Deep Reinforcement Learning Nanodegree

Project 1: Navigation

Learning Algorithm

1. Algorithm

   I use a deep Q-learning with experience replay to approximate the optimal action-value function. Also, an action is determined by Epsilon Greedy.

2. Hyperparameters

   I used the following hyperparameters in my code:

   - BUFFER_SIZE = int(1e5) # replay buffer size
   - BATCH_SIZE = 64 # minibatch size
   - GAMMA = 0.99 # discount factor
   - TAU = 1e-3 # for soft update of target parameters
   - LR = 5e-4 # learning rate
   - UPDATE_EVERY = 4 # how often to update the network

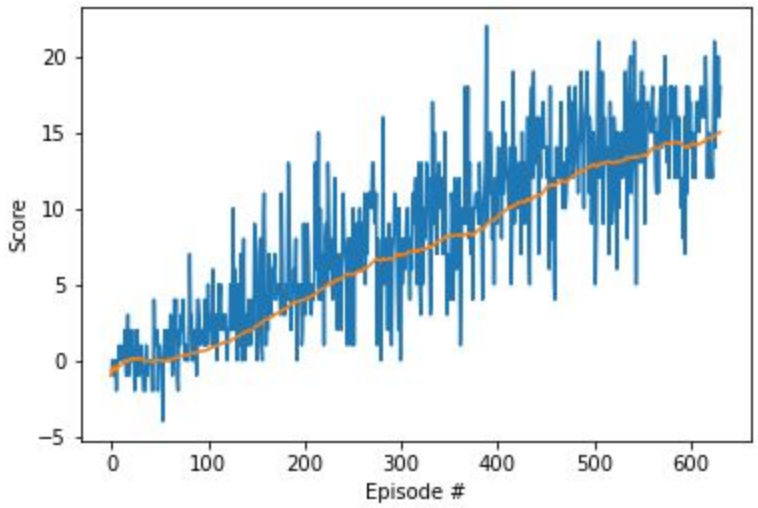
3. Model architectures

   The architecture of Q network and the number of nodes in each layer is as follows:

   - Input layer = 37 (state_size)
   - Hidden layer 1 = 32
   - relu
   - Hidden layer 2 = 16
   - relu
   - Output layer = 4 (action_size)

Plot of Rewards

The reward graph is shown below. The blue line is the score for each episode. The orange line is the average of the last 100 scores. My agent got a score of +13 in about 500 episodes.



Ideas for Future work

1. Double DQN (DDQN)
2. Prioritized experience replay
3. Dueling DQN
4. Learning from multi-step bootstrap targets (as in A3C - *you'll learn about this in the next part of the nanodegree*)
5. Distributional DQN
6. Noisy DQN