Exercise 1

Tommaso Brumani - 100481325
ELEC-E8125 - Reinforcement Learning

September 18, 2022

Task 1

The agent's 'episodes - episode rewards' training plot exported from Wandb is reported in Figure 1.

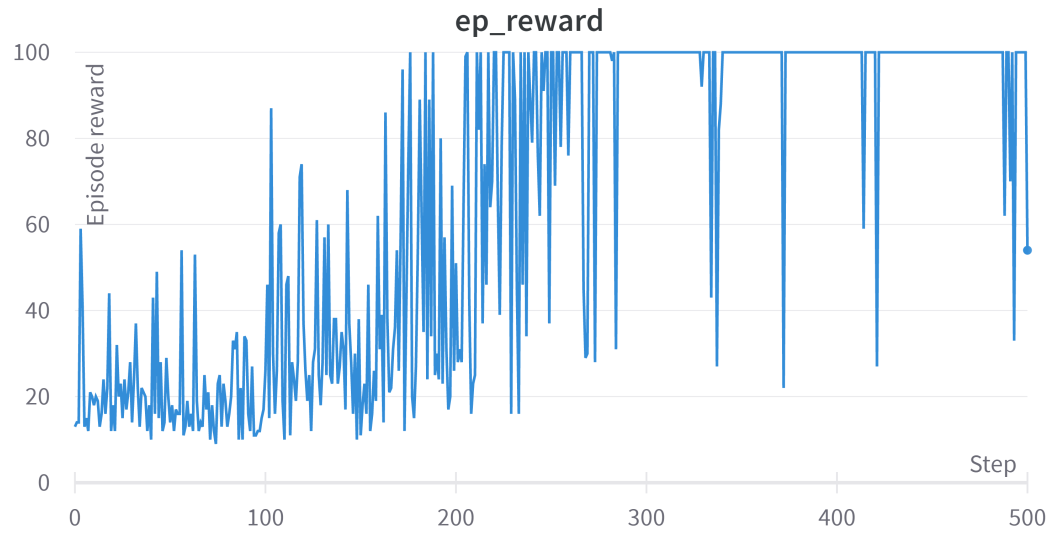

Figure 1: 'episodes - episode rewards' training plot.

The average test reward for the model was 927.8.

Question 1.1

After testing the model multiple times with different random seeds for 1000 timesteps, it appears that its performance worsens considerably when trying to balance the pole for a longer amount of time than the one it was trained for (100 timesteps).
One possible explanation for this is that the model has learned a policy that, while optimized for a shorter timeframe, may have poor results when dealing with a larger number of timesteps.

Task 2

Repeating the training and testing of the model with five different seeds, the following average test values were obtained:

1. (seed = 100) avg. reward = 433.6
2. (seed = 200) avg. reward = 608.0
3. (seed = 300) avg. reward = 422.8
4. (seed = 400) avg. reward = 439.7
5. (seed = 500) avg. reward = 625.2

Question 2.1

The performance of the model appears to show a significant amount of variation according to the chosen seed: this is likely because the agent is initialized differently based on the seed, leading to different results after 100 timesteps of training.

Question 2.2

The stochasticity implies that the same algorithm may have different performances given different initialization.
This means that it is important to optimize the way the initialization is carried out, and possibly train multiple times the same algorithms with different seeds in order to have a better idea of their generalized performance.

Task 3

1. The reward function for the first *Reacher* environment, used to achieve constant clockwise rotation, is:

```python
def get_reward(self, prev_state, action, next_state):

    # give reward if the angle of the first link in next
       state is smaller than in current state
    if next_state[0] < prev_state[0]:
        return 1
    return 0
```

2. The reward function for the second *Reacher* environment, used to reach the marked position in $\mathbf{x} = [1.0, 1.0]$ is:

```python
def get_reward(self, prev_state, action, next_state):

    # fetch cartesian coordinates for next state
    pos = self.get_cartesian_pos(next_state)

    # reward is negative of euclidean distance from target
      x = [1.0, 1.0]
    rew = -sqrt((pos[0]-1.0)**2 + (pos[1]-1.0)**2)

    return rew
```

The models for the two *Reacher* environments are contained respectively in the files 'Reacher-v1_params_T3_1.pt' and 'Reacher-v1_params_T3_2.pt'.

