Report

June 29, 2019

1 Learning Algorithm

The learning algorithm used to train this agent was DQN(Deep-Q Network).

It follows this sequence:

Learning

- Obtain random minibatch of tuples (s,a,r,s′) from D
- Set target yi=r+gamma * max(q(s′,a,w-)
- Update weights (yi - q(s,a,w)) * gradient(w)q(s,a,w)
- Every c steps w <- w-

2 DQN Agent:

- Action Value model:

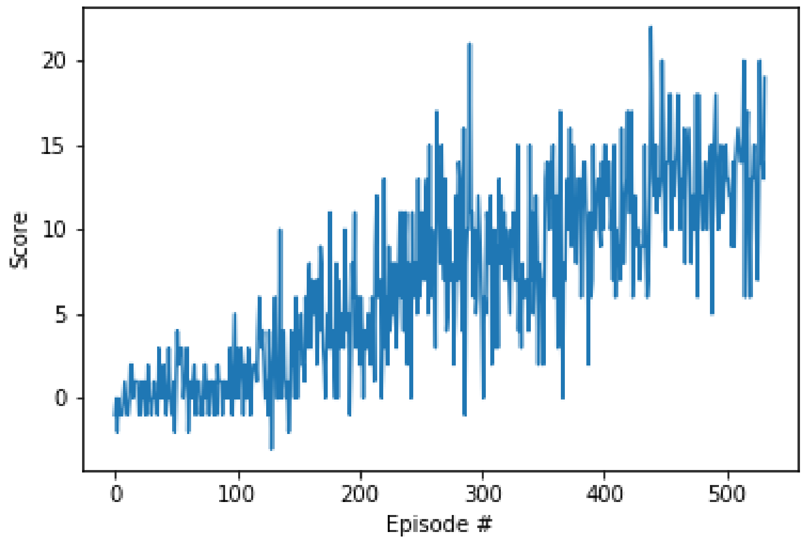
```
Input: state_size (int): 37
Output: action_size (int): 4
Layer 1 - fc1_units (int), Number of nodes in first hidden layer: 64
Activation Layer 1: RELU
Layer 2 - fc2_units (int): Number of nodes in second hidden layer: 64
Activation Layer 2: RELU
```

- Agents Hyper Parameter:

```
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 Plot of Rewards

```
Episode 100 Average Score: 0.52
Episode 200 Average Score: 3.25

Episode 300 Average Score: 7.45

Episode 400 Average Score: 9.02

Episode 500 Average Score: 12.30

Episode 532 Average Score: 13.04

Environment solved in 432 episodes! Average Score: 13.04
```



BaseScores.png

4 Ideas for Future Work

It would be interesting to try the following modidifications:
Double DQN (DDQN) Prioritized experience replay Dueling DQN
It would also be interesting to try learning directly from the pixels