Udacity Multiagent Project

Thomas Lecat

April 2021

1 Environment

The environment is described as Markov Decision Process (S, A, R, p, γ) with unknown dynamics. The goal is to approximate the optimal policy π^* that maximizes the upcoming cumulative reward in every state.
The environment represents two agents playing tennis. The action space A of each agent is continuous and consists in 2 actions in the $[-1, 1]$ range.

2 Agent

2.1 Multi-agent DDPG

A modified version of the DDPG agent is implemented as the base solution. We follow the paradigm of centralized training and decentralized execution. DDPG uses two distinct neural networks:

A "critic" network parameterized by θ^Q seek to approximate the optimal Q value Q^*, defined as:

$$Q^*(s, a, a_{other}) = \max_{\pi} \mathbb{E}_{\pi}(\sum_t \gamma^t r_t | S = s, A_0^{a_0} = a, A_0^{a_1} = a_{other}), \forall (a, a_{other}, s) \in A^2 \times S$$

where $A_t^{a_0}$ and $A_t^{a_1}$ represent the actions of the first and second agents respectively at step t. The network takes the current observation and action of the agent as well as the action of the other agent as input. It outputs a single value.

A deterministic "actor" network parameterized by θ^μ that approximates the function $\arg\max_a Q(\cdot|s)$. It takes the current observation as input. The weights of the actor are shared between agents.

The loss function of the critic is defined as the mean squared TD error of the Bellman optimality equation:

$$\mathcal{L}_{critic}(\theta^Q) = \mathbb{E}_{s,a,a_{other},r,s'}\left[((r + \gamma Q(s', \mu'(s'|\theta^{\mu'}), a_{other}|\theta^{Q'}) - Q(s, a, a_{other}|\theta^Q))^2\right]$$

where $\theta^{Q'}$ and $\theta^{\mu'}$ are the parameters of target networks which are slowly updated towards their respective local networks at every training step. The target networks makes the TD target $r+\gamma Q(s', \mu'(s'|\theta^{\mu'}), a_{other}|\theta^{Q'})$ independent from the parameters θ^{Q}, which stabilizes the training.

The loss function of the actor strives to maximise the Q value:

$$\mathcal{L}_{actor}(\theta^{\mu}) = -\mathbb{E}_{s,a_{other}} Q(s, \mu(s|\theta^{\mu}), a_{other}|\theta^{Q})$$

The losses are minimized by Adam optimizers, a variant of the Stochastic Gradient Descent algorithm. At each step, the gradient is approximated from a minibatch of experiences $(s, a, a_{other}, r, s', d)_i$. To avoid a strong correlation between the steps of the minibatch, transitions are stored in a large replay buffer during rollouts and then sampled uniformly.

3 Hyperparemeters

The actor network is composed of 2 fully connected layers, with respectively 256 and 128 neurons. The 3 last local observations of the agent are stacked and form the input of the network. The actor optimizer is Adam with a learning rate of 10^{-3}.
The critic network is composed of 2 fully connected layers, with respectively 256 and 128 neurons. The output of the first layer and the action are concatenated before flowing into the second layer. The critic optimizer is Adam with a learning rate of 10^{-3}.

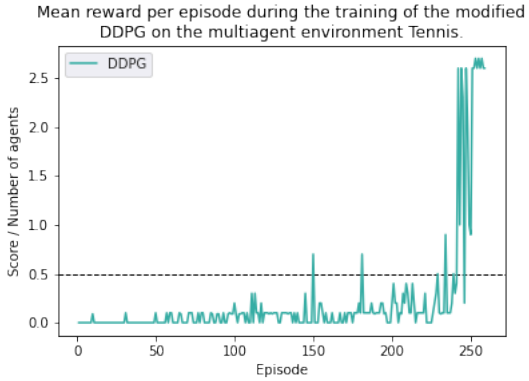
To increase exploration, random noise created by an Ornstein-Uhlenbeck process is added to the actions during rollouts.

The table below summarizes the main hyperparameters:

Hyperparameters		
actor learning rate	α^{μ}	1e-03
critic learning rate	α^{Q}	1e-03
discount factor	γ	0.99
target networks update coefficient	τ	0.006
buffer size		1,000,000
batch size		128
critic weight decay		0
Update frequency (number of steps)		1
Number of Adam steps per update		1

4 Results

We plot the mean reward per agent per episode during training:



The environment is solves in less than 400 iterations.

5 Next steps

As a next step, other multi-agent RL algorithms such as Q-Mix could be tested on this problem and compared to the current implementation.
Hyperparemeters could also be further tuned using strategies such as HyperOpt or Population Based Training.