Task 4

To plot the required information about the second behavior of the *Reacher* environment, the 'plot_rew.ipynb' file was modified as follows:

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

    actions[i, j] = np.argmax(action_probs)

    rewards[i,j] =  env.get_reward(None, None, state)

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

The 'state - reward' and 'state - best action' plots are reported in Figure 2 and Figure 3, respectively.

Question 4.1

The highest rewards appear to be achieved when the joint for the first and second links approach angles of approximately π and -1.5 respectively, or when both approach 1.5.

Question 4.2

Determining whether or not the model has learned an optimal policy to reach the goal would require knowing the optimal value function for the problem.
Nevertheless, looking into the learner's *reset()* function, it appears that the joint angles are initialized in a uniformly random manner between the extremes of 0.1 and -0.1 radians. Since the full range of values for these angles should be $[\pi, -\pi]$, it follows that during training the state could only be initialized in a limited number of distinct configurations, and thus the best action to adopt in certain different states could have gone under-explored and is thus likely not optimized.

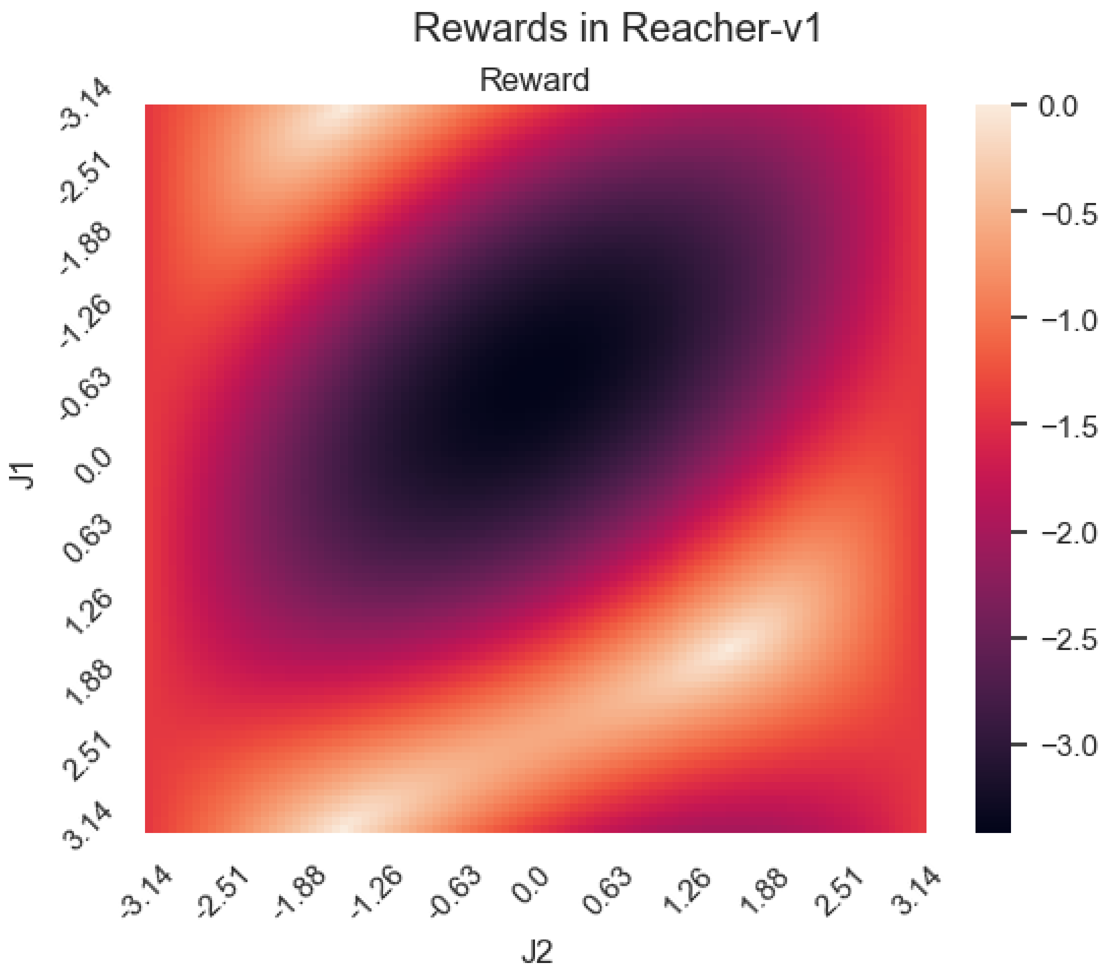

Figure 2: 'state - reward' plot.

