Project3: Collaboration and Competition - MADDPG for Tennis environment

UDACITY

Contents

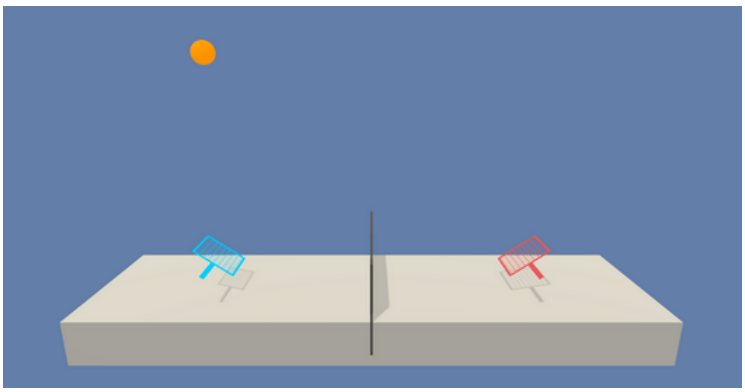

Figure 1: Udacity Tennis Environment

1 Environment description

The environment is called Tennis. The goal is to train two agents, that should bounce a ball over a net using rackts. A screenshot of the environment is shown in figure 1.

From an agents perspective, the environment behaves non-stationary. Most of the algorithms learned up to now, may not converge or show instability. Therefore, dedicated algorithms to solve multi-agent environments are needed.

Statespace and dynamics

The observation space of each agent has 8 dimensions, $\mathcal{O}_\rangle \subset \mathbb{R}^8$ and contains the kinematic information of the ball and the racket:

$$O_i = [x_{ball}, y_{ball}, u_{ball}, v_{ball}, x_{racket_i}, y_{racket,i}, u_{racket,i}, v_{racket,i}]. \tag{1}$$

The state of the environment is 12 dimensional, $S_i = O_1 \cup O_2 \subset \mathbb{R}^{12}$. From a single angents perspective, the environment is not fully observable. The environment dynamics are unknown, thus model-free reinforcement learning shall be used.

Actionspace

Every agent has two continous actions, corresponding to moving toward or away from net in $[-1, 1]$ and jumping $[-1, 1]$.

$$A_i = [-1, 1] \times [-1, 1] \tag{2}$$

Reward structure

The reward structure is an agent receives a reward of 0.1 for a hit and a reward of -0.01 for a miss. Thus each agent wants to keep the ball at play.

Environment is considered solved, when the agent receives an average score of 0.5 (keeps the ball at play for at least 5 times).

2 Multi-agent DDPG algorithm description

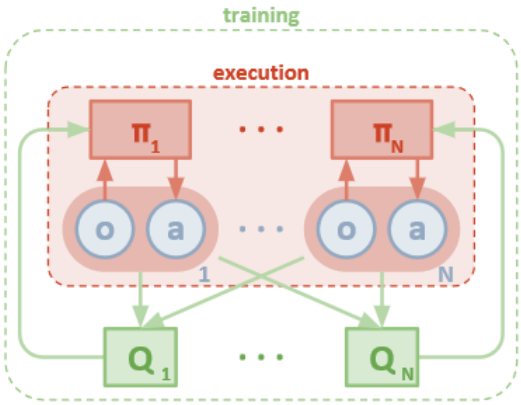

Figure 2: Concept of decentralized actor, centralized critic approach

It has been observed that single agent policy-gradient or value based approaches often perform poorly in multi-agent environments.
Thererore in[Low+20] the idea was proposed to centralize training, but decentralize execution - see figure 2. This means that one policy per agent $\pi_i(o_i)$ is learned, only takeing into account the agents observation. But for every agent a centralized action-value function

$$Q_i^{\pi}(x, a1, ...a_N) \tag{3}$$

is learned, where by x is some state information like $x = (o_1, ...o_N)$. As every agent can learn an individual action-value function, different reward structures per agent are allowed. This permits to train for collaboration or competition.
To train the agents critic a loss function similar to the DDPG approach can be used, whereby it is assumed that the policy of the other agents are known:

$$L(\theta, \mathcal{D}) = \sum_i \left(y_i - Q_\theta(x_i, a_1, ..., a_N)\right)^2, y_i \quad = r_i + \gamma Q_\theta(x_i', a_1', ...a_N')|_{a_n'=\pi_n(o_{n,i})} \tag{4}$$

