Exercise 3

Christian Montecchiani - 100655834
ELEC-E8125 - Reinforcement Learning

October 3, 2022

1 Task 1

In the first Task of this third assignment, we were required to implement Q-learning as it is presented in [1] in Section 6.5. We needed to implement and compare two exploration methods:

1. Fig. 1 shows the mean episode reward using a **constant** value of $\varepsilon = 0.1$.
2. Fig. 1 shows the mean episode reward using **GLIE**, which reduces the value of ε over time. In the exercise, our goal was to obtain $\varepsilon = 0.1$ at episode $k = 20000$. We can achieve the goal by using the constant value b is given by the following equation:

$$\varepsilon = \frac{b}{b+k}$$

   By doing the computation and rounding to the closer integer we obtain a value of $b = 2222$

By comparing the plot in Fig. 1, using GLIE leads to better rewards both in the short and long periods. The Cartpole is able to maintain the pole stable for 100 timesteps before the 5000-th episode. Moreover, the highest reward that it is able to achieve is 200. This result is never achieved if we train the same model with constant ε.

The code for the first method is shown below:

```
def get_action(state, q_axis, q_table, epsilon=0.0):
    ########## Your code starts here ##########
    q_value = q_table[get_table_idx(state, q_axis)]
    actions_len = len(q_value)

    action = np.argmax(q_value)
    u = np.random.random()
    if u <= epsilon:
        action = np.random.randint(actions_len)

    return action
```

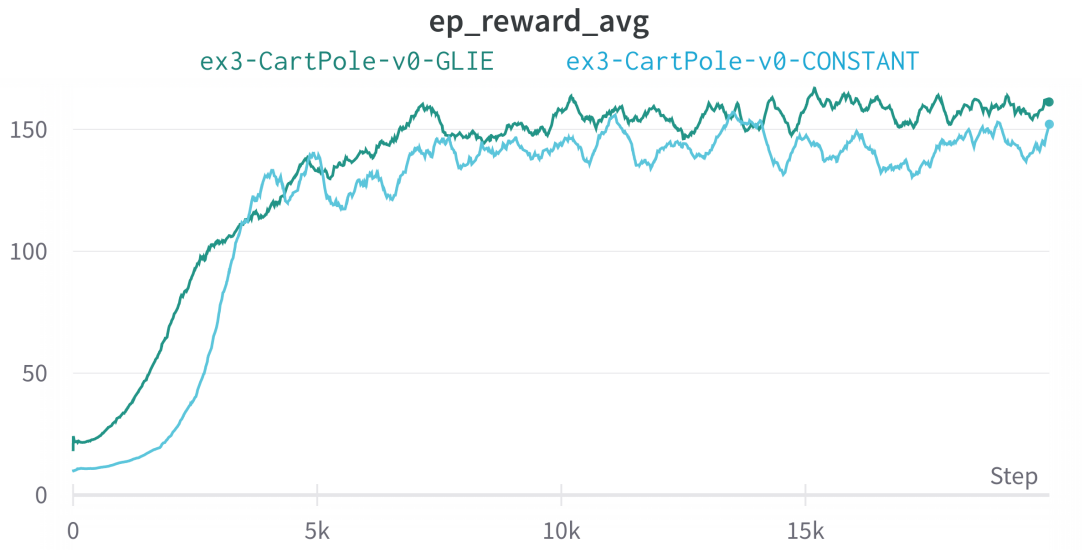

Figure 1: Episode average reward with using GLIE and with constant $\varepsilon = 0.1$.

```
def update_q_value(old_state, action, new_state, gamma, reward,
    done, alpha, q_axis, q_table):

    old_table_idx = get_table_idx(old_state, q_axis) # idx of q
       (s_old, *)
    new_table_idx = get_table_idx(new_state, q_axis) # idx of q
       (s_new, *)
    ########## Your code starts here ##########

    old_q_value = q_table[old_table_idx][action]
    max_q_value = np.max(q_table[new_table_idx])

    if not done:
        update = old_q_value + alpha*(reward + gamma*
           max_q_value - old_q_value)
    else:
        update = old_q_value + alpha*(reward - old_q_value)

    q_table[old_table_idx][action] = update
    ########### Your code ends here ##########
    return q_table
```