Feedback

1. The time spent solving the exercise was approximately 2-3 hours. Setting up the environment also required around 1 hour, mostly because of problems with using Windows.
2. The only task that presented some difficulties was Task 3.2, as it was sometimes hard to understand the reason why a certain reward function was not working as expected (although that is probably the point of the exercise).

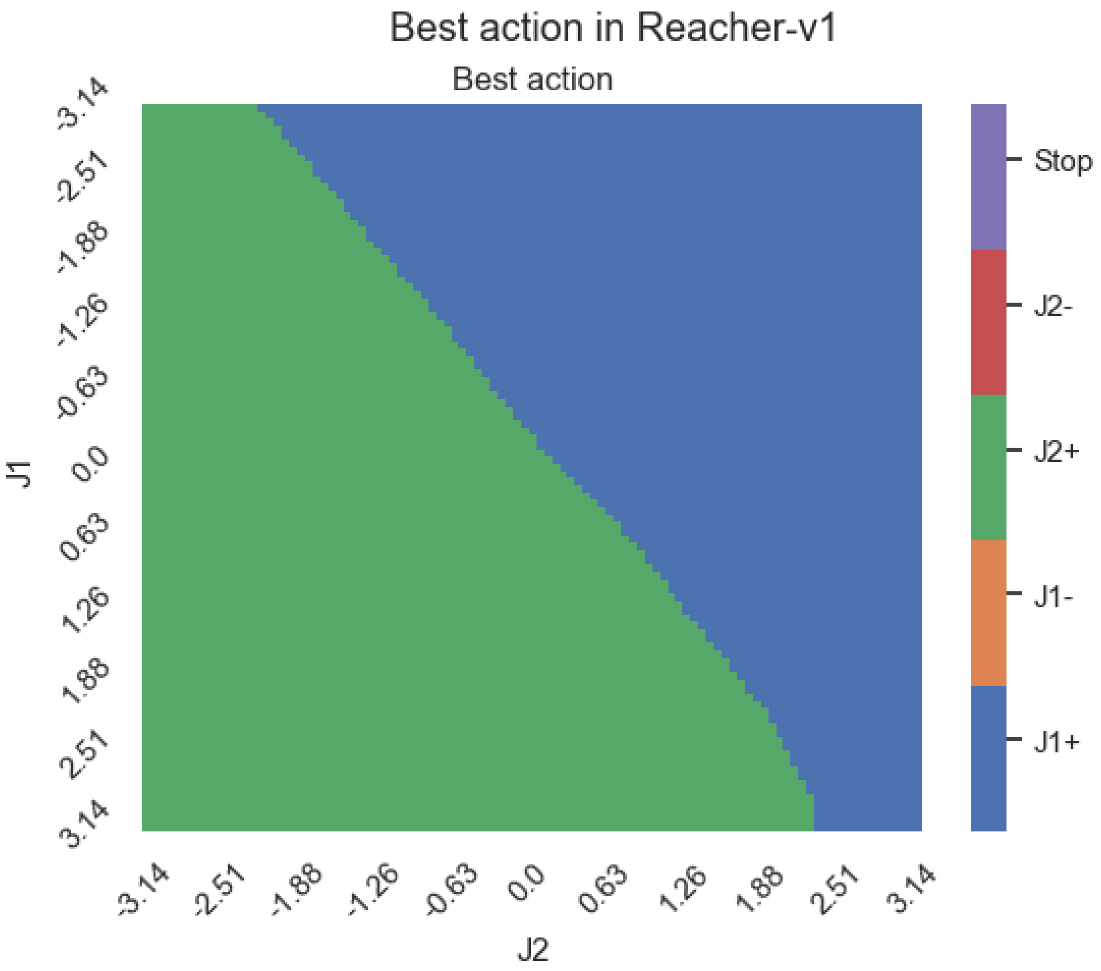

Figure 3: 'state - best action' plot.