Udacity

Deep Reinforcement Learning Nanodegree

<u>Project 3- Collaboration and Competition</u>

Algorithm

In this project, we use the DDPG algorithm (Deep Deterministic Policy Gradient) and the MADDPG algorithm, a wrapper for DDPG. MADDPG stands for Multi-Agent DDPG. DDPG is an algorithm which concurrently learns a Q-function and a policy. It uses off-policy data and the Bellman equation to learn the Q-function, and uses the Q-function to learn the policy. This dual mechanism is the actor-critic method. The DDPG algorithm uses two additional mechanisms: Replay Buffer and Soft Updates.

In MADDPG, we train two separate agents, and the agents need to collaborate (like don't let the ball hit the ground) and compete (like gather as many points as possible). Just doing a simple extension of single agent RL by independently training the two agents does not work very well because the agents are independently updating their policies as learning progresses. And this causes the environment to appear non-stationary from the viewpoint of any one agent. In MADDPG, each agent's critic is trained using the observations and actions from both agents , whereas each agent's actor is trained using just its own observations.

Goal

The task is episodic, and in order to solve the environment, your agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents). Specifically,

- After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores.
- This yields a single score for each episode.

The environment is considered solved, when the average (over 100 episodes) of those scores is at least +0.5.

Hyperparameters

From ddpg_agent.py

GAMMA = 0.99
TAU = 5e-2
LR_ACTOR = 5e-4
LR_CRITIC = 5e-4

WEIGHT_DECAY = 0.0
NOISE_AMPLIFICATION = 1
NOISE_AMPLIFICATION_DECAY = 1

From maddpg_agent.py
BUFFER_SIZE = int(1e6)
BATCH_SIZE = 512
LEARNING_PERIOD = 2

Neural Networks

In this prohect, there are 8 *neural networks*. For the training, we create one *maddpg agent*.

```
maddpg = maddpg_agent()
```

In turn, *maddpg agent* creates 2 _ddpg agents_:

```
self.agents = [ddpg_agent(state_size, action_size, i+1, random_seed=0)
```

Each of two agents (red and blue) create 4 neural networks:

```
self.actor_local = Actor(state_size, action_size).to(device)

self.actor_target = Actor(state_size, action_size).to(device)

self.critic_local = Critic(state_size, action_size).to(device)

self.critic_target = Critic(state_size, action_size).to(device)
```

Classes Actor and Critic are provided by **model.py**. The typical behavior of the actor

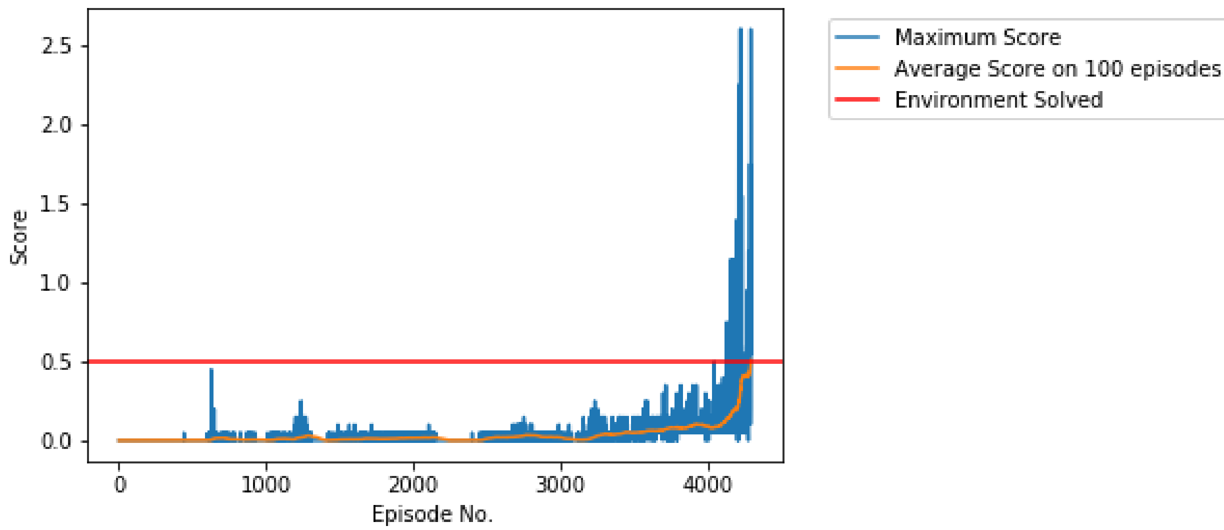
```
actor_target(state) -> next_actions

actor_local(states) -> actions_pred
```

Both the actor and critic classes implement the neural network with 3 fully-connected layers and 2 re ctified nonlinear layers.

Results of Training

While using Udacity Workspace with GPU enable, the desired average reward **+0.5** was achiev ed in **4291** episodes in **00h:45m:33s**. Complete output is too large and can be seen in the Ten nis Notebook. Below is just summary of the result

Plot



Ideas for Future Work

1. Tuning hyperparameters value for better result.
2. Continue training the model rather than breaking the loop when goal achieved initially.