A primary motivation behind MADDPG is that, if we know the actions taken by all agents, the environment is stationary even as the policies change [Low+20]. To provide exploration a noise term is added to the behaviour policy

$$a_t = \pi_\phi(o_t) + \mathcal{N}_t. \tag{5}$$

To generate the noise $\mathcal{N}_t$, gaussian noise was used. The following tricks were introduced to make the training of the algorithm more stable.

- Target Q-network (see project1 for details)
- Replay buffer (see project1 for details)
- Gradient clippling of critic network update.

3 Implementation details

As a baseline implementation the code from the Udacity "Physical deception" coding exercise was used. Some adjustments were made to cope with the Tennis environment. The action-value function approximation or critic network architecture is shown in figure 4. The policy network architecutre is shown in figure 3. The actor and critic network are identical for both agents.

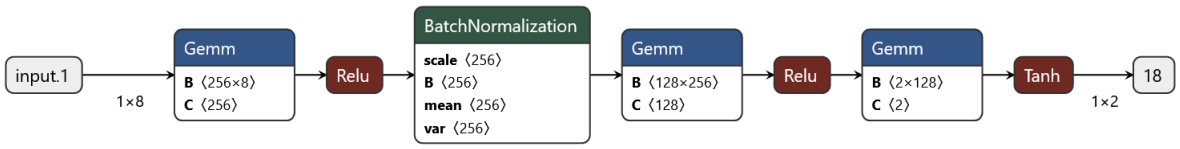

Figure 3: Architecture of the actor-network $\pi_i(o_i)$. Visualized with: netron.app.

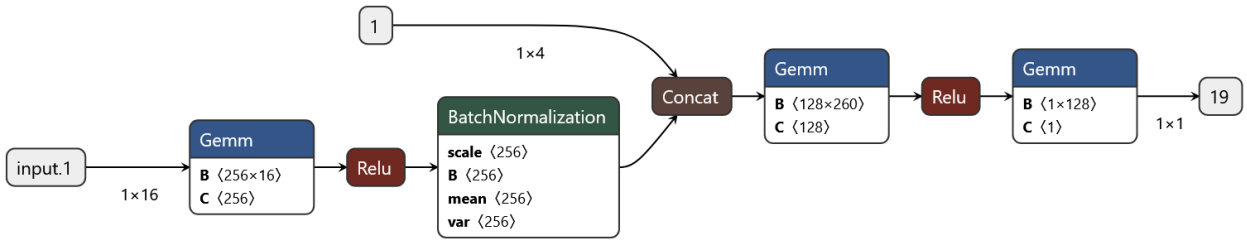

Figure 4: Architecture of the agents critic-network $Q_i^\pi((o_1, o_2), a_1, a_2)$. Visualized with: netron.app.

The following hyper-parameters were used for training:

```
BUFFER_SIZE = int(1e6)  # replay buffer size
BATCH_SIZE = 256          # minibatch size
GAMMA = 0.996               # discount factor
TAU = 1e-3                      # for soft update of target parameters
UPDATE_EVERY = 3        # udpate every n episodes
NB_LEARN = 4                # number of gradient descrent steps per update
LR_ACTOR = 1.1e-4       # learning rate of the actor network
LR_CRITIC = 3.3e-4       # learning rate of the critic

NOISE_DECAY = 0.99    # decay noise exponentially over episodes
BEGIN_TRAINING_AT = 500  # wait with adding noise
NOISE_START = 1.0  # maximum noise power
NOISE_END = 0.1     # minimum noise power
```

