Deep Reinforcement Learning Nanodegree – Project 1 “Navigation”

Bernhard Langwallner

August 26, 2018

In this report I briefly summarize the learnings and final modeling decisions taken as part of the Navigation project. I observed it was possible to solve the environment in $400 - 500$ episodes with a number of DQN architectures and choices of ε. In the following I highlight the setup that worked best in my experiments.

Deep Q-Network Architecture.
After experimenting with different numbers of layers (particularly 2, 3 and 4) I concluded that 3 standard feed-forward layers with ReLu activation was a good compromise. With state space dimension of 37 and output/action space dimension of 4 the problem is not too high-dimensional, so too high numbers of hidden layers or units within the layers do not seem to be justified.

I observed fairly good results with different choices of hidden layers: e.g., hidden layer sizes of 256/128/64, 128/128/128, 128/64/32, 64/32/16. Given the right choice of ε (see below), all these architectures allow to solve the problem in well below 1000 episodes. I finally chose 128/64/32 for the sizes of the three hidden layers.

Q-Learning mode.
I experimented with both plain deep Q-learning, as well as double deep Q-learning [1]. During experimentation Double DQN seemed to outperform plain DQN, so I used it in my final version.

Choice of ε.
As outlined in the lectures and as observed in some of the previous coding exercises, the choice of ε (starting value, decay factor/speed, final value) has a large effect on the speed of learning. I decided to go with a multiplicative decay with minimum value in the long run: $\varepsilon = \max(\varepsilon_0 \cdot \varepsilon_{\text{decay}}^k, \varepsilon_{\min})$, where k denotes the episode. After starting with relatively conservative, and previously seen values of $\varepsilon_{\text{decay}}$ (e.g., 0.995) I decreased it further and further and observed quite fast training progress. My interpretation is that the environment is not too complex (i.e. it does not generate too much variation in the state space) and thus the agent needs relatively little exploration compared with other environments.

The final choice of parameters was as follows:

$$\varepsilon_0 = 1, \qquad \varepsilon_{\text{decay}} = 0.97, \qquad \varepsilon_{\min} = 0.005.$$

Results.
As outlined above the environment could be solved in < 500 episodes with an array of different architectures and choices of ε. Here we have the results of a training run with the final setup. After an initial phase with little progress (due to high share of exploration) the agent starts to obtain higher scores after ~ 50 episodes. After ~ 200 episodes the progress seems to flatten but the average scores still rises until average score of 13 over 100 episodes is reached.

Further work.
I observed the actual number of episodes required to solve the environment to change a lot with

[1] see the paper “Deep Reinforcement Learning with Double Q-learning” by van Hasselt, Guez, Silver

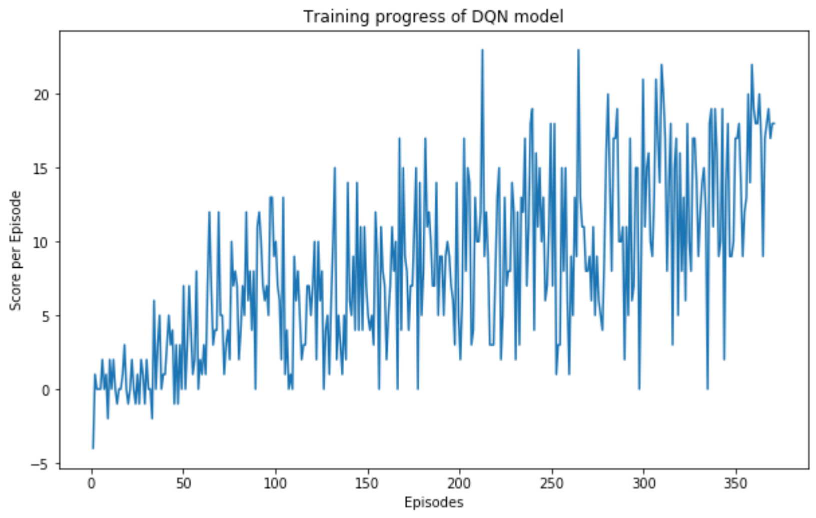

Figure 1: Example training run of the agent. The environment was solved in well under 500 episodes.

each training run, which might have to do with the random nature of initial condition, and probably more strongly, with experience replay. I thus think, further research into stabilizing the training procedure could be valuable. Particularly going away from purely random choice of steps from the memory buffer and using prioritized experience replay could be interesting. As to the DQN mode, I doubt in this simple environment more complex setups (dueling DQNs etc.) would mean a game change in training speed. It would, however, be interesting to try out models to be looked at later in the Nanodegree on this environment (policy based, actor-critic etc.).