1.1 Task 1.3

In this task, we were required to plot the heatmap of the value function in terms of x and θ. Below is shown the code used:

```
fig, axs = plt.subplots(1, 4, figsize = [20, 5])
x_axis, th_axis = 1, 3

# Get values table
values_0 = np.max(q_table_0, axis =-1)
values_1 = np.max(q_table_1, axis =-1)
values_10000 = np.max(q_table_10000, axis =-1)
values_20000 = np.max(q_table_20000, axis =-1)

x_dot_theta_dot_0 = np.average(values_0, axis= (x_axis, th_axis))
x_dot_theta_dot_1 = np.average(values_1, axis= (x_axis, th_axis))
x_dot_theta_dot_10000 = np.average(values_10000, axis= (x_axis, th_axis))
x_dot_theta_dot_20000 = np.average(values_20000, axis= (x_axis, th_axis))


# Plot the heatmap
sns.heatmap(x_dot_theta_dot_0, ax=axs[0])
sns.heatmap(x_dot_theta_dot_1, ax=axs[1])
sns.heatmap(x_dot_theta_dot_10000, ax=axs[2])
sns.heatmap(x_dot_theta_dot_20000, ax=axs[3])

# Set titles and axes
training_episodes = [0, 1, 10000, 20000]
for i, train_eps in enumerate(training_episodes):
    axs[i].set_xlabel ('Theta', fontsize = 8)
    axs[i].set_ylabel ('x', fontsize = 8)
    axs[i].set_title(f'Heatmap with {train_eps} training
       episodes', fontstyle = 'italic')


fig.savefig(fname="heatmaps_glie.png")
```

1.2 Question 1.1

Figures 2 and 3 show the heatmaps[1] obtained with the following number of training episodes.

1. After 0 training episode. The value function is 0 over all the states.
2. After 1 training episode. Just a few states have values different from 0.
3. After 10000 training episodes, it is similar to the last one, which corresponds to the

[1]The figures show the result obtained with `seed = 408`

complete training with 20000 episodes. This means that the most valuable states are identified and they have the highest value function.

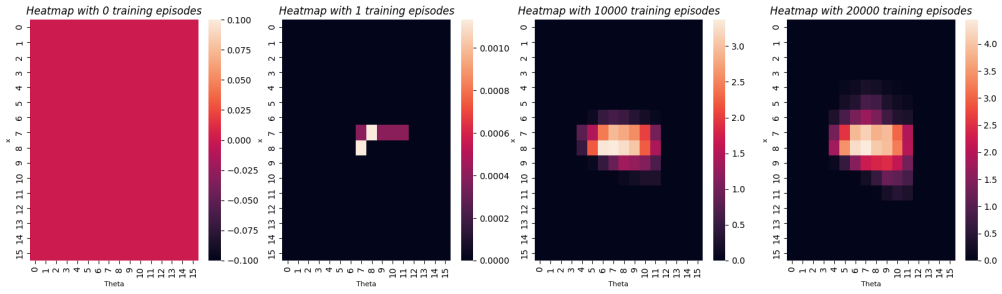

Figure 2: Heatmap value function with constant $\varepsilon = 0.1$.

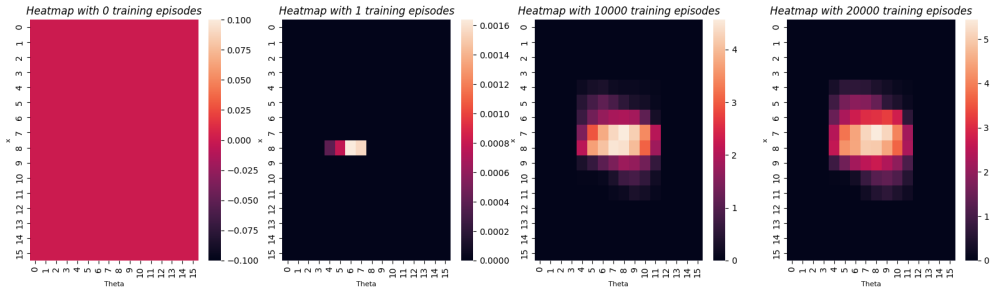

Figure 3: Heatmap value function with GLIE.

Fig. 4 shows the heatmap of the value function in terms of x and θ of the *Cartpole* trained for 20000 episodes and `seed = 0`. Below is shown the code used:

```
fig, axs = plt.subplots(figsize = [7, 5])
x_axis, th_axis = 1, 3


#values = np.max(q_table_0, axis =-1)
values = np.max(q_table, axis =-1)

x_dot_theta_dot = np.average(values, axis= (x_axis, th_axis))


# Plot the heatmap
sns.heatmap(x_dot_theta_dot, ax=axs)

# Set titles and axes
axs.set_xlabel ('Theta', fontsize = 8)
axs.set_ylabel ('X', fontsize = 8)
axs.set_title(f'Heatmap with 20000 training episodes',
   fontstyle = 'italic')


fig.savefig(fname="heatmaps_glie_20000.png")
```

1.3 Task 1.3

By effectively making the policy *greedy*, that means setting $\varepsilon = 0$, we were asked to test different initial values:

- Initial estimates of the Q function at 0. Fig. 5 shows the mean episode reward.
- Initial estimates of the Q function at 50. Fig. 6 shows the mean episode reward.

1.4 Question 2.1

As it is possible to see from the figures above, the model that performs better is the one with initial estimates of the Q function 50.

