Exercise 3

Nguyen Xuan Binh - 887799
ELEC-E8125 - Reinforcement Learning

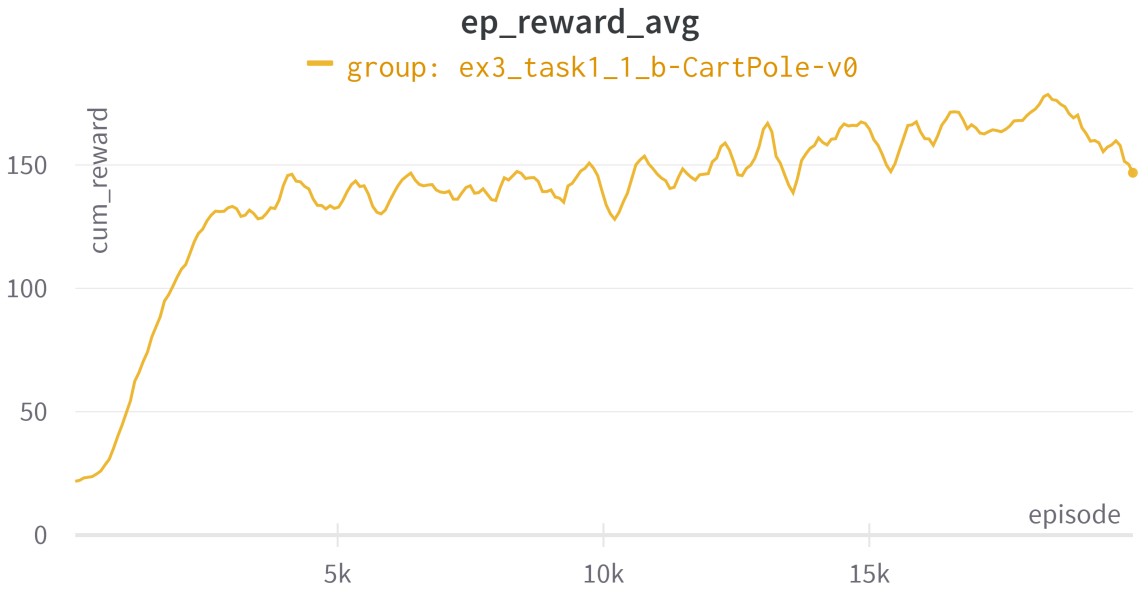
July 26, 2023

Task 1.1 - 25 points

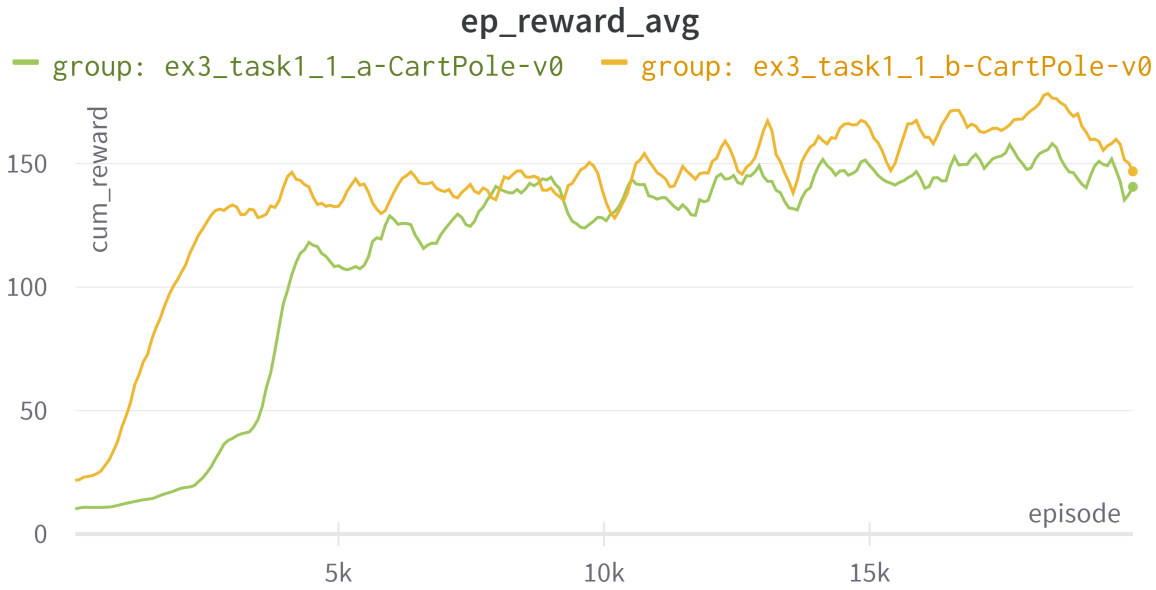
Implement Q-learning as presented in [1] Section 6.5 for the Cartpole environment in file train.py. We need to compare two exploration methods:
(a) using a constant value of $\epsilon = 0.1$;
The Python code for Task 1 is

```
def get_action(state, q_axis, q_table, epsilon=0.0):
    # if epsilon == 0.0, the policy will be greedy -- always
       choose the best action
    # TODO: Implement epsilon-greedy
    idx = get_table_idx(state, q_axis)
    if np.random.random() < epsilon:
        return np.random.choice(q_table[idx].shape[0])
    else:
        return np.argmax(q_table[idx])

def update_q_value(old_state, action, new_state, gamma, reward,
    done, alpha, q_axis, q_table):
    # TODO: Task 1.1, update q value
    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, *)

    old_coordinates_with_action = (*old_table_idx, action)
    q_table[old_coordinates_with_action] = q_table[
       old_coordinates_with_action] + alpha * (reward + (1 -
       done) * gamma * np.max(q_table[new_table_idx]) - q_table
       [ old_coordinates_with_action])

    return q_table
```

