P1:Navigation

Learning algorithm

In this project I implemented DQN(Deep Q-Network) algorithm. DQN is a value-based algorithm. It origins from Q-learning. But Q-learning cannot deal with big state space because of the dimension disaster of Q table. DQN combines Q-learning with neural network. It is a good way to replace Q-table with function. This process is also known as Function Approximation.

Model

The model is defined in file model.py with PyTorch. It consists of 3 fully connected layers:

- Layer1: 37 input and 64 output, activation function ReLU
- Layer2: 64 input and 64 output, activation function ReLU
- Layer3: 64 input and 4 output, activation function ReLU

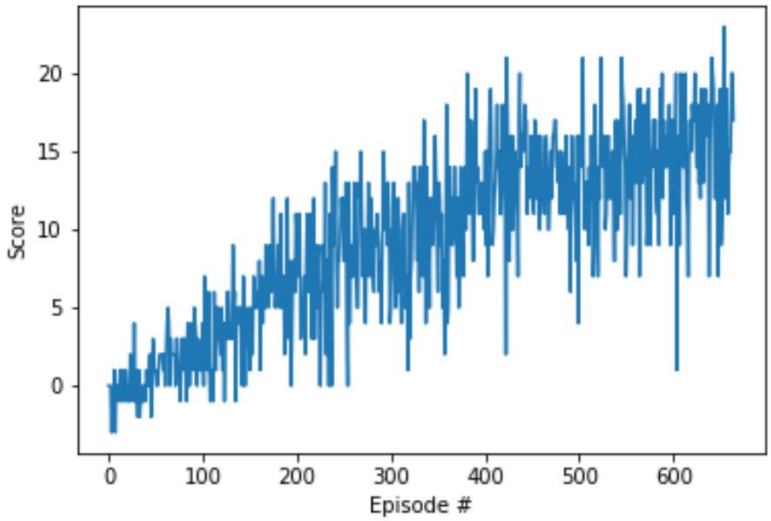
Since the dimension of state space and action space is 37 and 4, the dimension of input and output is also 37 and 4.

Hyperparameters

- Replay buffer size = 1e5
- Batch size = 64
- Gamma = 0.99
- TAU = 1e-3
- Learning rate = 5e-4
- eps_start=1.0, eps_end=0.01, eps_decay=0.995

Plot

The agent achieve the target score/solves the environment at episode 620.




```
Episode 100     Average Score: 0.73
Episode 200     Average Score: 3.76
Episode 300     Average Score: 6.79
Episode 400     Average Score: 9.94
Episode 500     Average Score: 12.33
Episode 600     Average Score: 13.81
Episode 700     Average Score: 14.68
Episode 720     Average Score: 15.11
Environment solved in 620 episodes!     Average Score: 15.11
```

Future Work

Try different models such as Double DQN, Dueling DQN and make a comprison.

Find an efficient way to tune the hyperparameters(by further reading)