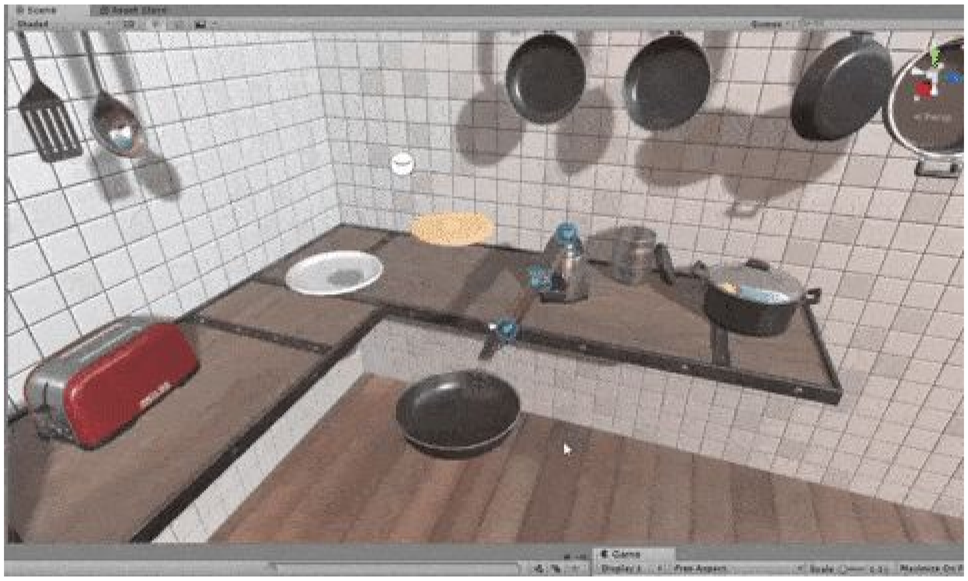
Udacity Beta Testing _ Navigation Project

Beta Testing Report

July 1st, 2018

Algorithm Overview

The goal here is to develop a functional code after understanding the course content. In this process test the instructions and video content for its correctness. I simply used DQN as explained in the course. I modified the model and tuned hyper parameters. Reinforcement learning is essential when it is not sufficient to tackle problems by programming the agent with just a few predetermined behaviors. It is a way to teach the agent to make the right decisions under uncertainty and with very high dimensional input by making it experiencing scenarios. In this way, the learning can happen online and the agent can learn to react to even the rarest of scenarios which the brutal programming would never consider.



DQN ARCHITECTURE

In the current architecture I have 3 hidden layers with 128 neurons. Input is a 37 state vector and output is 4 action vector containing score. I used xavier weight initialization for faster training time. Additionally, there is a ReLu. score for the 4 actions is given to the loss function. We get a score value for each of the actions for deciding on the best.

The whole DQN architecture and the Q-learning setup was developed in python using numpy and matplotlib libraries and Unity environment.

Agent

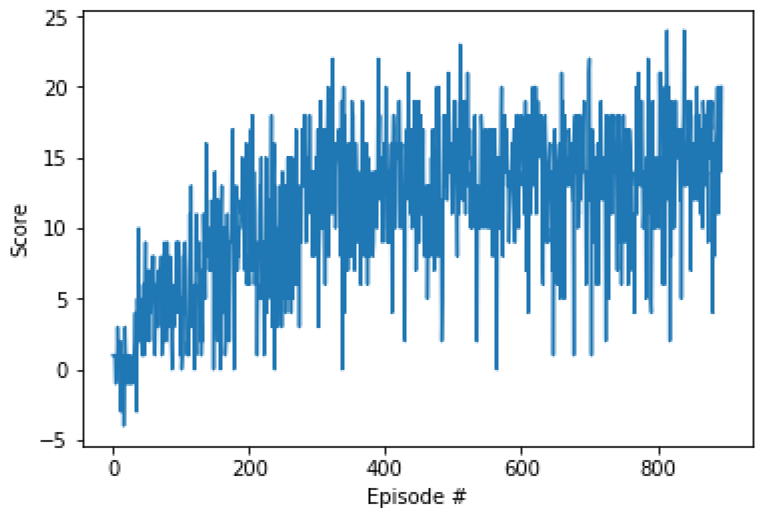
DQN parameters: The exploration probability is linearly decreased from 1.0 to 0.01 in 1000 updates. The size of the replay memory is set to 100000 and the mini-batches are sampled once it has 500 experiences.

Training

Udacity workspace did not work for me for first 2 days. I used my Mac laptop. Trained for 2000 episode. After getting the average score of 15.0, I saw that it gradually decreases, so I stopped at 15 (early stopping to avoid overfitting) The pipeline for the entire DQN training process is shown in Algorithm 1 in the notebook.

Training parameters: The Gradient descent update rule used to update the DQN parameters is Adam with a learning rate 5e − 4, GAMMA = 0.99 and TAU = 0.001. These parameters were chosen on a trial and error basis observing the convergence of the loss value.

```
Episode 100	Average Score: 3.28
Episode 200	Average Score: 7.44
Episode 300	Average Score: 10.16
Episode 400	Average Score: 12.95
Episode 500	Average Score: 13.35
Episode 600	Average Score: 13.74
Episode 700	Average Score: 13.62
Episode 800	Average Score: 13.06
Episode 895	Average Score: 15.02
Environment solved in 795 episodes!	Average Score: 15.02
```



RELATED WORK

Google Deepmind’s efforts to use Deep learning techniques to play games have paved way to looking at artificial Intelligence problems with a completely different lens. Their recent success, the Go agent that has been giving a stiff competition to the experts.

- I like to implement some improvements such as prioritized experience replay, Double DQN, or Dueling DQN! In next few day.

Overall, I enjoyed this work.