Project Navigation

Introduction

In this project an agent will be trained to collect bananas in a limited square world. The agent can take four actions: forward, backward, left and right.

The agent gets a +1 in reward for a yellow banana and a -1 in reward for a blue banana.

For the state space there are 37 different states which includes, velocity and ray based perception of the robots forward looking direction.

The task is to get an average score of 13 in 100 episodes.

Method and parameters

In this project I did several different tests of approaches and settings to get a better feeling and understanding of the different methods. Below is a short summary of all the tests I did. DQN stands for Double Q Network.

- Test1: DQN, min_eps=0.1, eps decay_rate=0.995
- Test2: DDQN, min_eps=0.1, eps decay_rate=0.995
- Test3: DDQN min_eps 0.01, eps decay_rate=0.995
- Test4: DQN min eps 0.01 eps decay_rate=0.995, experience replay
- Test5: DDQN min_eps 0.01, eps decay rate=0.995
- Test6: DQN softmax, slow temp decrease
- Test7: DQN softmax, fast temp decrease
- Test8: DQN softmax fat temp drop with priotized experience replay

Hyper parameters that remained constant:

- Episodes: 1000
- Time steps: 1000
- Batch size: 64
- Gamma: 0.99
- Tau(Double DQN update factor): 0.001
- Update target network every: 4 episodes

Neural network for DQN

- Optimizer: Adam
- Learning rate: 0.0005
- Layer config:
    - Input: Number of states(37)
    - Inner: 64
    - Output: Number of actions(4)
- Activation function: Relu

Many different neural network for dueling DQN were tried. None of them were successful the average 100 rolling scores always regressed back to around 0. I cant really figure out why dueling DQN was so hard to make it work.

For all the methods the folling was never changed.

- Eps start 1.0 for all relevant eps approaches.
- Temperature 1000 start for all soft max approaches.

Every method test ran for 1000 episodes with a max of 1000 time steps, in reality the environment only allows 299 time steps, though.

For temperature decay I tried two different decay curves. These curves can be seen in Figure 1. In this figure we can see slow and fast decay of the temperature. There doesn't seem to be such a large difference from this graph, however, in the results we can see a significant difference in performance between these two.

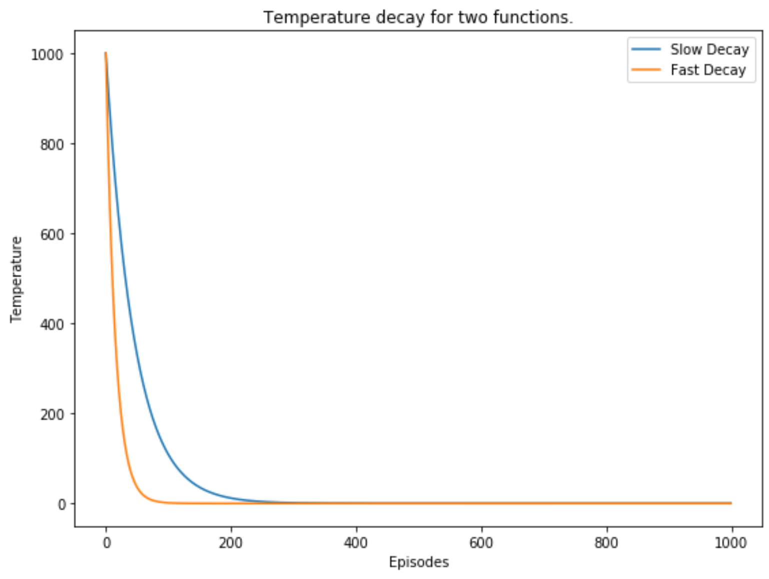

Figure 1: Temperature decay for two decay methods called slow and fast decay.

