Project 3: Collaboration and Competition
Udacity Deep Reinforcement Learning

Bastian Broecker

Abstract. *This document shows the reinforcement learning approach applied in the third project "Collaboration and Competition" (Multi-Agent Learning) as part of the udacity nanodegree: Deep Reinforcement Learning. I decided to implement a multi-agent variant of DDPG called MADDPG. This method utilizes a centralized critic and a decentralized actor to overcome the dynamic nature of a multi-agent system. I compare it's performance with a vanilla DDPG.*

1. Project Environment and Goal Description

In this environment two agents control a tennis racket each with the goal 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 drops the ball -0.01 are deduced. Thus, the goal is to keep the ball in play.

The observation space consists of 8 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 have to average a score of $+0.5$ (over 100 consecutive episodes).

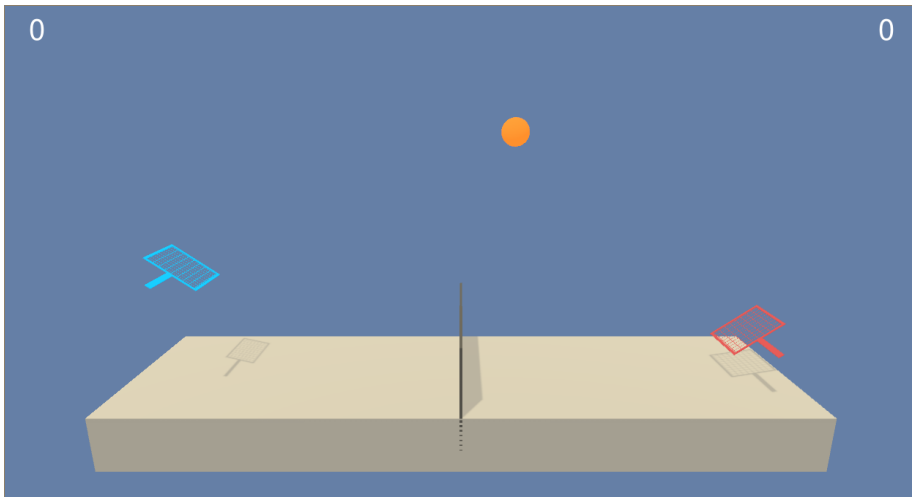

Figure 1. Environment

2. MADDPG

Multi-agent deep deterministic policy gradient (MADDPG) [Lowe et al. 2017] is a multi agent variantion of DDPG [Lillicrap et al. 2016], which I already used and described in the previous project. The problem of multi-agent settings is that their environments are dynamic, this means the next state is not only determined by an action of a single agent, it depends on the action of all agents, this makes estimation of future reward difficult, which can lead to a unstable training.

MADDPG uses a decentralized actor in combination with a centralized critic. This means, the actor only applies the observations of the own agent to determine the action, the critic is using the information of all agents and their policy to update its Q-estimation network.

In a setting of N agent the centralized action-value Q_i^μ of each agent i is update by:

$$L(\theta_i) = \mathbb{E}_{x,a,r,x'}[(Q_i^\mu(x, a_1, ..., a_N) - y)^2], \tag{1}$$

$$y = r_i + \gamma Q_i^{\mu'}(x', a'_1, ..., a'_N)|_{a'_j=\mu'_j(o_j)} \tag{2}$$

where μ_i is the policy of agent i, μ'_i the target policy, $x = (o_1, ..., o_N)$ the combined observations/states of all agents, r_i the reward of i and $a_1, ...a_N$ the actions of all agents.

The primary motivation behind MADDPG is that, if we know the actions of all agents the environment can be seen as a stationary environment.

2.1. Hyperparameters and Network

I compare a vanilla DDPG with decentralized critic to a MADDPG with a centralized critic. Both implementations have separate actor and an critic networks, for each agent, two hidden layers with 128 nodes each. The hidden layers have a ReLu activation function, where the actor has tanh at a output activation the critic has a linear output activation. Both variations use the following hyper parameters.

- Actor Learning Rate: $1e^{-3}$
- Critic Learning Rate: $1e^{-3}$
- Batch Size: 128
- Buffer Size: 10000
- γ: 0.95 (reduction of future rewards)
- Update every steps: 1
- Number of batches per step: 3

Both approaches converge quite stable and the DDPG with the decentralized critic converges even a little bit quicker to the required average of 0.5 (over 100 episodes), see Figure 2. The MADDPG's critic uses the states and action of all agents, therefore it has a larger input-space than the DDPG, which could be one explanation for the slower convergence. Since the DDPG's training is also quite stable, one might suggest that the dynamics of the environment are not so drastic and the agents are not strongly depending on the others agents action.

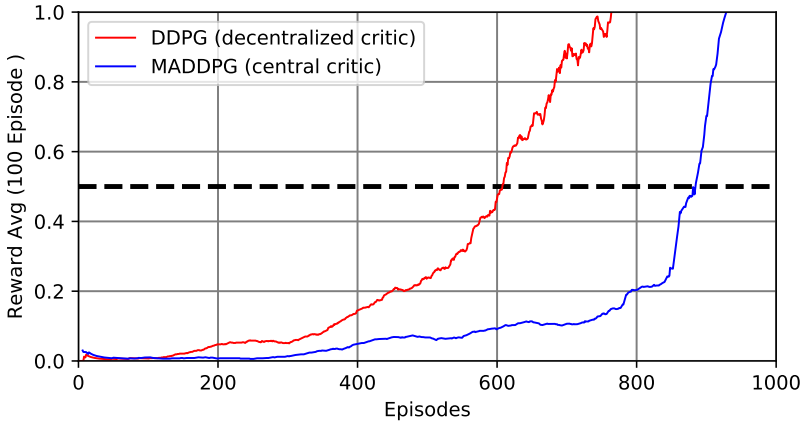

Figure 2. DDPG and MADDPG

3. Ideas for Future Work

In order to improve the performance of the performance of MADDPG. I would like to try automatized hyper-parameter tuning, for example Bayesian parameter optimization and see how much this can help finding the best settings, network sizes and network activation-functions. Further more, I would like to see if extensions like, prioritized experience buffer and n-step bootstrapping could improve the training performance, especially for other environments.

References

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y., editors, *ICLR*.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6382–6393, Red Hook, NY, USA. Curran Associates Inc.