report.md

Leticia Lorena Rodríguez - Decembre 2018 - Deep Reinforcement Learning Nanodegree - Udacity

Report: Collaboration and Competition Project

Introduction

This project solves the Unity3D environment Tennis.

Solution

It's solve the environment using a MutipleAgent DDPG. Each agent uses Actor and Critic networks defined in the following way:

	Actor		
Layer	Input Size	Output Size	Activation
1	24	400	relu
2	400	200	relu
3	200	2	tanh

	Critic		
Layer	Input Size	Output Size	Activation
1	(24+2)*2 = 52	400	relu
2	400	200	relu
3	200	1	linear

The DDPG Algorithm is defined:

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

The parameters used are:

```
BUFFER_SIZE = int(1e5)  # replay buffer size
BATCH_SIZE = 50        # minibatch size
GAMMA = 0.99            # discount factor
TAU = 1e-3              # for soft update of target parameters
```

```
LR_ACTOR = 1e-5         # learning rate of the actor
LR_CRITIC = 1e-4        # learning rate of the critic
```

In this case, using a MultiAgent, both agents are sharing the Replay Buffer. The learning is performed over both agents experience but the action selection only using the state of the current agent.

As it's indicated in the paper: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments

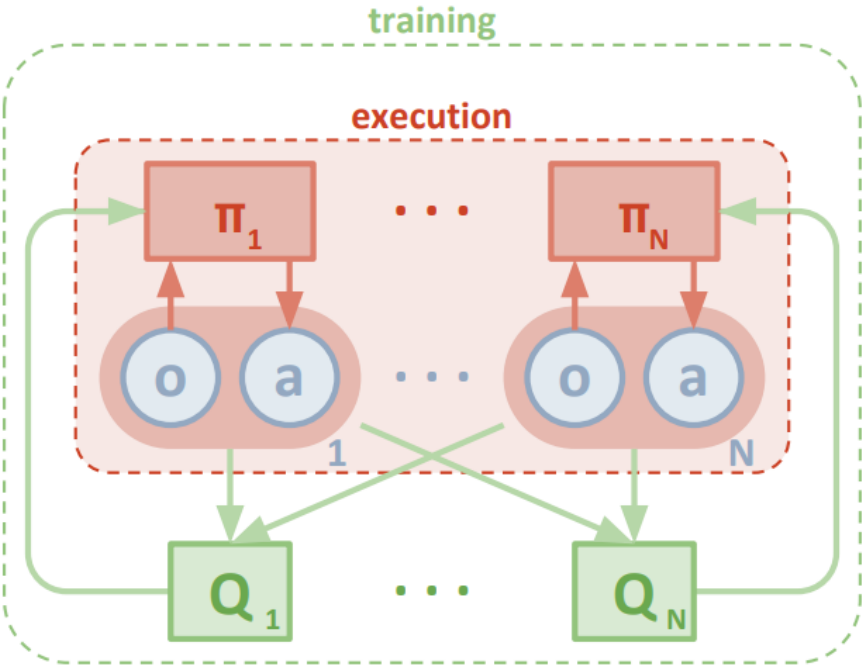
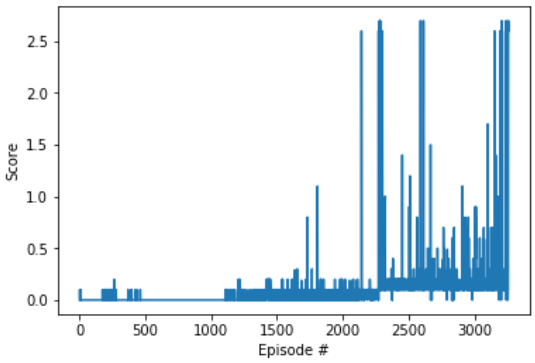


Figure 1: Overview of our multi-agent decentralized actor, centralized critic approach.

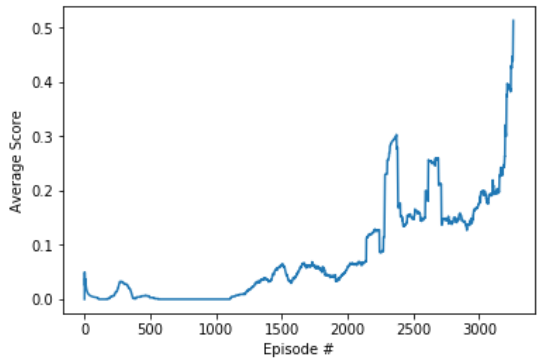
I've tried different batch sizes: 50, 200, 600 but 50 works well. I've noticed that the network definition have a strong impact in learning. The learning rate should also be carefully choosen.

The random generation have strong impact in results.

The plot of the rewards is:



The average reward evolution is:



Finally, the environment was solved in in episode: 3259 with an avg score: 0.51 .

Future work

Some ideas for future work are:

- Tunning network training: changing learning rates, optimizers, etc.
- Try implementing a MultiAgent PPO