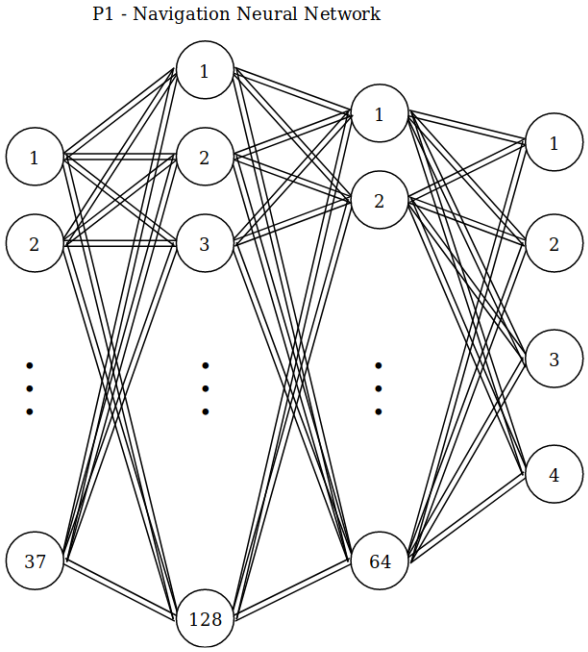
Mathias Babin
08/27/2018

P1 – Navigation Report

1) Learning Algorithm

For this project I implemented the Deep Q-Learning Algorithm, both Double DQN and Priority Experience Replay(PER) improvements. The Neural Network architecture I used includes 37 input neurons, with 128 neurons in first hidden layer, 64 in the second, and 4 in the output layer; This of course corresponds to the 37 values that define the environments state space, and the 4 possible actions the agent can take. For this network, learning rate was set to 0.0005.



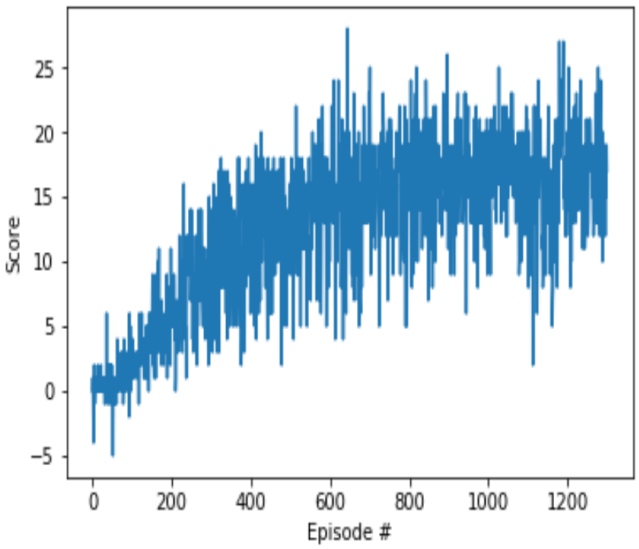
PER was implemented using a sum tree as the data structure for the replay buffer, as it achieves O(l*og n*) time for sampling. PER's coefficient 'a' was set to 0.95 as I wanted sampling to mostly occur on experiences with high priority values, and epsilon was set to only 0.01. The size of this replay buffer was 150 000, and batches contain 64 experiences.

Updates to the network occur every 5 timesteps, and soft updates to the network used a Tau of 0.001. As an observation, adjustments to Tau of 0.01 and 0.0001 seemed to make updates too volatile to learn efficiently, or made the network learn too slowly.

2) Results

The agent managed to solve the environment in roughly 500 episodes, depending on weights it started with this value would vary between 480-560. By 1300 episodes it always achieved an average score of at least 16 over 100 consecutive episodes.

```
Episode 100 -> Avg Score: 0.611
Episode 200 -> Avg Score: 3.844
Episode 300 -> Avg Score: 7.844
Episode 400 -> Avg Score: 10.688
Episode 500 -> Avg Score: 12.311
Episode 557 -> Avg. Score: 13.00
Environment solved on episode 557.
Episode 600 -> Avg Score: 13.555
Episode 700 -> Avg Score: 14.933
Episode 800 -> Avg Score: 15.699
Episode 900 -> Avg Score: 16.666
Episode 1000 -> Avg Score: 16.244
Episode 1100 -> Avg Score: 16.788
Episode 1200 -> Avg Score: 15.755
Episode 1300 -> Avg Score: 16.677
```



3) Future Improvements

To further improve the performance of the agent in this project, Dueling DQN could be implemented. Also I did not adjust the weights of the network to properly account for PER sampling, and therefore they will be biased. Finally, more experimentation with the NN architecture could lead to potentially better results, but of the architectures I tried this one gave the best results.