(b) using GLIE (i.e. greedy in limit with infinite exploration) which reduces the value of ϵ over time. Its formula can be found from the lecture.



This is the combined figure, which can show that the GLIE method performs better than the constant ϵ value method.

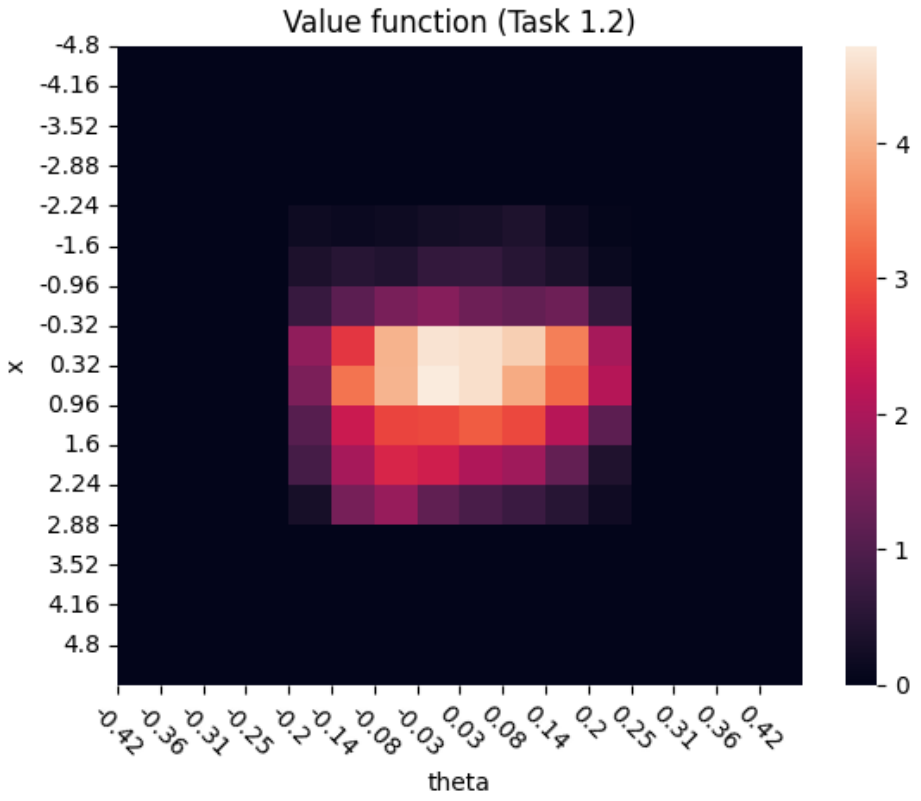


Task 1.2 - 10 points

Use your Q-function values estimated with GLIE to calculate the optimal value function of each state. Complete the plot_value.ipynb to plot the heatmap of the value function in terms of x and θ, such that θ is on horizontal and x is on vertical axis. For plotting, average the values over $\dot{x}$ and $\dot{\theta}$. Attach the heatmap in your report.

