Project report

Deep Q Learning algorithm

The deep neural network has following layers:

- Fully connected layer - input: 37 (state size) → output: 64
- Fully connected layer - input: 64 → output: 64
- Fully connected layer - input: 64 → output: 4 (action size)

Parameters used in DQN algorithm:

- Maximum steps per episode: 1000
- Starting epsilon: 1.0
- Ending epsilon: 0.01
- Epsilon decay rate: 0.8

DQN Architecture:

a) The main loop of the DQQN method executes the specified number of episodes (1,000 in here), for each episode, follows the steps:

1. Reset the Environment and captures information from env_info record containing information like current state, current reward, and current episode state.

2. Records the current state information

3. Initializes the score for the current episode to 0.

4. Finally, executes a set of time steps making up a single episode (1,000 in this case).

5. Save the model if average score more than 13 and continue.

b) For each 1,000 time steps of an episode, follows the steps below:

1. Obtains from the agent the next action to take (Based on the current state and the current value of epsilon)

2. Tells the environment to take a step using the indicated action.

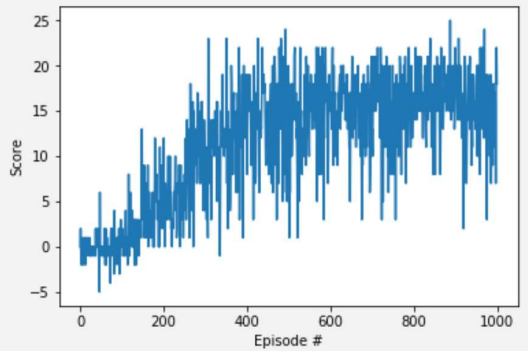
3. Obtains the next state from environment, reward is based on the current state and the specified action, and indication of episode state.

4. Asks the agent to make a training step, based on current state, specified action, next state and the resulting reward.

Result:

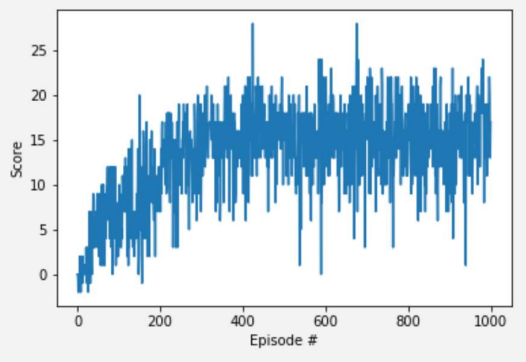
With First layer output set as 128

Episode 100 Average Score: -0.09
Episode 200 Average Score: 2.945
Episode 300 Average Score: 7.50
Episode 400 Average Score: 11.80
Episode 435 Average Score: 13.00
Environment solved in 335 episodes! Average Score: 13.00



With First layer output set as 64

Episode 100 Average Score: 4.15
Episode 200 Average Score: 8.34
Episode 299 Average Score: 13.03
Environment solved in 199 episodes! Average Score: 13.03



Ideas for future work

1. Increase number of layers in the model, because seems like increasing number of neurons makes the model learn slow.
2. Implement degrading learning rate for training. This will help faster learning in initial stages and closure on better results using lower learning rate in final stages.
3. Implement Double Deep Q Network