P1 Navigation Project Report

Implementation:

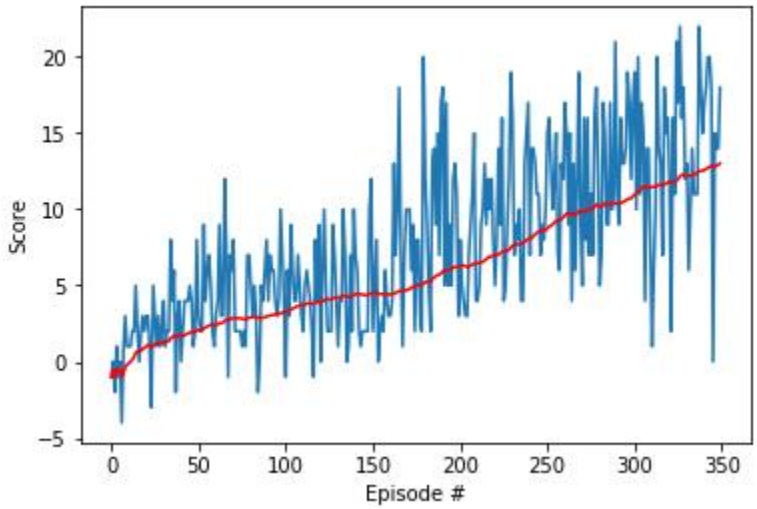
- The learning algorithm is DQN, a value based RL learning algorithm. The following are the code structure of the implementation.
    - File dan_agent.py includes definition of Agent and ReplayBuffer class.
    - File model.py includes definition of neural network structure of DQN
    - File Navigation.ipynb includes the main training function dqn which will run a training procedure to complete the training
    - File checkpoint.pth included a trained model.

Learning Algorithm and Plot of Rewards:

- The DQN agent model is a neural network which includes following layers
    - 3 Linear (Dense) layers
    - 1st and 2nd linear layers are followed by relu activation layer
    - 3rd linear layer is output layer which outputs the value for each stage-action pair
- Here are the hyperparameters used in training and experiment results.
    - eps start=1.0, eps end=0.01, eps decay=0.5
    - BATCH_SIZE = 64, GAMMA = 0.99, TAU = 1e-3, LR = 5e-4, UPDATE_EVERY = 4

```
Episode 100     Average Score: 3.23
Episode 200     Average Score: 6.23
Episode 300     Average Score: 10.86
Episode 350     Average Score: 13.01
Environment solved in 250 episodes!     Average Score: 13.01

<matplotlib.figure.Figure at 0x7f70bf99d080>
```



- Conclusion:
    - In addition to the experiment presented above, multiple attempts had been done and the results show that the training converges faster by reducing eps_decay.
    - I also tested different learning rate and the result indicates that by reducing learning rate from 0.001 to 0.0005, the final convergence of the reward will increase from 15.0 to 16.0

Idea for Future Work:

- Change DQN to DDQN
- Apply Prioritized Experience Replay
- Try Dueling DQN
- Apply Actor-Critic learning algorithm
- Increase layers of original DQN neural networks