Deep Reinforcement Learning - Project 1

Peter Toke Heden Ahlgren[1]
[1]Mjølner Power ApS

Keywords: Navigation, Dueling, DoubleQ 24th February, 2021

1 Introduction

In the following report we solve the Unity Environment "Banana" with reinforcement learning agents. The environment is a square world, with randomly placed yellow and blue bananas. Each yellow banana picked up by the agent gives a score of $+1$, while a blue gives a score of -1. For the current project an average score of 13 over 100 episodes of the game is considered a success. The agent gets 37 different inputs as its state and can take any of 4 actions: *forwards*, *backwards*, *left* and *right*.

2 Reinforcement Learning Agent

As agent we use a deep reinforcement learner, *i.e.*, a reinforcement learner agent where the Q-table is replaced by a neural network. For this structure to successfully learn, a few tricks has to be implemented: instead of only one network replacing the Q-table we use a *local* and *target* network, such that it is not the same network that for every step is updated and take an action - we also utilize a replay buffer, giving the agent memory, such that also old experiences can still be used. Besides this basic structure, we also experiment with dueling network structure and double Q learning.

The dueling network structure, adds a second head to the network, such that the inputs are fed into the same structure, but after passing a few common hidden layers, are fed through an advantage network, calculating the advantage of taking different actions, and a value network estimating the state value function, stabilizing learning.

Double Q learning prevents the tendency of the agent to overestimate Q-values, especially in early stages of training, since the algorithm takes the action having max Q-value and the

max of "noisy" numbers has a tendency to be higher than the expected value. In Double Q learning, both the local and target network are used in choosing the best action, such that a value that is wrongly the highest in one network, will be affected in opposite direction if the two networks do not agree.

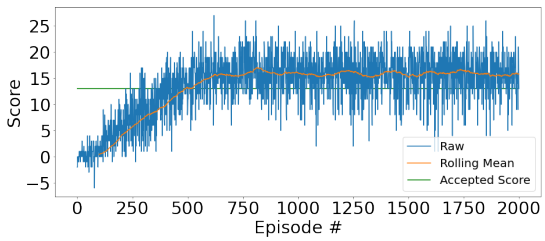
3 Chosen Parameters

For the basic structure, we choose a network having 3 hidden layers, with 64 nodes in each. For the dueling networks structure, we choose 2 hidden layers for the common network, and 1 for each of advantage and state value networks. Number of nodes is 64 in all of them. Activation functions are ReLU between all layers in both setups.
For all examples we use the same values of the following variables:

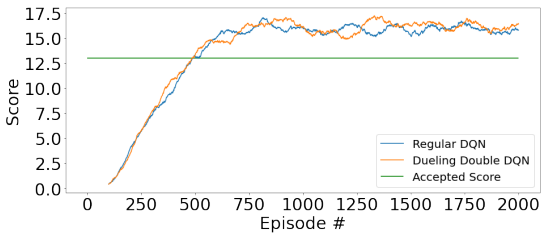
- Replay buffer size: $1\mathrm{e}5$
- Minibatch size: 64
- Discount factor, γ: 0.99
- Parameter for soft updating of target, τ: $1\mathrm{e}{-3}$
- Learning rate: $5\mathrm{e}{-4}$
- Update network every n'th step: 4
- ϵ-greedy start value: 1.0
- ϵ-greedy end value: 0.01
- ϵ-greedy decay value: 0.995

4 Results

The resulting rewards for the basic agent structure is depicted in figure 1a. As seen it takes not much more than 500 episodes for the agent to achieve performance that is better than the desired average score of 13. However, the average score still increases and after plateauing



(a) Reward for the basic agent structure. Blue the raw data, orange the running average.



(b) Comparing running average rewards of the basic agent structure (blue) and and agent utilizing both dueling networks as well as double Q (orange).

Fig. 1. The reward dynamics for the examined agents. In green, the average of 13 for 100 episodes is depicted, while smooth lines are running averages for equally many episodes.

it stays at a level of around 15, having a lot of variation, but not more than the benchmark results, provided in the project description. We use the agent state at episode 800 as our final agent, and accept that as the final solution.

After seeing the basic vanilla implementation one could have high hopes for what would happen if we implemented yet more advanced agent structures. However, with the 2 approaches used in the current project, no real difference is seen. Figure 1b shows the running average reward for the basic and most complex agents (the most complex agent is the one using both double Q and dueling networks) and it is seen that there is not any real difference in either the speed of convergence or the max level of performance.

This is somehow surprising, though different mechanisms might work for one environment, but not for others. Something very similar showed in the example of the Lunar lander, solved earlier in this course. Finally, after all, the performance of the basic agent is not that bad either. Maybe it is just hard to make a lot better.

5 Future Work

It seems natural to start future work by figuring out why there is not observed any difference between the basic and complex agent. After looking into that, implementing the remaining features of the "Rainbow" architecture, would be a nice addition. Finally a thorough analysis of the best performing network structure would also be very interesting, but might be the

most difficult to actually conclude anything from, except that this or that number and size of hidden layers, gave this performance.