MADDPG Collaboration and Competition project
Tennis

Jean-Baptiste Gheeraert

UDACITY

DEEP REINFORCEMENT LEARNING NANODEGREE
UDACITY

September 20, 2019

Table des matières

Chapitre 1

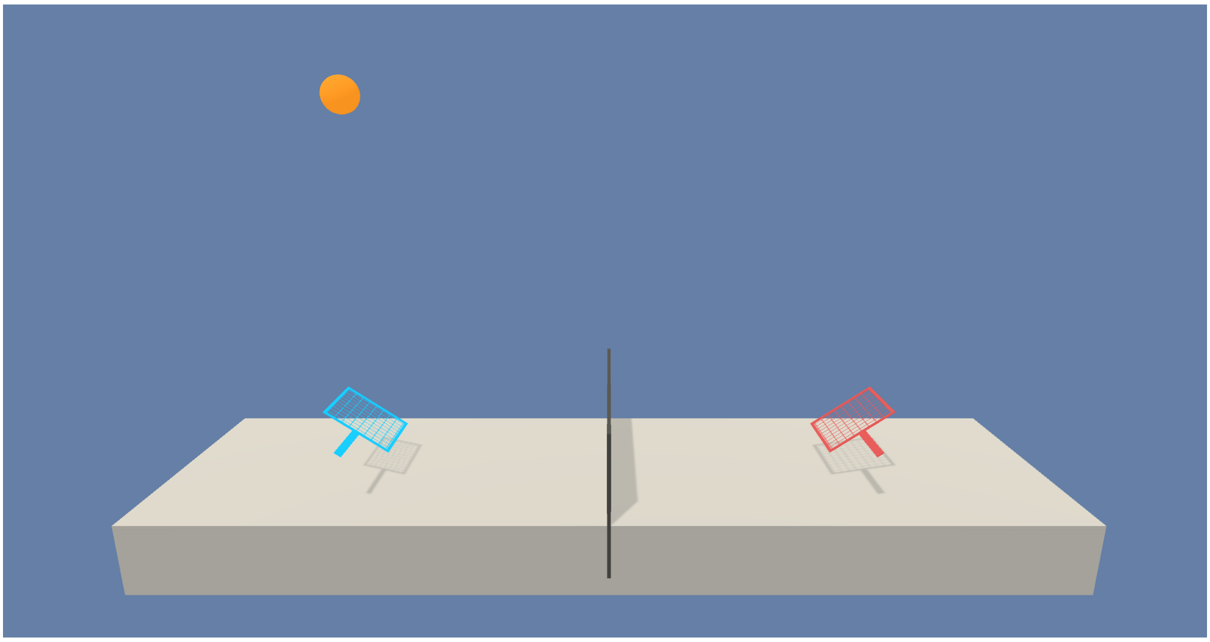
Project description

1.1 Environment

In the Tennis environment, two agents control rackets to bounce a ball over a net. If an agent hits the ball over the net, it receives a reward of +0.1. If an agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01. Thus, the goal of each agent is to keep the ball in play.

The observation space consists of 8 variables corresponding to the position and velocity of the ball and racket. Each agent receives its own, local observation. Two continuous actions are available, corresponding to movement toward (or away from) the net, and jumping.

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).



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

1.2 Learning algorithm

The algorithm used here is a Multi-Agents Deep Deterministic Policy Gradient (MADDPG) [1]. A MADDPG is composed of multiple DDPG agents.

During a step, the actors choose actions depending on their only observation. The critics however can use the whole state information and the actions of the other agents in order to better evaluate the optimal action value function.
This allows to better estimate future rewards as the critic learns to approximate the other agents' strategies.

The structure of each agents comes from the ddpg-pendulum project of the nanodegree. I add a MultiAgents class and modify the Agent class in order to take into account the share of information between agents.
The algorithm is very unstable and was hard to train. Some small modifications changed a lot in the final result.

