Exercise 3

Tommaso Brumani - 100481325
ELEC-E8125 - Reinforcement Learning

October 3, 2022

Task 1.1

To implement the Q-learning algorithm, the **train.py** file was modified by adding the following code:

- in the **update_q_value** function:

```
########## Your code starts here ##########

if done:
    maxval = 0.0
else:
    maxval = np.max(q_table[new_table_idx])

q_table[old_table_idx][action] = q_table[old_table_idx][
   action] + alpha*(reward + gamma*maxval - q_table[
   old_table_idx][action])

############ Your code ends here ##########
```

- in the **get_action** function:

```
########## Your code starts here ##########

# determine wether to choose greedy policy
choose_rand = np.random.random() <= epsilon

if choose_rand:
    # take random action
    return np.random.randint(0, q_table.shape[4])
else:
    # take greedy action
    q_s = q_table[get_table_idx(state, q_axis)]
    return np.argmax(q_s)

########## Your code ends here #########
```

The model was then trained and tested using two exploration methods:

(a) For $\boldsymbol{\epsilon = 0.1}$ the file **q_table_T1_1_a.pkl** was produced, and the average reward by episode achieved during training is presented in the plot in Figure 1.

Figure 1: average reward by episode for $\epsilon = 0.1$.

(b) For **GLIE $\boldsymbol{\epsilon}$** the file **q_table_T1_1_b.pkl** was produced. In order to achieve $\epsilon = 0.1$ after 20000 episodes, the correct value for parameter b was determined to be $\mathbf{b = 2222}$. The average reward by episode achieved during training is presented in the plot in Figure 2.

Figure 2: average reward by episode for GLIE ϵ.

Task 1.2

To plot the **heatmap** of the calculated optimal value function in each state, the **plot_value.ipynb** file was modified by adding the following code:

```
########## Your code begins here. ##########

# Calculate the value function
values = np.max(q_table, axis=4)

# Plot the heatmap
values_array = values.mean(axis=(1,3))

ax = sns.heatmap(values_array.T, xticklabels=th_axis,
    yticklabels=x_axis)
plt.title('Heatmap')
ax.set_xlabel('theta')
ax.set_ylabel('x')
plt.show()

########## Your code ends here. ##########
```

The heatmap itself, bearing the $\mathbf{x}$ state variable on the vertical axis and the $\boldsymbol{\theta}$ variable on the horizontal axis, is reported in Figure 3.

