Deep reinforcement learning Project 1 report

This is a report for deep reinforcement learning project 1.

Code structure

The structure of the code is split into 3 files, Navigation.ipynb, model.py and agent.py. The main procedures, such as initialising the agent, initialising the unity environment, and looping through the episodes. Model.py contains the Pytorch implementation of neural network. Agent.py contains the q-learning agent implementation, where the q-learning updates and calls model.py.

Algorithm

The Deep Q learning algorithm is as explained in the lessons. A brief description of the algorithm is stated here.

The goal of the agent is to always select actions that gives the maximum reward At each state (s), the agent would chose an action(a), and each action would yield a reward (r). The agent would learn in multiple episodes in order to chose the best policy. The action value function is defined as

$$Q^*(s,a) \quad = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') | s,a\right]$$

However this method is known to be unstable, and 2 modifications have to be applied. That is the experienced replay and periodically adjust the action values against the target value.

Experienced replay is where the agent's experience at each time step is stored into a buffer. During the Q learning update, a sample is drawn randomly from the pool of stored samples.

A different network is required to generate the targets in the Q learning update. Every C updates, clone the network Q to obtain $\hat{Q}$ and then use $\hat{Q}$ for the next C updates before getting the new Q value.

Chosen hyperparameters:

For dqn:

- n_episodes=2000
- max_t=1000
- eps_start=1.0
- eps_end=0.01
- eps_decay=0.995
- BUFFER_SIZE = 100000
- BATCH_SIZE = 64
- GAMMA = 0.99
- TAU = 0.001
- LEARNING_RATE = 0.0005
- UPDATE_EVERY = 4

ANN architecture

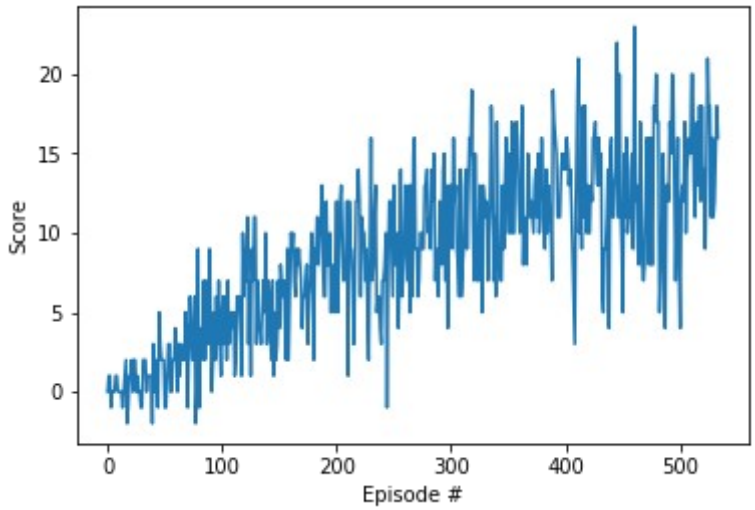
A total of 4 fully connected networks with relu activation functions. The input dimension and output dimensions are as shown:

1. hidden layer 1
    - input: no. of states

    - output: 128
2. Hidden layer 2
    - input: 128
    - output: 164
3. Hidden layer 3
    - input: 64
    - output: 32

Final results

The agent have to achieve an average score of 13 or more for 100 consecutive episodes.
The model managed to solve the environment in 434 episodes to achieve the score.



Ideas for future work

CNN could be tested out to see if it could take even lesser episodes to achieve the score. Also a larger range of hyperparameters can be tried to see if the agent would be better.