Project 3. Collaboration and Competition

Zhamila Issimova

January 31, 2019

Abstract

In this report implementation of MADDPG algorithm for collaboration and competition problem is described. It contains brief information about this deep reinforcement learning technique as well as design choices which were made to train machine learning agent in Unity environment.

1 Introduction

The goal of the project is to teach two agents to play tennis game by controlling two rackets and trying to make a ball bounce over a net. This is a continuous reinforcement learning task for 2 competing agents, hence, multi-agent actor-critic algorithm [1],which is considered as extenstion of Deep Deterministic Policy Gradients (DDPG) algorithm [2] is used to solve this Tennis [3] environment problem. This DDPG algorithm has actor-critic architecture, where actor learns policy function and controls how our agent acts. Critic, on the other hand, learns value function and measures how good these actions are. Therefore, actor-critic algorithms combine value and policy gradient approaches. The MADDPG contribution is multi-agent decentralized actor, centralized critic approach in learning which will be discussed in next sections. The instability in learning and other results will be also presented and discussed.

2 Methodology

2.1 Environment

Collaboration and competition problem is represented in Tennis environment, which is provided by The Unity Machine Learning Agents Toolkit (ML-Agents). Since the goal of each agent is to maximize the score and keep the ball in play, a reward of +0.1 is provided if an agent hits the ball over the net, and a reward of -0.1 is provided if an agent lets a ball hit the ground or hits the ball out of bounds. After each episode, we add up the rewards that each agent received (without discounting), to get a score for each agent. This yields 2 (potentially different) scores. We then take the maximum of these 2 scores. This yields a single score for each episode. When average of this score is higher than +0.5 over 100 consecutive episodes, the environment is considered as solved. 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.

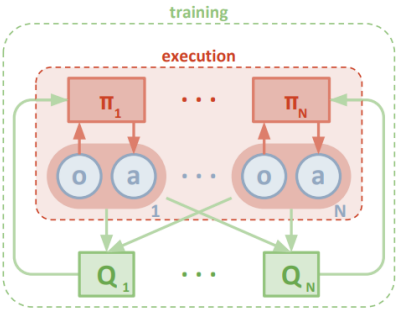

Figure 1: Multi-agent decentralized actor, centralized critic approach

2.2 MADDPG algorithm

Since action space is continuous and we have two competing agents, Multi-Agent Actor-Critic algorithm (or better known as MADDPG) perfectly fits to solve this problem. MADDPG algorithm can be regarded as an extension to DDPG algorithm which was studied earlier. The general approach for MADDPG is to use DDPG agents with all its tricks such as separate neural networks for actor and critic, exploration noise, experience replay buffer, local and target networks, soft update of these networks in layers were realized. These DDPG agents will interact with each other in multi-agent decentralized actor, centralized critic approach. In other words, actor network will be able to see only its own states(observations) and based on them make some actions. Critic networks will, on the other hand, have full observations of environment, i.e. critic will know all observations and actions of each agent. This approach can be seen in Figure 1.

2.3 Deep Neural Network

Since it is an actor-critic algorithm, separate neural networks were created for actor and critic. Actor network consists of 2 hidden fully connected layers with 256 and 128 units respectively. Hidden layers outputs were passed through ReLU function and final output was passed through tanh function to obtain action vector.
Critic network also consists of 2 hidden fully connected layers with 256 and 128 units respectively. It takes as input full observations (states and actions of all agents), since critic needs to evaluate in a centralized manner how good actor networks are performing. Hidden layers outputs of critic networks were passed through leaky ReLU function.

3 Results

The number of episodes were set as 15000. For the first 2000 episodes, random actions were chosen instead of following the policy for exploration purposes. The environment was solved in 4408 episodes as it can be seen from Figure 2. The problem was considered as solved if average score over 100 episodes is higher than 0.5. The average score during whole training can be seen on Figure 3.

```
Episode 4000    Average Score: 0.14
Episode 4100    Average Score: 0.10
Episode 4200    Average Score: 0.06
Episode 4300    Average Score: 0.07
Episode 4400    Average Score: 0.43
Episode 4408    Average Score: 2.60
Environment solved in 4408 episodes!    Average Score: 0.51
```

Figure 2: Environment solved in 153 episodes

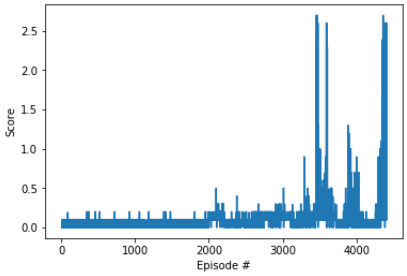

Figure 3: Learning curve

Due to the multi-agent nature of this problem, a bit of instability was experienced during training. For instance, on Figure 3 scores can be seen that between episodes 3300 and 4300 at some instances agents were performing well (above 0.5) and again failing to receive 0.5 reward, thus forming spikes in learning curve.

4 Conclusion

In this project extensive experience with MADDPG algorithm was obtained. MADDPG was adapted to Unity "Tennis" environment. The goal of collaboration and competition project was solved and agents were trained to have average score more than 0.5. In future, it is possible to extend this work and make agent to learn directly from visualized input using only raw pixels by solving "Soccer" environment.

References

[1] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments", *CoRR*, 2017.

[2] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, "Continuous control with deep reinforcement learning", *CoRR*, 2015.

[3] "Unity-Technologies/ml-agents", GitHub, 2019. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/master/docs/Learning-Environment-Examples.md. [Accessed: 03- Jan- 2019].