udacity Deep Reinforcement Learning Nano Degree Project 3 - Collaboration and Competition – Project Report

data.camp097@audi.de

April 23, 2020

Contents

1 Introduction

This report summarizes my implementation and results for the third project in the udacity Deep Reinforcement Learning Nanodegree. In the first section, the algorithm is briefly described, in the second section my implementation is described shortly. The results are presented and in the third section and a brief outlook on possible future improvements is given in the fourth section.

2 Algorithm

The solution I implemented is not really a dedicated multiagent RL algorithm. The problem is fully symmetric, both agents have the same observation space and reward function and collaboration does not require any communication. Hence it should be enough to train just one agent with the experiences gathered by the two identical agents in self-play.

3 Implementation

To achive this I used the DDPG code from my 2nd project [1] and modified it only microscopically at the following places:

- Outer Learning Loop. I removed the constraint on time steps and terminate if one of the agents reaches a terminal state.
- Solution Criterion. The solution criterion is given as the average of the max over 100 rounds.
- Agent step and act. The return values of the environment are all lists with 2 elements, one each for each agent. Hence a few modifications are required to adapt the OU-Noise, the act and step methods of the agent class.
- Replay Buffer. The experience tuples of both agents are added to the replay buffer.

This is enough to solve this environment. If the task would be more complex or the agents would have different observation spaces or reward functions, this approach would most certainly not work.

3.1 Network Architecture

The architecture of the networks for actor and critic was left unchanged, so it might not be the best choice and we might get away with something less complicated.

3.1.1 Network Architecture Actor

The actor network is a basic MLP with two hidden layers (128,128) and ReLU-Nonlinearity. There is one batch normalization before the second layer and the weights are inititialized Kaiming-like [1].

3.1.2 Network Architecture Critic

The critic network is a basic MLP with two hidden layers (128,128) and ReLU-Nonlinearity. There is one batch normalization before concatenation with the action before the second layer and the weights are inititialized Kaiming-like [1]. This staged configuration of input was suggested in the original DDPG paper [2].

3.2 Hyperparameters

I present here my first set of hyperparameters that solved the environment. Starting from the hyperparameters from the 2nd project, I increased both learning rates by a factor of 30 to get any learning started at all. I halved the exploration noise as I figured the two agents with their different experiences contribute to the exploration.

[1] My code for the second project again was based on the DDPG example in the udacity DRL Nanodegree repo https://github.com/udacity/deep-reinforcement-learningwith only minimal modifications to run on the unity ML environment.

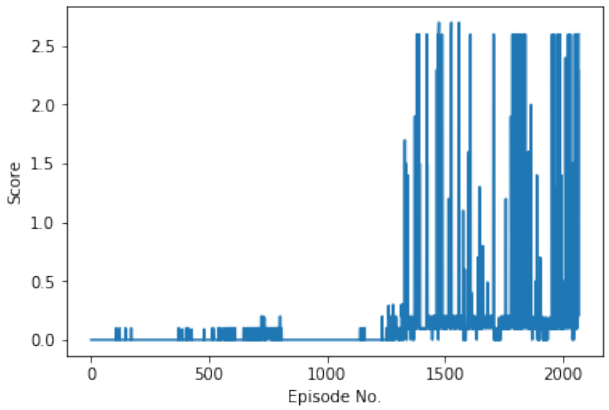

Figure 1: Learning Curve

- Size of the replay buffer: 1e5
- Batch size: 128
- Discount factor γ: 0.99
- Soft update τ: 0.001
- Exploration (OU) noise σ: 0.05
- Exploration (OU) noise θ: 0.15
- Exploration (OU) noise μ: 0
- Learning rate actor: 1e-3
- Learning rate critic: 1e-3
- Weight decay: 0

4 Results

This section briefly shows the learning curve of the implemented algorithm. The learning curve (score as a function of episode number) is shown in figure 1. The training yields:

```
Episode 100 Average 0.00
Episode 200 Average 0.00
Episode 300 Average 0.00
Episode 400 Average 0.00
Episode 500 Average 0.01
Episode 600 Average 0.02
Episode 700 Average 0.02
Episode 800 Average 0.06
Episode 900 Average 0.00
```

```
Episode 1000 Average 0.00
Episode 1100 Average 0.00
Episode 1200 Average 0.01
Episode 1300 Average 0.04
Episode 1400 Average 0.31
Episode 1500 Average 0.32
Episode 1600 Average 0.28
Episode 1700 Average 0.21
Episode 1800 Average 0.29
Episode 1900 Average 0.33
Episode 2000 Average 0.26
Episode 2067 Average 0.51

Environment solved in 2067 episodes!
```

Learning is neither fast nor stable. Future work should address both issues.

5 Outlook

There are many interesting topics for future exploration and research:

- The most obvious next step would be to implement a true multi- agent algorithm like MADDPG. [3]
- As suggested in the udacity classroom it could be fun to try an even more challanging task in continuous control as e.g. the unity ML Soccer.
- It might stabilize the learning if we would decay the exploration as is often used in value basesd methods with ϵ-greedy exploratioin strategies. This would mean to modify the parameters of the Orstein–Uhlenbeck-Noise in our agent.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," 2015.

[2] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[3] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2017.