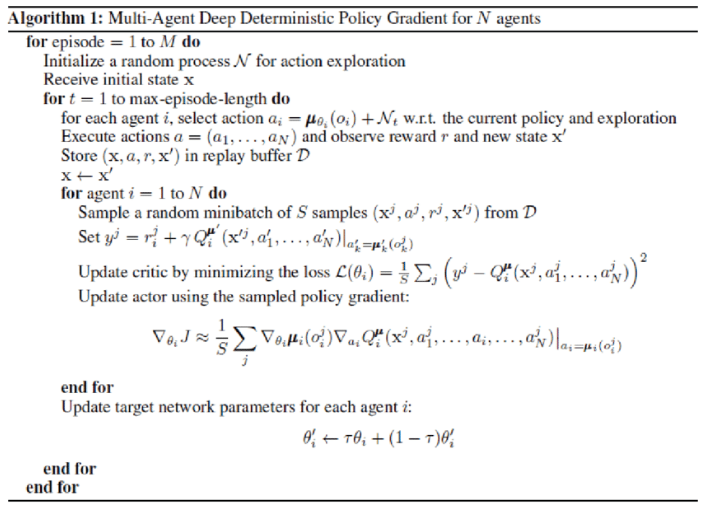
P3 Collaboration and Competition Report

Learning Algorithm

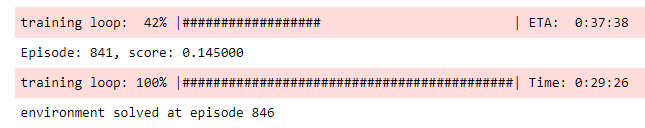
* The algorithm used here is MADDPG which includes two DDPG agents that share the same actor-critic structures. The following picture is the pseudo code of the MADDPG.

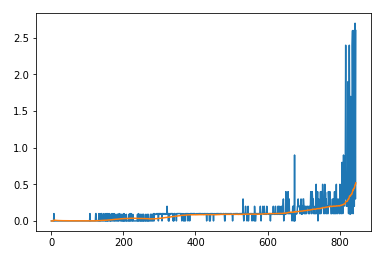


* Hyperparameters:

|  |  |
| --- | --- |
| 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



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.