Results

The results from the different methods can be seen in Figure 2. Here the average rolling mean of the 100 last scores are plotted. In addition we also have a horizontal line that indicates the requirement to pass the assignment, the score of 100. Well to be correct the rolling 100 average should be above 13. From the figure we can see that all of the methods pass over the horizontal line at least for some time except for one method. The results from test 4, the dueling DQN, performs poorly and seems to have an asymptote around 0. Test1 and test2 barely passes the requirement. We can see quite a significant difference between test2_ddqn and test2_ddqn_min_eps_0_01. test2_ddqn_min_eps_0_01 does much better than test2_ddqn and the only difference between the is that the former have a min eps of 0.1 and the latter a min eps of 0.01. From this we can say that we need to make sure that we reduce the probability to explore otherwise we don't give the method a change to exploit enough to accumulate more scores.

When we start to use softmax with temperature for action selection we can see that we do quite well when we a decay rate of the temperature that is high enough. There is a very big difference between having a high and a slow decay of the temperature. The “fast decay” approach seemed to have been the best method in this project. It reaches the 13 score requirement the fastest and keeps going to higher scores and seems to stabilize around 16.

We can see that prioritized experience replay in combination with softmax also does quite well. Its not that easy to compare as I only gave the prioritized experience replay a buffer of 10000 samples to increase computational speed which in comparison to the regular experience replay which had 100000 samples in memory is quite a large difference. It would have been interesting to see how much better prioritized experience replay with a larger bugger size would have been.

