P3 Collaboration and Competition Report

Learning Algorithm

- The algorithm used here is MADDPG which includes two DDPG agents that share the same actor-critic structures. The following picture is the pseudo code of the MADDPG.

Algorithm 1: Multi-Agent Deep Deterministic Policy Gradient for N agents

for episode $= 1$ to M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial state $\mathbf{x}$
  for $t = 1$ to max-episode-length **do**
    for each agent i, select action $a_i = \boldsymbol{\mu}_{\theta_i}(o_i) + \mathcal{N}_t$ w.r.t. the current policy and exploration
    Execute actions $a = (a_1, \ldots, a_N)$ and observe reward r and new state $\mathbf{x}'$
    Store $(\mathbf{x}, a, r, \mathbf{x}')$ in replay buffer $\mathcal{D}$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    for agent $i = 1$ to N **do**
      Sample a random minibatch of S samples $(\mathbf{x}^j, a^j, r^j, \mathbf{x}'^j)$ from $\mathcal{D}$
      Set $y^j = r_i^j + \gamma Q_i^{\boldsymbol{\mu}'}(\mathbf{x}'^j, a_1', \ldots, a_N')|_{a_k' = \boldsymbol{\mu}_k'(o_k^j)}$
      Update critic by minimizing the loss $\mathcal{L}(\theta_i) = \frac{1}{S}\sum_j \left(y^j - Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_N^j)\right)^2$
      Update actor using the sampled policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{S} \sum_j \nabla_{\theta_i} \boldsymbol{\mu}_i(o_i^j) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(\mathbf{x}^j, a_1^j, \ldots, a_i, \ldots, a_N^j)\big|_{a_i = \boldsymbol{\mu}_i(o_i^j)}$$

    end for
    Update target network parameters for each agent i:

$$\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$$

  end for
end for

- Hyperparameters:

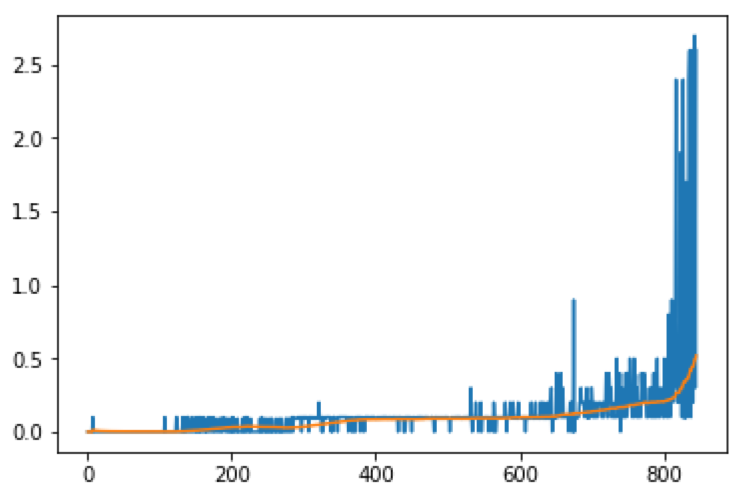
replay buffer size	1e6
minibatch size	128
discount factor (GAMMA)	0.99
soft update of target parameters (TAU)	0.01
learning rate of the actor	1e-4
learning rate of the critic	2e-4
L2 weight decay	0
how often to update the network	1
how many training steps in each network update	1
OUNoise noise decay rate (NOISE_DECAY)	0.9999
OUNoise internal setting	mu=0 theta=0.15 sigma=0.2 scale=1.0

- Neural Networks Structure:
  - DDPG Target and Local Actor Structure
    - Input = (batch, state size = 24)
    - self.fc1 = Linear (state size = 24, 256)
    - RELU
    - self.fc2 = Linear (256, 256)
    - RELU
    - self.fc3 = Linear (256, action size = 2)
    - Tanh

  - DDPG Target and Local Critic Structure
    - Input1 = (batch, state size = 24), Input2 = (batch, action size = 2)
    - fcs1 = Linear (state size, 256)
    - RELU
    - Concatenate (output of fcs1, Input2)
    - fc2 = Linear (256 + action size, 256)
    - RELU
    - Fc3 = Linear (256, 128)
    - RELU
    - Fc3 = Linear (128, 1)

Plot of Rewards

```
training loop:  42% |##################                           | ETA:  0:37:38
Episode: 841, score: 0.145000
training loop: 100% |###########################################| Time: 0:29:26
environment solved at episode 846
```



Ideas for Future Work

- Can try to use multiple PPO agents to solve the environment
- Can further investigate different network structure such as adding batch normalization layers, dropout layers or deeper linear layers with skip connection.
- Further optimize hyperparameter settings
- In each update step, the training always starts with agent 0 and then agent 1. We can alternate agent 0 and 1 as the first agent which gets trained.