Exercise Session 2

Reinforcement Learning

Sebastian Griesbach / Prof. Carlo D'Eramo

Summer semester 2024 / June 25th

In this exercise, you will implement a Deep Q Network and learn to use PyTorch. You will need to install Gymnasium with box2d as a dependency, e.g. by using the command ‘pip install gymnasium[box2d]‘ (not sure if this is true anymore but I remember using Python versions newer than 3.10 sometimes causes dependency issues) **Please only upload Your agent.py and dqn.py files as as a .zip file. Write your answers to exercises 2 and 5 as comments at the bottom of the agent.py file.**

Exercise 1

The exercise consists of the following files:

- **agent.py** This file contains the DQN agent which is learning, it follows the same agent pattern you already know. But its update function does not take a single transition but a batch of transitions instead. You need to modify this file.
- **dqn.py** This file contains the architecture of the actual Deep Q Network which we will use as our function approximation. You need to modify this file.
- **replay_buffer.py** This file contains a simple implementation of a replay buffer. A Memory that saves transitions and lets you sample batches from them. You don't need to modify this file but do take a look, there is not much going on there.
- **trainer.py** This class runs the actual training process and the interaction with the environment. You don't need to modify this file but you should understand it.
- **run_exercise.py** This is your program entry point. Here all the classes are instantiated, the hyperparameters are set and the training process gets called. You might need to modify this file.

Please take a look at all components and make sure you understand how they work together to create the algorithm. This is all the code that is needed to create a deep reinforcement learning algorithm. (If you don't count the libraries we use).

Exercise 2

Briefly explain the difference between the **train** and the **evaluate** function in **trainer.py**.

Exercise 3

Open **dqn.py** and implement the neural network as described in the comments. Refer to the PyTorch documentation to find out how to do so.

Exercise 4

Open **agent.py** and complete the **update** function of **DQNAgent** following the comments. Again refer to the PyTorch documentation on how to perform the specific steps. Also, take a look at the other parts of the code, especially the **_update_target_net** function and understand what it does. If you think you are done execute **run_exercise.py**. If everything works correctly you should see clear learning progress after the training. Although 50 training episodes are probably not enough to converge to a good policy that lands most of the time. Often it just overs above the ground at that point. When everything looks good increase the number of training episodes to 200 and see if the lander now lands correctly most of the time.

Exercise 5

To which of the Gymnasium classical control tasks can you apply this algorithm as it is now? (This doesn't mean it needs to solve it right away) What are the criteria of an environment such that DQN is applicable?