This is the code used to plot the heatmap below

```
# Calculate the value function
values = np.max(q_table, axis=-1).mean(axis=1).mean(axis=2)
# Plot the heatmap
sns.heatmap(values)
plt.xlabel("theta")
plt.ylabel("x")
plt.xticks(range(discr), th_axis.round(2), rotation=-45)
plt.yticks(range(discr), x_axis.round(-45), rotation=0)
plt.title("Value function (Task 1.2)")
plt.show()
```



Question 1 - 15 points

What do you think the heatmap would have looked like (justify why for all the cases. Attaching the plots is not required):

(a) before the training?

The heatmap will look monotone because before the training, the Q-table is just initialized with a constant value (0 or 50)

(b) after a single episode?

After a single episode, the Cartpole is at the start of exploration stage, so most of the x values are dark (value 0) except at the middle position, and many values of theta in the middle (-0.08 to 0.08) will be white (high values) as the pole is constantly trying to balance itself. All other combinations are still unexplored so they have dark colors.

(c) halfway through the training?

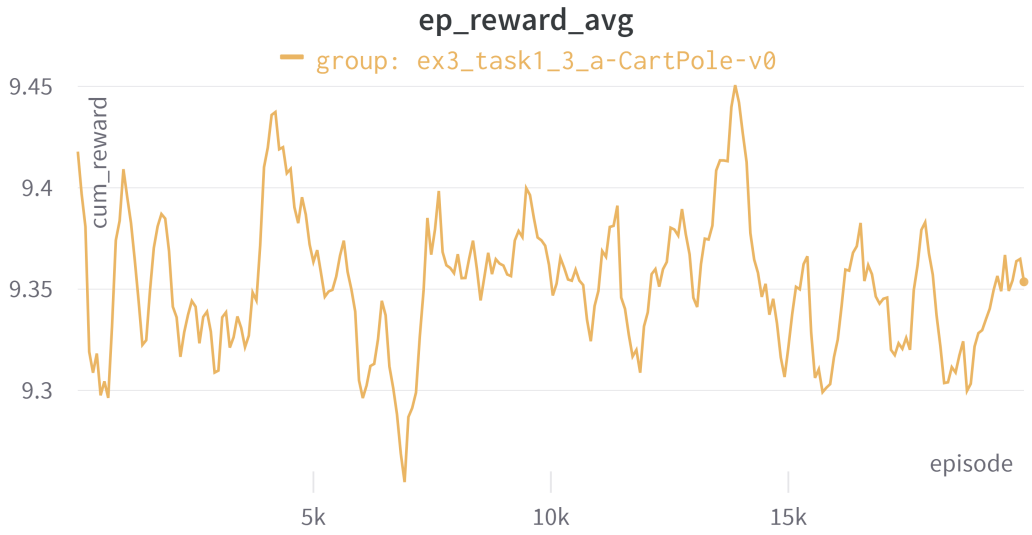
It would look quite similar to the heatmap at the end of the 20000 episodes as the cart pole has already learned quite well how to balance itself

Task 1.3 - 10 points

Set ϵ to zero, effectively making the policy greedy w.r.t. current Q-value estimates. Run the training again while

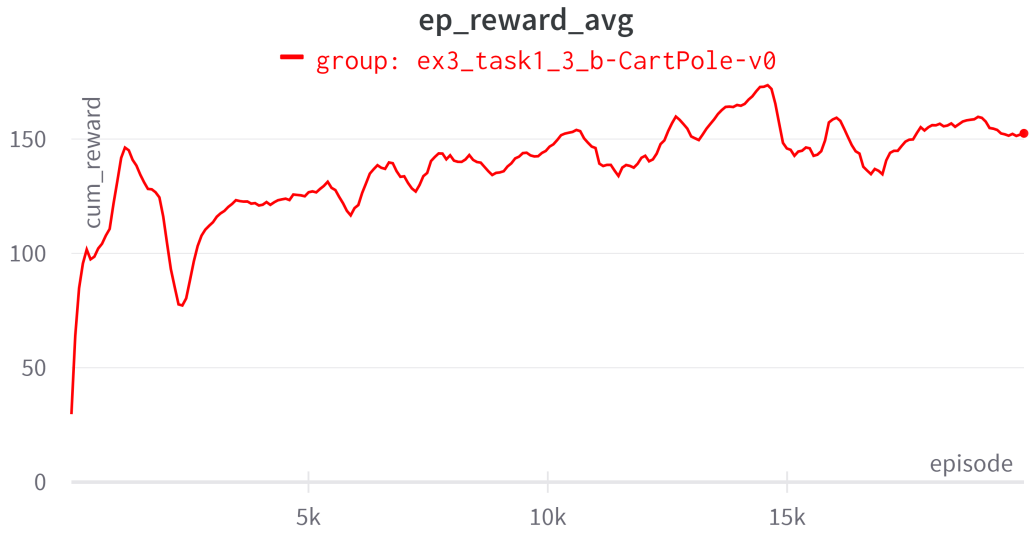
(a) keeping the initial estimates of the Q function at 0