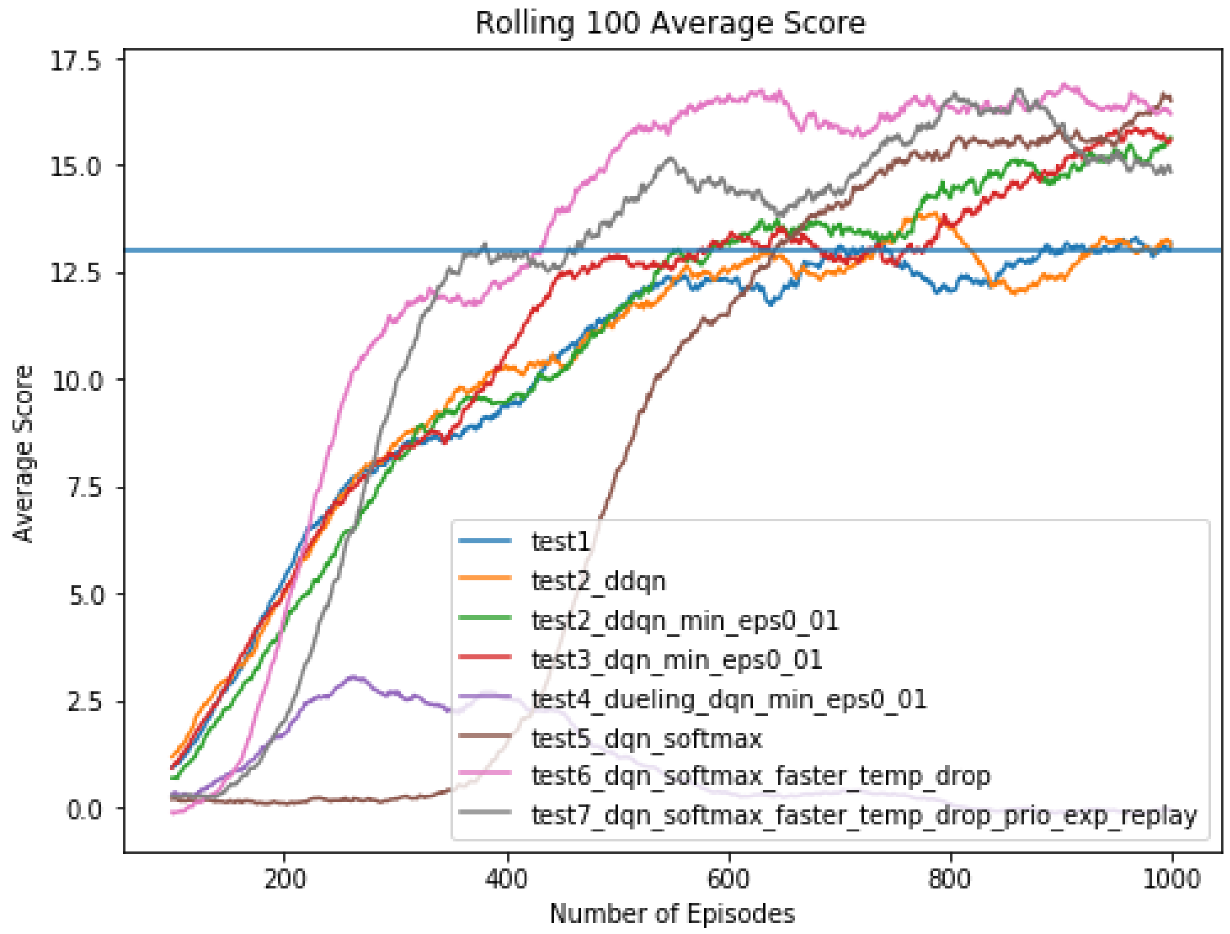

Figure 2: Rolling 100 average score for different methods.

I was trying to see if there was an interesting difference in the variance of the scores of the different methods. However when I have plotted the rolling 100 standard deviation of all the episodes I cant really see any clear indication of that. In Figure 3 the rolling 100 standard deviation is plotted for all the different methods.

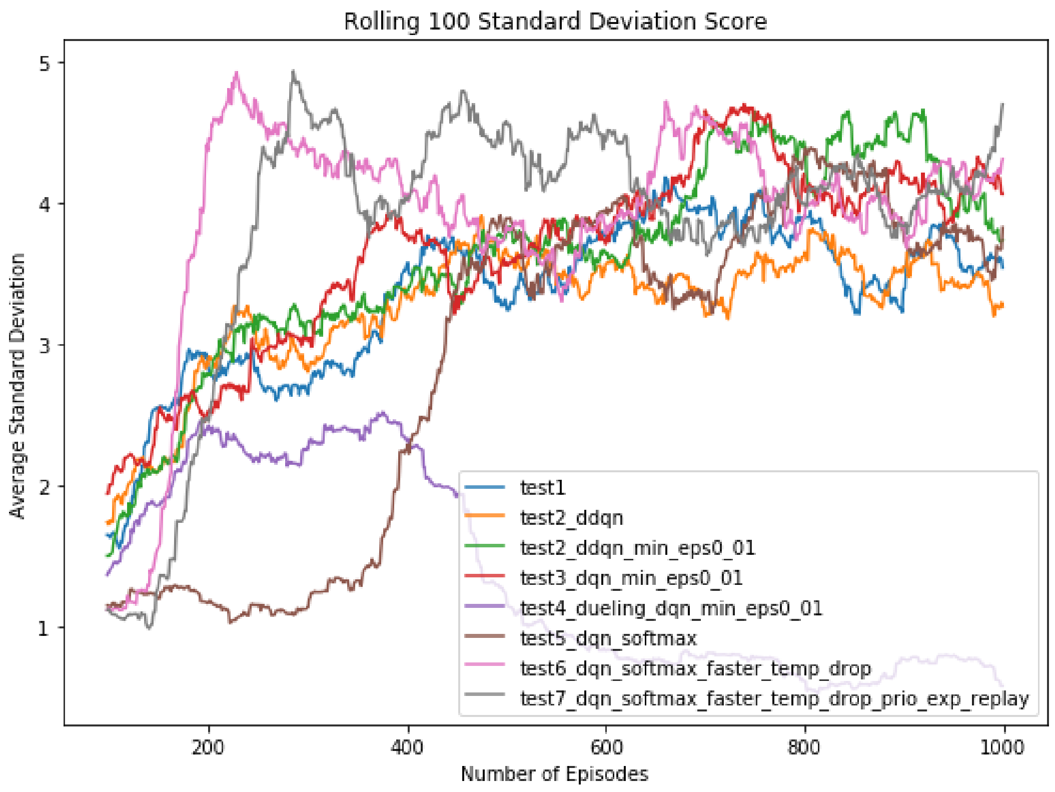

Figure 3: Rolling 100 standard deviation for the different methods.