Report for Project 1: Navigation

Udacity Deep Reinforcement Learning Nanodegree

April 8, 2020

1 The Implementation

The main part of the code is written in the Jupyter Notebook `Navigation.ipynb`, which guides the user when executing the code. The intelligent agent has been implemented in the `agent.py` file, and it uses a neural network defined in `model.py`.

1.1 The Neural Network

The network as defined in `model.py` by default has two hidden layers with 64 neurons each, without dropout layers (similar to the original Deep Q-Network architecture).

1.2 The Agent Class

The agent class is capable of learning according to the Deep Q-Learning algorithm as discussed in class.

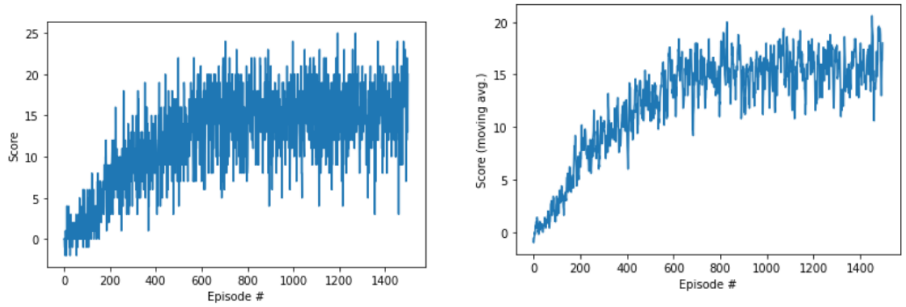
Within its `__init__` method, several internal variables are defined: state size, action size, the network weights (local w and target weights w^-), the optimizer to be used (Adam) and the `ReplayBuffer` for experience replay.

The `step` method triggers a learning step if the time step for learning is reached (e.g., only update the network every 4 training steps). To do this, it triggers the `learn` method, which backpropagates the errors made from sampled experiences through the network to update weights. The actual (soft) update is done in `soft_update`.

The `act` method is used to return the action suggested by the epsilon-greedy policy for the current state.

The `save` and `load_weigths` functions are used to save trained agents for further use and to load previously-trained agents, respectively. They are quite straightforward, only some checks for stringent dimensions of the network are done in the `load` method.

The `ReplayBuffer` class is taken from the lessons and comprises an initialization method as well as methods for adding experiences, sampling from the stored experiences, and a function for returning the current length of the buffer.



(a) Original.

(b) Moving average with window size 5.

Figure 1: The scores reached by the intelligent agent.

2 Results

As Figure 1 shows, the agent is capable of learning quite fast, the environment has been solved within 485 episodes.

3 Ideas for Improvements

With more fine-tuning of the hyperparameters or potentially introducing further layers, the ability of the agent to learn could potentially be speeded up. Besides, it is questionable if even higher scores can be attained, the current ages seems to saturate around an average score of 13-16.