Collaboration and Competition – Tennis

Anush Manukyan

1. Project description

Given two agents that control rockets to bounce a ball over a net. The goal is that the agents must bounce ball between one another while not dropping or sending ball out of bounds.
The agent receives a reward of +0.1 if it hits the ball over the net. It receives a reward of −0.1 if the ball hits the ground or sends out of bounds.
The observation space consists of 8 variables: the position of the ball and racket and the velocity of the ball and rocket. There are two continuous actions are available for each agent: (1) movement toward or away from the net, (2) jumping.

The environment is considered solved, when the agents gets a reward of in average at least +0.5, over 100 episodes.

2. Approach

Similar to previous project, here as well the task is to solve a continuous control problem. To solve such problem we can model it as a Markov decision process (MDP), which is defined using a $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{r}, \boldsymbol{\rho}_0, \boldsymbol{\gamma})$ tuple, where $\mathbf{S}$ is a state space, $\mathbf{A}$ is an action space, P is the transition probability distribution being at a state $\mathbf{s} \in \mathbf{S}$ and taking an action $\mathbf{a} \in \mathbf{A}$ to transient to a new state $\mathbf{s}' \in \mathbf{S}$, $\mathbf{r}$ is the reward function, which maps the state–action–pair to the set of real numbers. $\boldsymbol{\rho}_0$ is the distribution of the initial state $\mathbf{s}_0$ and $\boldsymbol{\gamma} \in (\mathbf{0}, \mathbf{1})$ is the discount factor.

By interacting with its environment, the agent learns to take the correct sequence of actions which maximizes the cumulative reward.

In this project we deal with multiagent collaborative task in continuous space action environment. As it is described above, the goal of the project is to keep the ball in the game without dropping or sending ball out of bounds.

To solve such problem **DDPG** algorithm has been used. Where both agents use the same network structure.

2.1 Deep Deterministic Policy Gradient (DDPG) Algorithm

DDPG is a model-free, policy-based algorithm, that adopts an actor-critic architecture, where the actor learns the policy, and the critic evaluates the learnt policy. As a function approximator two neural networks are used, one for policy approximation and another for value function approximation. In order to use the hardware efficiently the learning is performed in mini-batches. Hence, a reply buffer is used to store the experience tuples, such as (s_t, a_t, r_t, s_{t+1}), while interacting with the environment.

In order to solve issues with unstable learning, a few techniques are introduced: "soft" target update, which is used to slowly update the target values instead of directly copying the weights. Batch normalization, which normalizes each dimension across the samples in a minibatch to have unit mean and variance. And finally, a noise, introduced by Ornstein-Uhlenbeck process, is used in order to solve the issue with rarely performing exploration of actions. Therefore, noise is added on the space or action space parameter.

2.2 Hyperparameters

Replay buffers size: 1e5
Minibatch size: 1024
Discount factor: 0.99
Learning rate for actor: 1e-4
Learning rate for critic: 1e-3
Soft update of target parameters: 1e-3

2.3 Network Structure

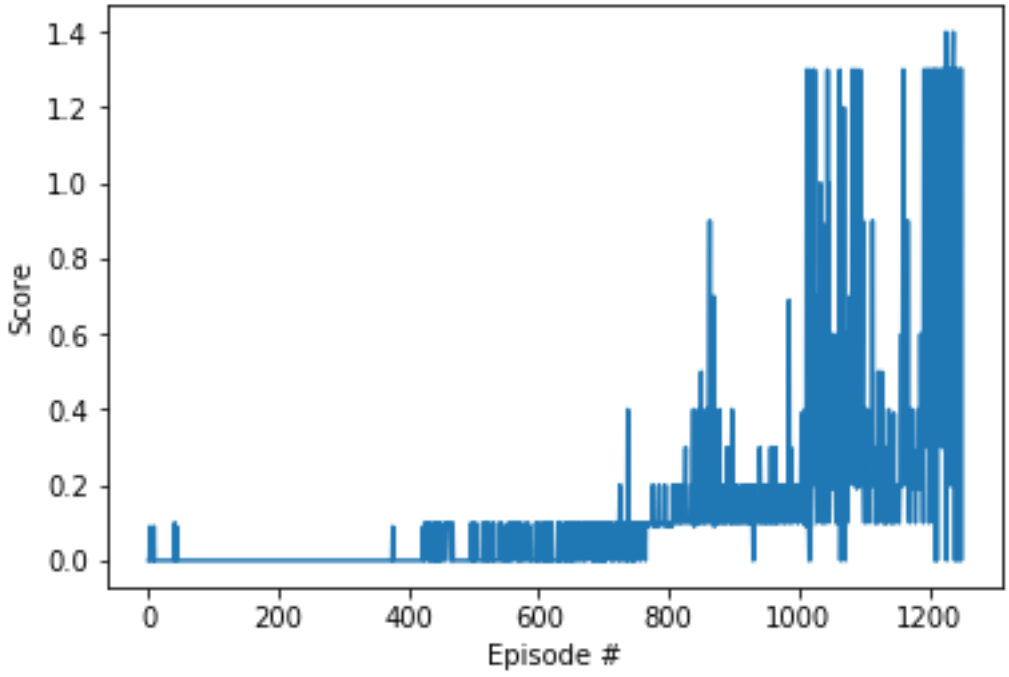
The network structure is kept the same as it was during project 2.

The actor network consists of one layer with 256 units. Here the input layer is directly mapped to the output layer. The input layer has a ReLu and the output layer a tanh activation function.

The critic network consists of three fully connected layers with 256, 256 and 128 units with ReLu activation functions.

3. Results

The environment has been solved at about 1200 episode, with an average score of +0.5.



4. Future improvements

DDPG proved to achieve good results for performing a collaborative task in a continuous control environment. However, after checking the state of the art, we can see that Proximal Policy Optimization (PPO) outperforms DDPG in various tasks. Hence, one of the improvements could be using PPO for training the agents.

Another possibility for future improvements can be the optimization of the hyperparameters, since for stable performance hyperparameters play an important role.

References

Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." *arXiv preprint arXiv:1509.02971* (2015).

Schulman, John, et al. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).

Mahmood, A. Rupam, et al. "Benchmarking Reinforcement Learning Algorithms on Real-World Robots." *arXiv preprint arXiv:1809.07731* (2018).