4 Training and Result

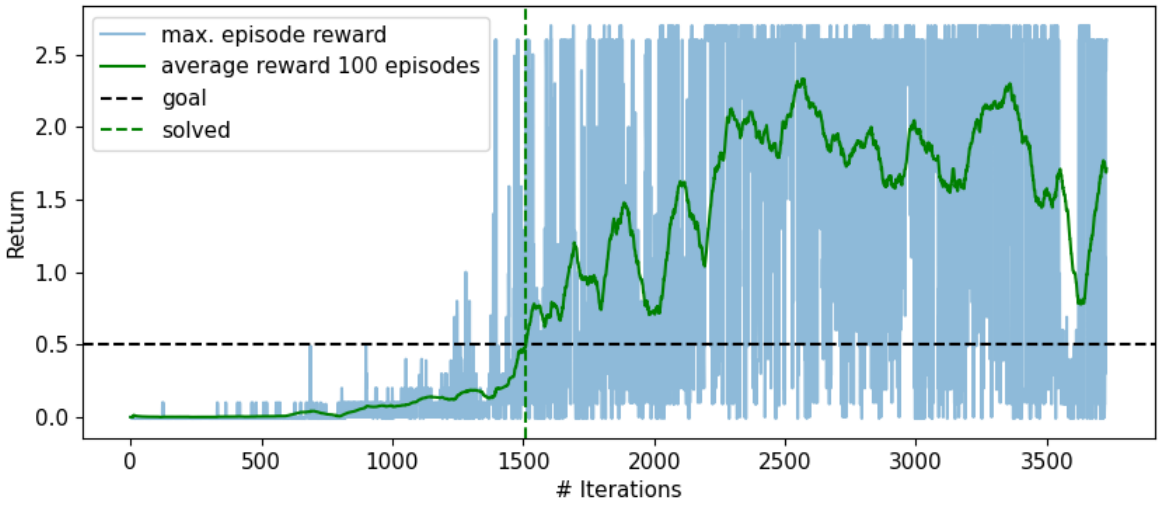

Figure 5: Progress of training

The training progress can be seen in figure 4. The environment was solved in 1511 episodes. The maximum return achived was 2.33. To me it is unclear, why never a episode return above 2.6 was achived. Maybe the episode length is somehow limited?

5 Summary and ideas for future work

The training of an agent was difficult, but could be achived using the MADDPG baseline algorithm provided in the coding exercise.

To improve performance, it would be interesting to further tune the hyperparameters (especially the noise parameters). In general it would be adviseable to shift training to a GPU, as training on my own CPU is slow.
Also it would be interesting to implement different DDPG improvements, to see if they also work for MADDPG, like state space noise for exploration [Pla+18] or a robust version of DDPG [Tio+20].

References

[Pla+18] Matthias Plappert et al. *Parameter Space Noise for Exploration*. Jan. 31, 2018. DOI: 10.48550/arXiv.1706.01905. arXiv: 1706.01905 [cs, stat]. URL: http://arxiv.org/abs/1706.01905 (visited on 12/11/2023). preprint.

[Low+20] Ryan Lowe et al. *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. Mar. 14, 2020. DOI: 10.48550/arXiv.1706.02275. arXiv: 1706.02275 [cs]. URL: http://arxiv.org/abs/1706.02275 (visited on 12/11/2023). preprint.

[Tio+20] Teckchai Tiong et al. "Deep Reinforcement Learning with Robust Deep Deterministic Policy Gradient". In: *2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE)*. 2020 2nd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE). Nov. 2020, pp. 1–5. DOI: 10.1109/ICECIE50279.2020.9309539. URL: https://ieeexplore.ieee.org/document/9309539 (visited on 12/11/2023).