The plot when initial estimate of Q function is 0



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

The plot when initial estimate of Q function is 50



This is the combined plot of both progresses

Question 2.1 - 5 points

In which case does the model perform better?

The case with better performance is when the initial estimate of the Q function is 50.

Question 2.2 - 15 points

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

To understand this phenomenon, it is required to look into the Q-value update equation

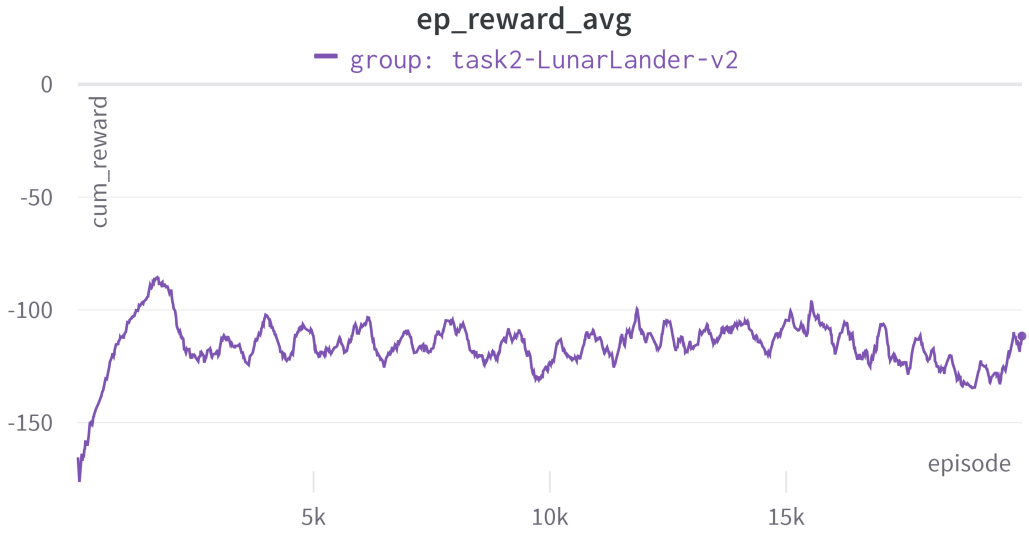
$$Q(S,A) = Q(S,A) + \alpha[R + \gamma \max_a Q(S^{'},a) - Q(S,A)]$$

The given reward is 1 if the Cart Pole remains balanced. From the equation, the term $\gamma \max_a Q(S^{'},a) - Q(S,A)$ will make the Q-values of many states decrease very fast when Q-values are initialized at 50, favoring early exploitation for the states with relatively unchanged Q-values. If they are initialized with 0, the Q-values of undesirable states will decrease much slower and the CartPole can still be stuck in exploration mode to find promising states.

Task 2 - 5 points

Run the training for Lunar Lander environment by using python train.py env=lunarlander_v2 epsilon=glie glie_b=<value-in-task-1.1>. Run it for 20000 episodes (which was enough for the Cartpole to learn). Attach the training performance plot in your report.

The training performance plot of the lunar lander is below



Question 3 - 15 points

Does the lander learn to land between the flag poles? Why/why not?

No, the lander does not learn to land between the flag poles. Despite training for over 20000 episodes, the lander cannot figure out a good policy because the state space is enormous with 8 different states. Furthermore, the step size of each state is also small, resulting in million of different states. Therefore, Q learning is not enough to handle such a large number of states to obtain any meaningful policy.