TD3 - Deep Reinforcement Learning

Harold Guéneau

2021

1 Introduction

This document is a rapid introduction to explain the intuition behind the Twin-Delayed Deep Deterministic Policy Gradient (TD3) algorithm, which is based on Deep Q-Learning and Actor-Critic methods. The TD3 agent can operate in continuous actions space.

2 Temporal-Difference Learning

Deep Q-Learning is based on Temporal-Difference Learning (TD-Learning), it is an algorithm that aim to approximate the optimal action-value function:

$$Q*(s,a) = \max_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi] \tag{1}$$

With $\pi(a|s)$ the policy.

To do that, TD-Learning will use experiences to update iteratively the Q-values, the following rule to update the Q-values is called SARSA:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \tag{2}$$

The $Q(s_{t+1}, a_{t+1})$ is chosen using the policy ($\epsilon - greedy$ policy for instance). We can also consider the expectation of the Q-value for the next state ($E[Q(s_{t+1}, a_{t+1})]$, Expected Sarsa), which is also on-policy learning, since we compute the expectation based on the policy. Another possibility is to consider the action with the maximum Q-value in the next state ($max_a[Q(s_{t+1}, a^i_{t+1}])$, which is an off-policy approach (SarsaMax/Q-Learning).

The TD-Learning is a very suitable algorithm for a large set of problems, however, it becomes harder to find the optimal action-value function when the action's dimension or the environment's dimension become continuous. Some solutions exist to tackle that, such tile-coding, but is not the most efficient way to approximate the optimal action-value function. Therefore, we will explore the use of deep-learning in this project.

3 Deep Q-Learning

Deep Q-Learning was first introduced by DeepMind on Atari game environments where the results were promising [3]. The idea is to consider a neural network that will approximate the Q-value for every given states and actions (even continuous ones). It is based on the following relation:

$$\begin{aligned} Q*(s_t, a_t) &= \max_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|\pi] \\ &= r_t + \gamma Q*(s_{t+1}, s_{t+1}|\pi) \end{aligned} \tag{3}$$

The aim of the algorithm then is to optimise the neural network's parameters Ω to minimize the following loss (mean-squared error):

$$Loss(\Omega) = E_{(s_t,a_t,r_t,s_{t+1})}[(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}|\Omega) - Q(s_t, a_t|\Omega))^2] \tag{4}$$

By minimizing this loss, the neural network will approximate the optimal action-value function, and deduce the optimal policy from it. In order to stabilize the learning process, it is common to use Replay Memory and two neural networks (Policy & Target Networks).

4 Deep Deterministic Policy Gradient (DDPG)

The DDPG agent will use a replay buffer as in Deep Q-Learning, to train its neural networks in an off-policy manner (with a target network to stabilize the training). The main difference is that DDPG has also a neural networks to approximate the optimal action (policy network). Thus, DDPG is able to approximate continuous actions. [2] This is then an actor-critic architecture, where the actor deduce the optimal policy and the critic evaluate the chosen actions.

The training process will begin as in Deep Q-Learning, where the agent will interact with the environment in order to store experiences in memory (for the off-policy training). Then, when the memory is long enough, the agent can sample a mini-batch from memory and train a Q-function (a neural network).

Instead of using the argmax in the learning process, the DDPG will use the best action based on the policy network.

Please find below the critic loss (Q-Network) and the actor loss (Policy-Network):

$$CriticLoss(\Omega) = E_{(s_t,a_t,r_t,s_{t+1})}[(r_t + \gamma Q(s_{t+1}, \mu(s_{t+1};\phi_i^-)|\Omega^-) - Q(s_t, a_t|\Omega))^2] \tag{5}$$

$$ActorLoss(\Omega) = E_{(s_t,a_t,r_t,s_{t+1})}[Q(s_{t+1}, \mu(s_{t+1};\phi_i)|\Omega)] \tag{6}$$

5 Twin- Delayed Deep Deterministic Policy Gradient (TD3)

As the DDPG is a very volatile algorithm, it has been decided to use a TD3 Agent, an extension of the DDPG one [1].

The idea is to extend it in three ways:

- It use two separate networks for the critics.
- It adds noise in the target action (and not only in the environment).
- It delays the update of the policy network, so that the twin networks (critic networks) update more frequently.

References

[1] Fujimoto et al. Addressing function approximation error in actor-critic methods. *International Conference on MachineLearning, Stockholm, Sweden, PMLR 80, 2018.*, 2018.

[2] Lillicrap et al. Continuous control with deep reinforcementlearning. *Conference paper at ICLR*, 2016.

[3] Mnih et al. Human-level control through deep reinforcementlearning. *Nature*, 2015.