DDPG solving unity environments

Abdelkarim Eljandoubi

September 25, 2022

1 Introduction

In this project, I trained a DDPG agent to solve two types of environment.

1.1 Reacher environment

A double-jointed arm can move to target locations. A reward of +0.1 is provided for each step that the agent's hand is in the goal location. Thus, the goal of your agent is to maintain its position at the target location for as many time steps as possible.

The observation space consists of 33 variables corresponding to position, rotation, velocity, and angular velocities of the arm. Each action is a vector with four numbers, corresponding to torque applicable to two joints. Every entry in the action vector should be a number between -1 and 1.

1.2 Crawler environment

In this continuous control environment, the goal is to teach a creature with four legs to walk forward without falling.

1.3 Solveing

An environment is considered solved, when an average score of +30 over 100 consecutive episodes, and over all agents is obtained.

2 Implementations

I have started from the code provided in Udacity as a solution for the coding exercise which implements the basic form of DDPG [2] for the agent.

2.1 Trainer

I adapt the trainer to muti-agent environments such that each agent adds its experience to a replay buffer that is shared by all agents.

2.2 Architectures

I used here the same architectures from the coding exercise with small changes. In fact, I added a batch normalization layer before the first fully connected hidden layer on both actor and critic as suggested in DDQN.

3 Results

Throughout experiences, I set

- replay buffer size to 10^6
- batch size to 128

- the target update frequency to 20
- number of optimization step to 10
- the actor learning rate to 10^{-4}
- the critic learning rate to 10^{-3}
- $\gamma = 0.99$
- $\tau = 10^{-3}$

Also, I trained the agent by the Adam optimizer because of its superior experimentally performance over vanilla.

3.1 Reacher

The agent succeeded to solve this environment with only 135 episodes (see fig. 1).

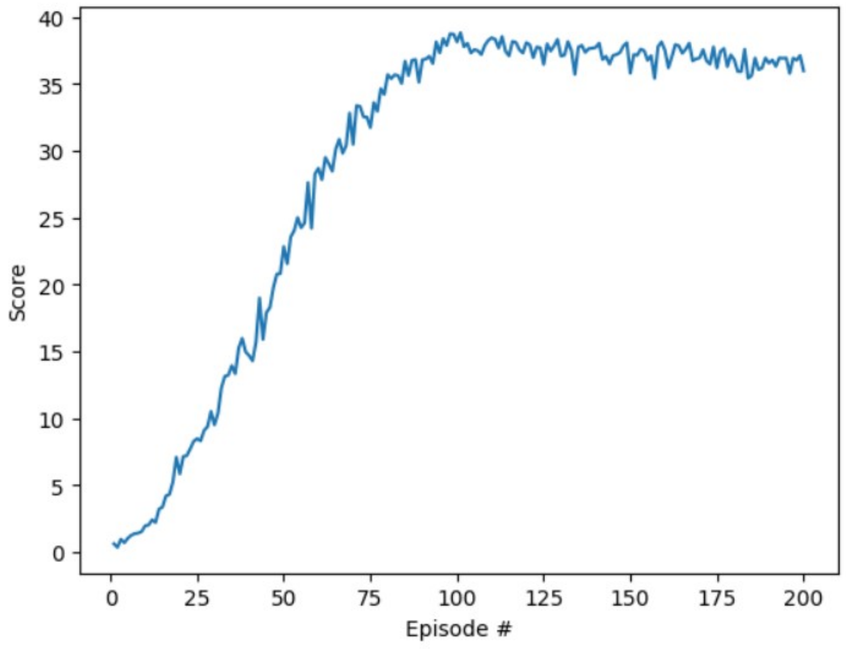

Figure 1: The score plot for Reacher.

3.2 Crawler

Here, the agent achieves the goal in the end, but with high deviations. (see fig. 2).

4 Future work

In addition to the current work, we can do the following to improve performance by :

- adding prioritized experience replay from [3].
- implementing D4PG from [1]

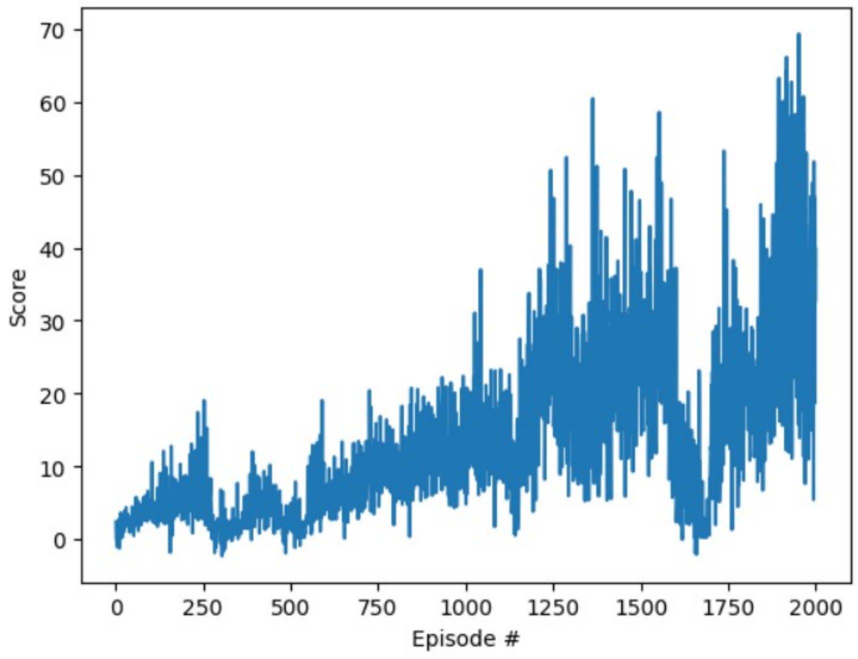

Figure 2: The score plot Crawler.

References

[1] Gabriel Barth-Maron, Matthew W. Hoffman, David Budden, Will Dabney, Dan Horgan, Dhruva TB, Alistair Muldal, Nicolas Heess, and Timothy P. Lillicrap. Distributed distributional deterministic policy gradients. *CoRR*, abs/1804.08617, 2018.

[2] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[3] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2015. cite arxiv:1511.05952Comment: Published at ICLR 2016.