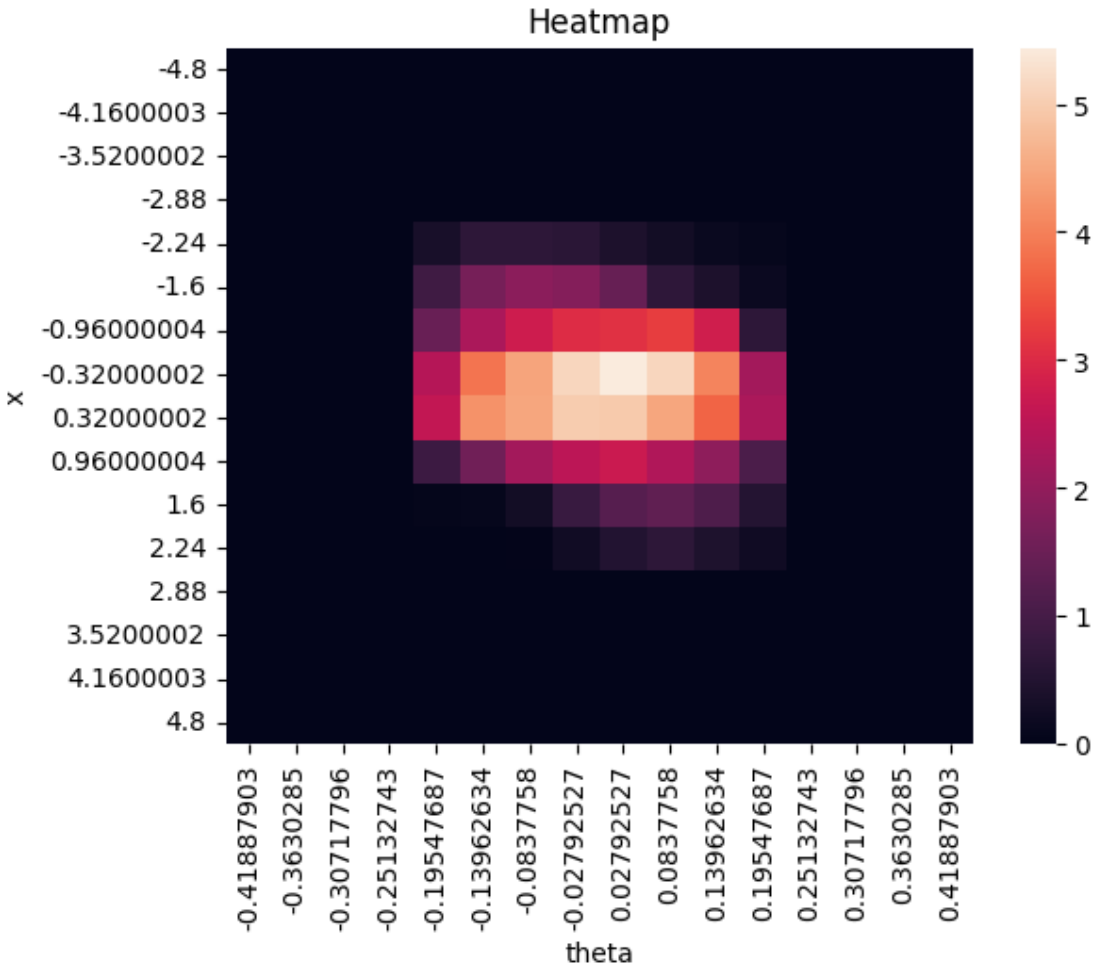

Figure 3: heatmap of the calculated value function.

Question 1

(a) **Before training**, we can expect the heatmap to be completely dark (or at least of a single color), as the value function for each state is initialized at zero.

(b) **After a single episode of training**, we can expect the heatmap to start showing some varied colors.
Since the model would not yet have had the time to explore many state-action pairs, however, the plot would likely be very different from that of the final heatmap.

(c) **Halfway through the training**, we can expect the heatmap to have an appearance very close to the one reached at the end of the training.
This is because the smoothed average episodic reward plot in Figure 2 appears to indicate that the reward at 10k episodes has already reached a value close to that obtained at 20k episodes, suggesting that the value function is similar to the one achieved in the end.

Task 1.3

When setting $\boldsymbol{\epsilon}$ to **zero** and running the training using different initializations of the Q function, the following results were obtained:

(a) By keeping the initial estimates of the Q function at **0**, the average return by episode obtained is the one reported in Figure 4.

Figure 4: average return by episode for $\epsilon = 0$ and Q initialized at 0.

(b) By setting the initial estimates of the Q function to **50** for all states and actions, the average return by episode obtained is the one reported in Figure 5.

Question 2

2.1) The model appears to perform better when the initial estimates of Q are set to 50: in the first case (Q initialized at 0) the average episode reward oscillates around a value of 9.35, whereas in the second case it appears to converge to 150 similarly to the epsilon-greedy training of the model.

2.2) This likely happens because when initializing all estimates of Q to zero, the model will choose a random action when visiting a state for the first time (as all state-action pairs have a value of zero), and on subsequent visit will always choose the action that was chosen the first time, as its associated value is the only one that has been increased above zero.
When initializing Q to 50, instead, it is likely that the summation

$$[R_{t+1} + \gamma \max Q(S_{t+1}, a) - Q(S_t, A_t)]$$

Figure 5: average return by episode for $\epsilon = 0$ and Q initialized at 50.

has a negative result for most sub-optimal actions, which will decrease their value function leading the greedy policy to explore other options in subsequent visits to that particular state, ultimately converging to associating the best actions with the highest values.

Task 2

When running the **Lunar Lander** environment for 20000 episodes with the same GLIE ϵ employed in Task 1.1 (b), the results presented in Figure 6 were obtained.

Question 3

In this case, the model fails to learn to land between the poles with a satisfactory frequency and only succeeds in doing so during testing about half of the time.
Assuming the reward function is well constructed, this is likely because the agent in this case has to move through an environment characterized by double the number of state variables when compared to the Cartpole instance, as well as having double the number of available actions.
This means that, assuming the number of different values admissible for each state variable to be roughly the same, the Q matrix for the Lunar Lander environment is much larger than that of the Cartpole, and thus needs a much greater amount of episodes of exploration in order to converge to a well-performing policy.

Figure 6: average return by episode for Lunar Lander with GLIE ϵ.