Collaboration and Competition Report

I used Multi-Agent Deep Deterministic Policy Gradients (MADDPG) to solve this Tennis, in which the two agents need to learn play tennis and maximize their scores.

The environment has a state space of 8 dimensions and a 2-dimenional action space with continuous values. Since 3 observations are stacked together(8*3), the state input for the networks actually has a size of 24.

Learning Algorithm

The final agents are trained using MADDPG with 2 agents. MADDPG is an extension of DDPG. It contains multiple players to work in coordinating and competing environment.

Why MADDPG?

First, the environment has continuous action space. DDPG is a good choice that works well for continuous space. Second, the task involves multiple agents to cooperate. So separate agents without sharing cannot handle the non-stationary environment from individual agent's point of view. MADDPG can solve this problem.

Details about MADDPG

Networks: In this task, there are 2 DDPG agents. Each of them has their own local and target Actor and Critic networks. They have the same architecture for Actor and Critic respectively. The Actor network is for updating the policy parameters, while the Critic network updates the value function parameters and give a better suggestion to the Actor network about which direction to move.

Memory: In this tennis environment, as the agents are essentially learning the same policy, I let them share the same replay buffer for both the Actor and Critic networks. (In the original paper, the Actor only uses the local observation while the Critic shares full observations.)

Soft-update: The target networks for these agents are updated slowly using soft update towards the local networks to ensure stability in training.

Noise: As the policies are deterministic, I add OU noise in training to introduce exploration. When watch the agents play, I disable the noise.

Clipping: To make the agents are robust, I also add gradient clipping with a max norm of 1 in the critic network.

Some hyperparameters used:

- Experience replay buffer size: 1,000,000
- Training batch size: 128
- Reward discounting Gamma: 0.99
- Soft update Tau: 0.001
- Learning rate - Actor: 0.0001
- Learning rate - Critic: 0.001
- Weight decay for L2 regularization: 0
- Noise: Ornstein-Uhlenbeck process with mu=0, theta=0.15 and sigma=0.2

Both agents in MADDPG share the same network architecture, but train their weights separately.

Here are the details of the Actor and Critic networks.

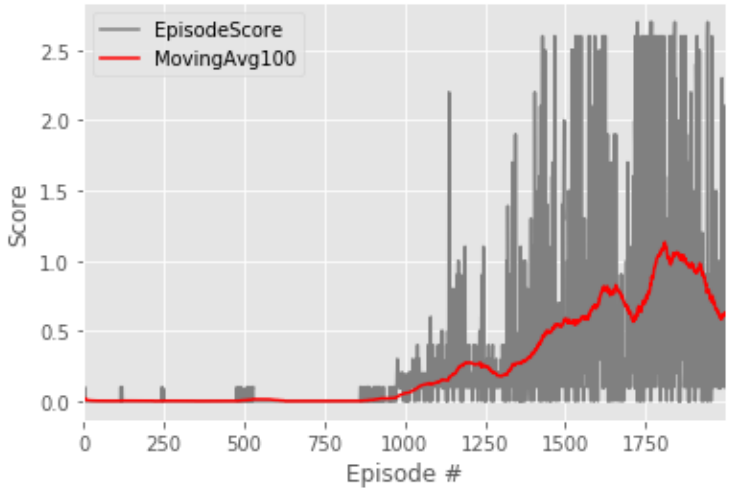
```
Actor(
  (fc1): Linear(in_features=33, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=4, bias=True)
)

Critic(
  (fcs1): Linear(in_features=33, out_features=256, bias=True)
  (fc2): Linear(in_features=260, out_features=256, bias=True)
  (fc3): Linear(in_features=256, out_features=128, bias=True)
  (fc4): Linear(in_features=128, out_features=1, bias=True)
)
```

Model Performance

The environment is considered solved when the average reward of the winner's reward over 100 episodes is 0.5+. The trained agents can solve it within 1,500 episodes with the architecture mentioned above. The best average 100-episode reward is about 1.06 around 1,800 episodes. Here are the details:

```
Episode 100 - Rolling Avg. Score (Max): 0.00
Episode 200 - Rolling Avg. Score (Max): 0.00
Episode 300 - Rolling Avg. Score (Max): 0.00
Episode 400 - Rolling Avg. Score (Max): 0.00
Episode 500 - Rolling Avg. Score (Max): 0.01
Episode 600 - Rolling Avg. Score (Max): 0.01
Episode 700 - Rolling Avg. Score (Max): 0.00
Episode 800 - Rolling Avg. Score (Max): 0.00
Episode 900 - Rolling Avg. Score (Max): 0.01
Episode 1000 - Rolling Avg. Score (Max): 0.06
Episode 1100 - Rolling Avg. Score (Max): 0.13
Episode 1200 - Rolling Avg. Score (Max): 0.27
Episode 1300 - Rolling Avg. Score (Max): 0.18
Episode 1400 - Rolling Avg. Score (Max): 0.35
Episode 1444 - Rolling Avg. Score (Max): 0.50
 * Environment first solved in 1444 episodes! Continue training...
Episode 1500 - Rolling Avg. Score (Max): 0.58
Episode 1600 - Rolling Avg. Score (Max): 0.68
Episode 1700 - Rolling Avg. Score (Max): 0.63
Episode 1800 - Rolling Avg. Score (Max): 1.06
Episode 1900 - Rolling Avg. Score (Max): 0.96
Episode 2000 - Rolling Avg. Score (Max): 0.62
```

Reward per Episode



Ideas for Future Work

For this work, we can make more improvements by:

- Policy Ensembles:
    - Instead of training only 1 policy, train M policies for each agent
    - Randomly pick one from M to generate the next action
    - Use the ensemble of the M policies to update the gradients
- Prioritized experience replay:
    - Same as DQN, when sample from the replay buffer, we may want to give different weights to different experiences. This should be able to help us use the past experiences more efficiently
- Batch normalization
- Update every x steps, instead of every single step