1.5 Question 2.2

As it is explained in Section 2.6 of the [1], even if the agent does a fair amount of exploration even if greedy actions are selected all the time. That is caused because the initial value set to 50 is higher than the new update of the Q- function update.

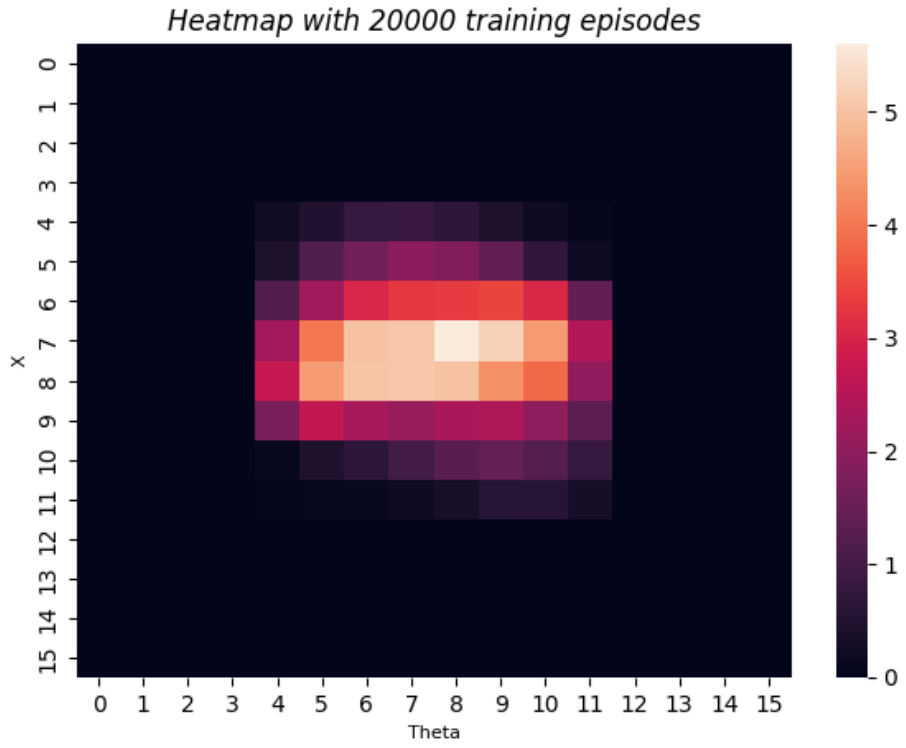

Figure 4: Heatmap value function with GLIE (trained for 20000 episodes and `seed = 0`.

2 Task 2

In the second Task, we were asked to train for 20000 episodes the Lunar Lander agent, using both strategy:

- Fig. 7 shows the result obtained using $\varepsilon = 0.1$.
- Fig. 8 shows the result obtained using GLIE, with $b = 2222$.

2.1 Question 3

The lander does not learn to land between the flag poles. By exploring the test videos the *Lunar Lander* is able to understand the position of the flags and go closer to them. However, 20000 training episodes are not enough to find the optimal policy, that is because the state space vector has more dimension than the *Cartpole* and consequentially also the Q-Table is bigger.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

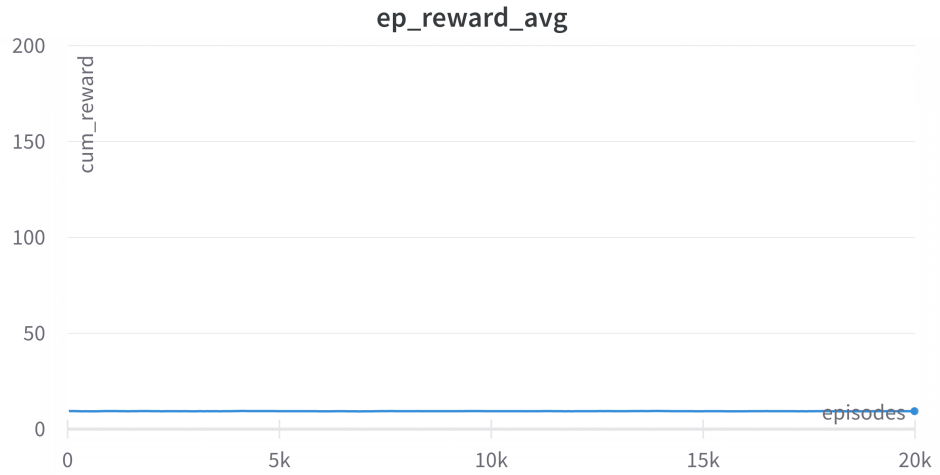

Figure 5: Mean episode reward with constant $\varepsilon = 0$ with initial estimate 0.

Figure 6: Mean episode reward with constant $\varepsilon = 0$ with initial estimate 50.

Figure 7: Mean episode reward with constant ε=0.1.

Figure 8: Mean episode reward with using GLIE.