)
        agent.store_outcome(previous_observation,
           action_probabilities, action, reward)

        # Draw the frame, if desired
        if render:
            env.render()
```

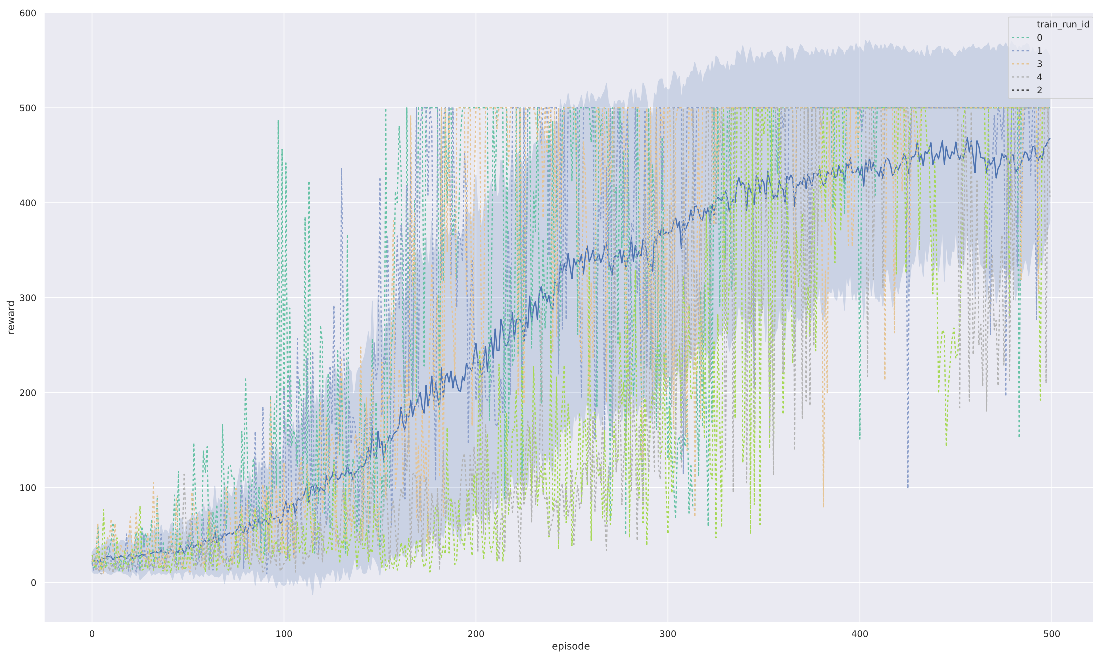

Figure 2: This is a sample figure.