Report

Implementation of Learning Algorithm

DQN
A network used in this project contains three fully connected layers that produce Q values.
(implemented in ***model.py***)

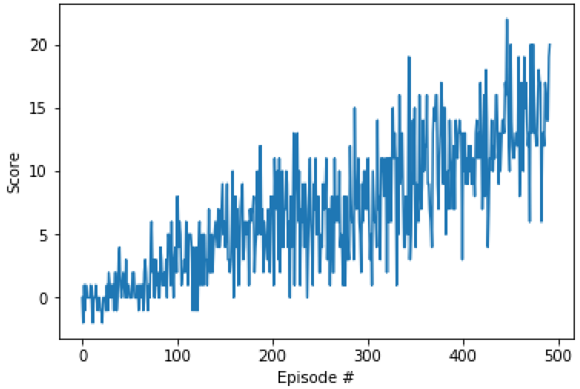
In DQN, there are two main processes to get Q values.

1. [Sample] : Store the observed experienced tuples in a replay memory.
   The agent store observed experienced tuples in a replay memory when function *step()* in class *agent is* called.
   (implemented in ***agent.py***)
2. [Learning] : Learn from the batch sampled from the memory using a gradient descent update step.
   The function step() decide to learn according to the param UPDATE_EVERY. If enough samples are available in memory(> BATCH_SIZE), data in replay buffer is randomly sampled. Q values are updated with new reward and discount factor GAMMA. Weights of models is updated with the param TAU.
   (implemented in ***agent.py***)

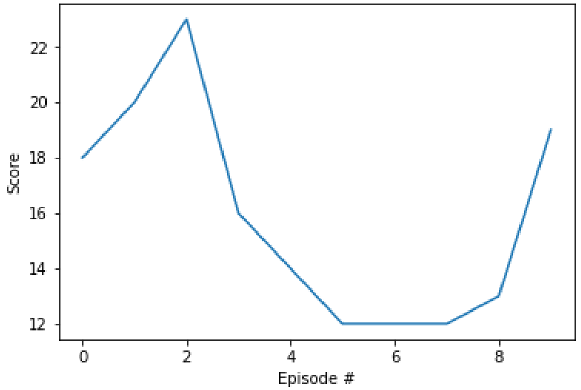
Epsilon-greedy action selection
To solve exploration vs. exploitation dilemma, Epsilon-greedy algorithm provides minimum exploration portion when calling agent.act() with hyperparameters (eps_start=1.0, eps_end=0.01, eps_decay=0.995) to select actions.
(implemented in ***agent.py***)

Report

Plot of Rewards



Training Result



Test Result

Episode 100	Average Score: 0.98
Episode 200	Average Score: 4.59
Episode 300	Average Score: 6.46
Episode 400	Average Score: 9.62
Episode 492	Average Score: 13.04

Environment solved in 392 episodes with average score 13.04.

Report

Ideas for Future Work

Advanced techniques such as Double DQN(DDQN), prioritized experience relay or Dueling DQN can be applied.

- DDQN
  Double Q-learning can make estimation more robust by selecting the best action using one set of parameters w, but evaluating it using a different set of parameters w'. These two value functions are basically maintained and randomly choose one of them to update at each stem using the other only for evaluating actions.*

  TD Target : $R + \gamma\, \hat{q}\,(S', \arg\max_a \hat{q}(S', a, w,), w')$

- Prioritized experience relay
  Prioritized experience relay set priority on tuples in the replay buffer to prevent important experiences from getting lost. $\left(\frac{1}{N} \cdot \frac{1}{p_i}\right)^b$ in new update rule below stands for the importance-sampling weight.

$$\Delta w = \alpha \left(\frac{1}{N} \cdot \frac{1}{p_i}\right)^b \delta_i \nabla_w \hat{q}(S_i, A_i, w)$$

- Dueling DQN
  The core idea of dueling networks is to use two streams
  - one stream estimates the state value function : $\mathrm{V(s)}$
  - one stream estimates the advantage for each action: $\mathrm{A(s,a)}$
  - final Q values : $\mathrm{Q(s,a) = V(s) + A(s,a)}$