Using the Multi-Agent Deep Deterministic Policy Gradient Algorithm to Solve the Tennis Environment

Alessandro Leite

1 Context

An important concept in machine learning comprises the tasks of deciding from experience the sequence of action to perform in an uncertain environment to achieve some objective [1]. Inspired by behavioral psychology [2], reinforcement learning (RL) proposes a formal framework in which an agent learns by iterating with an environment. Based on experiences gathered by this iteration, the agent learns to optimize some objectives represented in the form of cumulative rewards. Thus, reinforcement learning focuses on the problem of learning a behavioral policy, a mapping from states to actions, which maximizes cumulative long-term rewards [2]. A policy can be represented as a lookup table, listing the appropriate action from any state. Therefore, in a rich environment, this approach is infeasible, and the policy must be encoded as a parametrized function.

Over the last years, RL has achieved some breakthroughs by integrating deep learning techniques. This combination, named deep reinforcement learning (DRL) is appropriate for problems with high dimensional spaces. As a result, deep reinforcement learning can learn different levels of abstractions from the data, which helps it to be successfully employed in complicated tasks with lower prior knowledge.

2 Goal

This project uses the Unity platform to train two agents to play tennis. In this case, 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 eight 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, the agents must get an average score of $+0.5$ over 100 consecutive episodes to considered the environment as solved.

3 Method

The proposed solution relies on Deep Deterministic Policy Gradient (DDPG) algorithm proposed by Lillicrap et al. [3] and on multi-agent deep deterministic policy gradient (MADDPG) [4] algorithm. DDPG is an off-policy actor-critic algorithm that uses deep function approximators to learn policies in high-dimensional continuous action spaces. It is based on the deterministic policy gradient algorithm (DGP) algorithm Silver et al.. DPG algorithm maintains an actor function $\mu(s|\theta^{\mu})$ to specify an policy that maps states to a specific action. Similar to Q-learning, the critic $Q(s, a)$ function relies on Bellman equation and the actor is

Table 1: Hyperparameters

buffer size	mini-batch	discount factor	LR actor	LR critic	soft updates
10^6	200	$\gamma = 0.99$	$\alpha_{\text{actor}} = 10^{-4}$	$\alpha_{\text{critic}} = 10^{-3}$	$\tau = 10^{-3}$

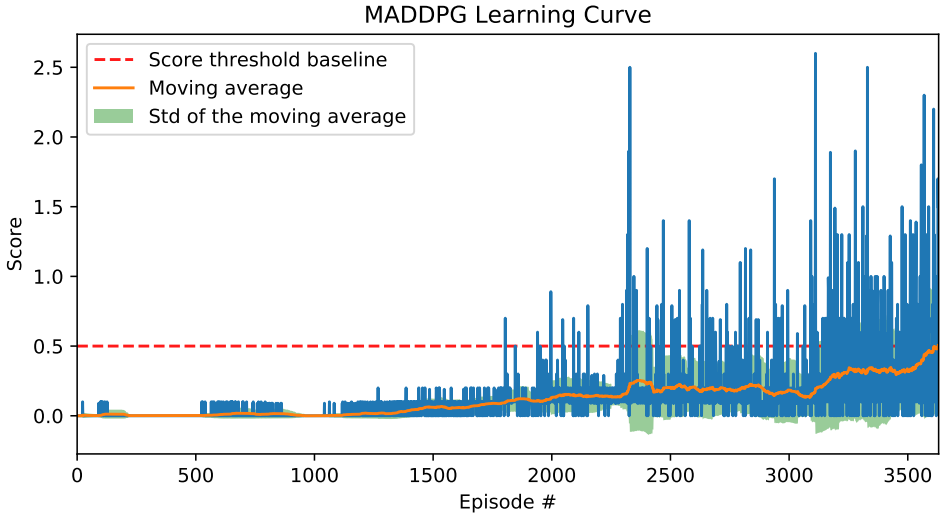

Figure 1: Learning curve of the MADDPG algorithms

updated by applying the chain rule to the expected return from the start distribution J with respect to the actor parameters [3]:

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_{\theta^\mu} Q\left(s, a|\theta^Q\right)\Big|_{s=s_t, a=\mu(s_t|\theta^\mu)}\right] \\ &= \mathbb{E}_{s_t \sim \rho^\beta}\left[\nabla_a Q\left(s, a|\theta^Q\right)\Big|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_\mu} \mu\left(s|\theta^\mu\right)\Big|_{s=s_t}\right]\end{aligned}$$

$$\mu'\left(s_t\right) = \mu\left(s_t|\theta_t^\mu\right) + \mathcal{N}$$

4 Results

The architecture is the same one proposed by Lillicrap et al.[3]. Table 1 illustrates the values of the parameters of the networks.

Figure 1 shows the performance of DDPG algorithm when solving the Reacher task. The algorithm required 360 episodes to solve the task.

For this kind of environment, the algorithms that employ a distributed approach to distribute the tasks usually lead to a better performance.

References

[1] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Foundations and Trends in Machine Learning*, vol. 11, no. 3–4, pp. 219–354, 2018.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations*, Y. Bengio and Y. LeCun, Eds., 2016.

[4] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in neural information processing systems*, 2017, pp. 6379–6390.

[5] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *31st International Conference on Machine Learning*, E. P. Xing and T. Jebara, Eds., vol. 32, no. 1, 2014, pp. 387–395.