For example I changes the noise and it helped a lot.
The architecture of the networks are as follow; the actor is composed of 3 fc units :

- First layer : input size = 24 and output size = 256
- Second layer : input size = 256 and output size = 128
- Third layer : input size = 128 and output size = 4

The critic is composed of 3 fc units :

- First layer : input size = 48 and output size = 256
- Second layer : input size = 260 and output size = 128
- Third layer : input size = 128 and output size = 1

The second layer takes as input the output of the first layer concatenated with the choosen actions.

The training hyperparameters are as follow :

- Buffer size : 100,000
- Batch size : 256
- γ : 0.99
- τ : 0.001
- learning rate actor : 0.0001
- learning rate critic : 0.0001
- weight decay : 0
- noise decay : 0.99

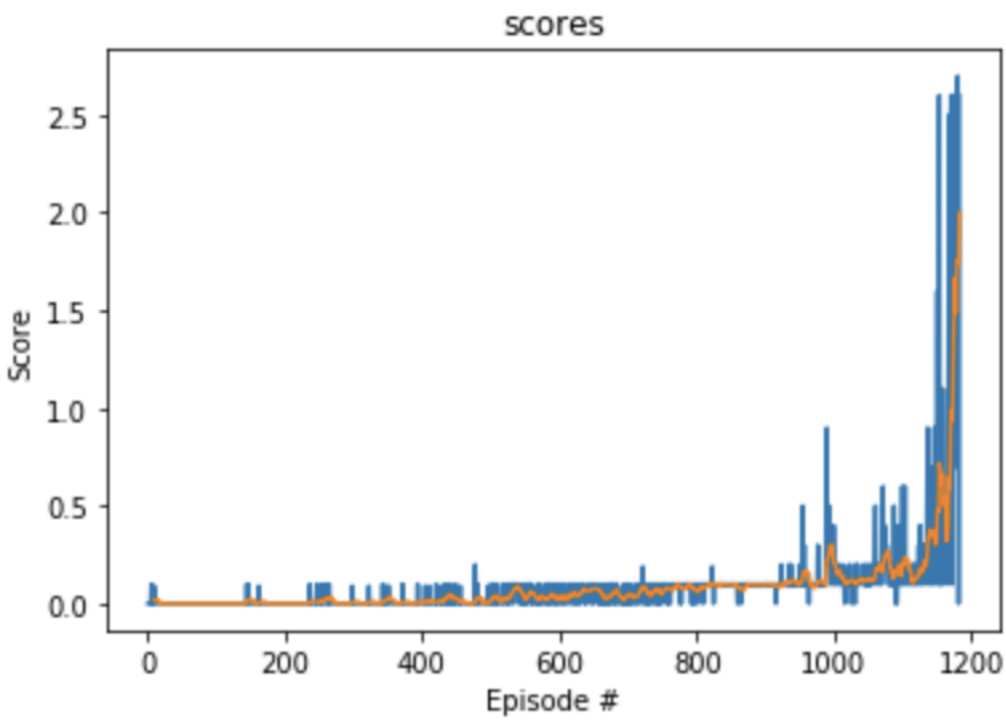
Chapitre 2

Plot of rewards

The environment has been solved in 1083 episodes.

```
Episode 100     Average Score: 0.00
Episode 200     Average Score: 0.00
Episode 300     Average Score: 0.01
Episode 400     Average Score: 0.01
Episode 500     Average Score: 0.02
Episode 600     Average Score: 0.04
Episode 700     Average Score: 0.05
Episode 800     Average Score: 0.07
Episode 900     Average Score: 0.09
Episode 1000    Average Score: 0.13
Episode 1100    Average Score: 0.15
Episode 1183    Average Score: 0.52
Environment solved in 1083 episodes!    Average Score: 0.52
```

Here is the graph of the score evolution :



Chapitre 3

Ideas of future works

Model such as AlphaZero [2] could also be used.

Bibliographie

[1] Ryan LOWE et al. "Multi-agent actor-critic for mixed cooperative-competitive environments". In : *Advances in Neural Information Processing Systems*. 2017, p. 6379-6390.

[2] David SILVER et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In : *arXiv preprint arXiv :1712.01815* (2017).