Navigation Project Borja González

Deep Reinforcement Learning Nanodegree December 27th, 2018

Overview

The goal of this report I briefly summarize the learnings and final modeling decisions taken as part of the Navigation project.

Deep-Q-Network Architecture

Since the input-space was relatively small (37 inputs) a FFNN was enough for the non-visual approach as a function approximator. The final architecture was made of 3 hidden layers made of 128, 64 and 64 neurons.

Q-learning mode.

As can be seen from the results, the above-mentioned architecture is enough to greatly outperform the proposed solution (which took almost 1800 episodes to solve the task) with the vanilla DQN so I used it for this first delivery

Choice of hyperparameters

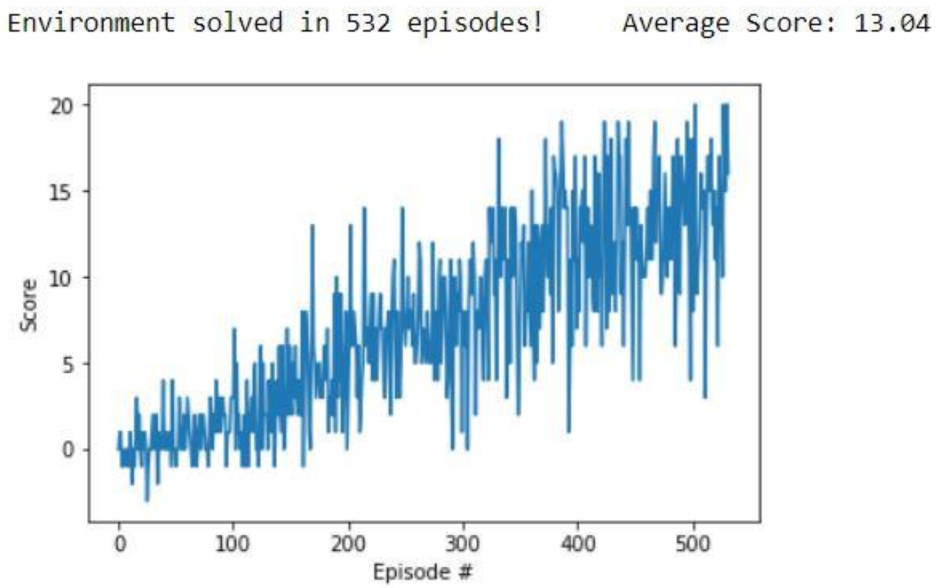
Hyperparameters from previous Udacity exercises, which are also common in DRL papers worked well for this problem also:

- Memory buffer size: 100 000
- Learning minibatch size: 64
- Discount factor for future rewards: 0.99
- Soft updating factor for target network's parameters: 0.001
- Learning rate: 0.0005
- Number of steps before updating the target network: 4

```
BUFFER_SIZE = int(1e5)   # replay buffer size

BATCH_SIZE = 64        # minibatch size

GAMMA = 0.99           # discount factor

TAU = 1e-3             # for soft update of target parameters

LR = 5e-4              # learning rate

UPDATE_EVERY = 4       # how often to update the network
```

Results plot

Below I include a plot with the results of the network training the DQN agent.



```
Environment solved in 532 episodes!     Average Score: 13.04
```

Further Work

First steps will be about optimizating the DQN algorithm (up to the rainbow approach) as well as testing the visual-input environment and CNNs.