Collaboration & Competition Project Report

Learning Algorithm

This model used a DDPG algorithm with the following parameters:

Replay buffer size: int(1e6)
Minibatch size: 256
Discount factor: 0.9
Tau for soft update of target parameters: 1e-3
Learning rate of the actor: 1e-3
Learning rate of the critic: 1e-3
L2 weight decay: 0

I used DDPG, a type of actor-critic algorithm, to solve this environment. In DDPG, both the actor and the critic have a target and a local network (for a total of 4 networks). The actor approximates the optimal policy deterministically by using the argmax of Q(s,a). The critic evaluates the policy and is updated using the TD-error, and the actor is trained using the deterministic policy gradient algorithm.

DDPG uses a replay buffer to prevent oscillation and divergence in the action values. The state (S), actions (A), rewards (R), and next states (S') are stored as tuples and small batches of the tuples are sampled to learn from. This is called experience replay and it helps break harmful correlations, plus the model can learn better from rare occurrences.

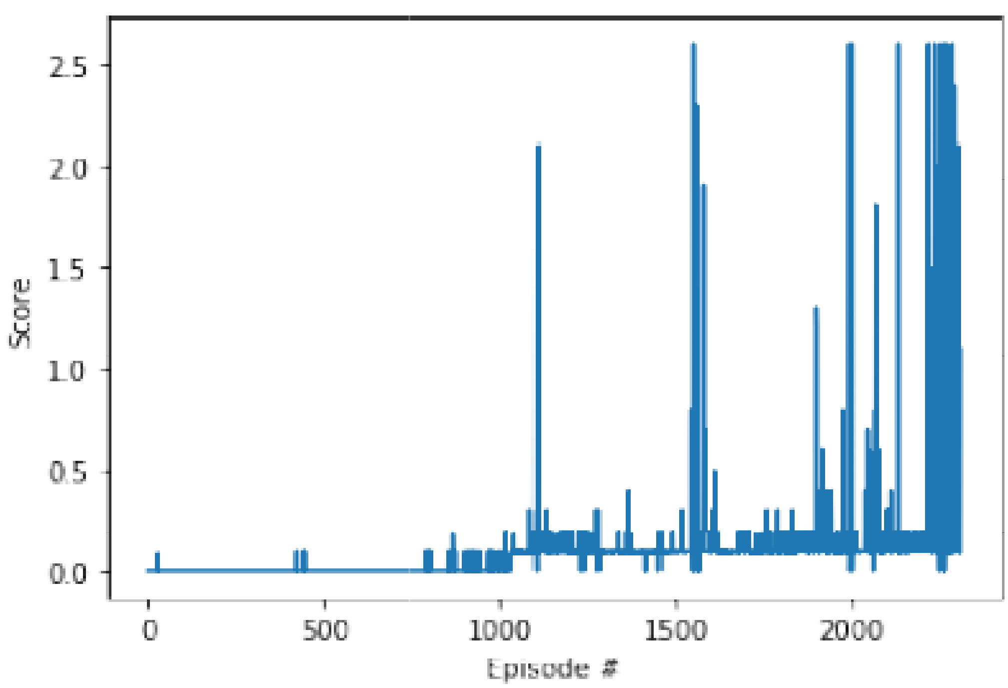
DDPG also uses soft updates to update the target networks. This means that target network is slowly blended with the regular network at every timestep. For instance, the target network will take 99.99% of the target network weights and only 0.01% of the regular network weights. This practice leads to faster convergence.

DDPG utilizes Ornstein-Uhlenbeck (O-U) noise for exploration. The algorithm probabilistically selects a random action like Boltzmann exploration or epsilon-greedy in order to ensure that it doesn't get stuck in the rut of just sticking with it's previous experiences.

The actor and critic each contained 2 hidden layers in each network, and I used batch normalization in the critic. In the actor, the hidden layers were of size 128 and 64 respectively and in the critic they were of size 400 and 300 respectively.

Plot of Rewards

Episode 100	Average Score: 0.001	Min Score: -0.010Max Score: 0.000
Episode 200	Average Score: 0.000	Min Score: -0.010Max Score: 0.000
Episode 300	Average Score: 0.000	Min Score: -0.010Max Score: 0.000
Episode 400	Average Score: 0.000	Min Score: -0.010Max Score: 0.000
Episode 500	Average Score: 0.006	Min Score: -0.010Max Score: 0.000
Episode 600	Average Score: 0.000	Min Score: -0.010Max Score: 0.000
Episode 700	Average Score: 0.000	Min Score: -0.010Max Score: 0.000
Episode 800	Average Score: 0.006	Min Score: -0.010Max Score: 0.100
Episode 900	Average Score: 0.010	Min Score: -0.010Max Score: 0.000
Episode 1000	Average Score: 0.015	Min Score: -0.010Max Score: 0.000
Episode 1100	Average Score: 0.079	Min Score: 0.000 Max Score: 0.0900
Episode 1200	Average Score: 0.152	Min Score: 0.000 Max Score: 0.0900
Episode 1300	Average Score: 0.106	Min Score: -0.010Max Score: 0.100
Episode 1400	Average Score: 0.103	Min Score: 0.000 Max Score: 0.0900
Episode 1500	Average Score: 0.097	Min Score: -0.010Max Score: 0.100
Episode 1600	Average Score: 0.230	Min Score: 0.090 Max Score: 0.2000
Episode 1700	Average Score: 0.115	Min Score: -0.010Max Score: 0.100
Episode 1800	Average Score: 0.109	Min Score: -0.010Max Score: 0.100
Episode 1900	Average Score: 0.153	Min Score: -0.010Max Score: 0.100
Episode 2000	Average Score: 0.173	Min Score: 0.100 Max Score: 0.1900
Episode 2100	Average Score: 0.175	Min Score: -0.010Max Score: 0.100
Episode 2200	Average Score: 0.169	Min Score: -0.010Max Score: 0.100
Episode 2300	Average Score: 0.478	Min Score: 0.090 Max Score: 0.2000
Episode 2309	Average Score: 0.502	Min Score: 0.990 Max Score: 1.1000
Environment solved in 2309 episodes!		Average Score: 0.502



Ideas for Future Work

I would be interested in comparing the performance of my current model that uses the DDPG algorithm to other models that use other algorithms, such as PPO or MADDPG.