Project: Collaboration and Competition

Submission Results

Submission Date: March 18, 2020

✓ Submission Passed

Download Submission

Feedback Details

Specification Review Code Review

Reviewer Note

Congratulations

Great submission!

You have implemented and trained the agent successfully. All functions were implemented correctly and the **MADDPG (Multi-Agent Deep Deterministic Policy Gradient)** algorithm seems to work quite well.

Following posts give an insight into some other reinforcement learning algorithms that can be used to solve the environment

Proximal Policy Optimization by Open AI

Introduction to Various Reinforcement Learning Algorithms. Part II (TRPO, PPO)

Training Code

✓ Saved Model Weights

✓ Training code

Reviewer Note

You implemented a Multi Agent Deep Deterministic Policy Gradients algorithm, a very effective reinforcement learning algorithm. Your code was functional, well documented, and organized for training the agent.

Suggested reading:

- **Google Python Style Guide**
- **Python Best Practices**

Pros of the implementation

- Implementing the MADDPG algorithm is a good choice as it is found to work very well with multiple agents continuous action space.
- Correct Implementation of the Actor and Critic networks.
- Good use of replay memory to store and recall experience tuples.
- Using the target networks for Actor and Critic networks is a good choice.
- Good choice to update the target network using soft updates and using tau for it.

The repository includes functional, well-documented, and organized code for training the agent.

✓ Framework

README

✓ `README.md`

✓ Instructions

✓ Getting Started

✓ Project Details

Reviewer Note

Good work with environment details. The README described all the project environment details.

- **Environment**: How is it like?
- **Agent** and its **actions**: When is it considered resolved?; What are the possible actions the agent can take?
- **State space**: Is it continuous or discrete?
- **Reward Function**: How is the agent rewarded?
- **Task**: What is its task?; Is the task episodic or not?

The README describes the the project environment details (i.e., the state and action spaces, and when the environment is considered solved).

Report

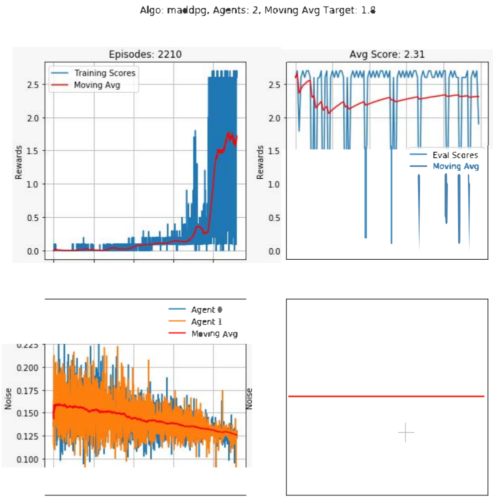
✓ Report

✓ Learning Algorithm

✓ Plot of Rewards

Reviewer Note

Nice work! The report informed the number of episodes needed to solve the environment and included a plot of rewards per episode.



A plot of rewards per episode is included to illustrate that the agents get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

The submission reports the number of episodes needed to solve the environment.

✓ Ideas for Future Work