Learning to Collaborate With Deep Reinforcement Learning

Uirá Caiado

According to [2], many important applications involve interaction between multiple agents, as multi-robot control, analysis of social dilemmas and so on. However, multi-agent environments present additional challenges for traditional reinforcement learning (RL) approaches. As each agent's policy is continually changing, the environment becomes non-stationary from the perspective of any individual agent.

In this project, I used the Multi-Agent Deep Deterministic Policy Gradient (DDPG) model to solve the Unity Tennis[1] environment, where two agents control rackets to bounce a ball over a net. A reward of +0.1 is provided for each step that the the agent hits the ball over the net. If the agent lets a ball hit the ground or hits the ball out of bounds, it receives a reward of -0.01.

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. To solve the environment, the agents must get an average score of +0.5 (over 100 consecutive episodes, after taking the maximum over both agents).

The algorithm implemented in this project uses an actor-critic architecture to deal with the high-dimensional action space. As described by [1], the model maintains a parameterized actor function $\mu(s|\theta^\mu)$ which specifies the current policy by mapping states to a specific action. The critic $Q(s, a;\ \theta)$ is learned using the Bellman equation as in Q-learning and is used to evaluate the actions chosen by the actor. The resulting evaluation is used to compose a baseline (or advantage function) that is then used to train the actor model.

The parameters θ from both actor and critic models are learned by neural networks with almost the same architecture. The input to these networks consists of a matrix 24×1 produced by the environment and the first layer is a fully-connected linear layer with 400 neurons and applies a rectifier nonlinearity. It is followed by a hidden layer also consisting of a fully-connected linear layer with 300 neurons followed by another rectifier. The output layer of the actor-network consists of a fully-connected linear layer with a vector 2×1 as output for each agent. The output layer of the critic network includes a fully-connected linear layer with a single output for each agent followed by a *tanh* activation function that naturally constrains the values between $[-1, 1]$. For the model used in this project, the critic network have some particularities. First, the actions are included in the first layer of the critic network, similar to what was done by [1]. Second, as proposed by [2], the critic network is augmented with extra information about the state and actions of the other agent, so the input space size is doubled.

The MADDPG model was introduced by [2], and it is an extension of the DDPG algorithm, proposed by [1]. One of the main differences from the original model is that, while the critic is trained with extra information about the policies of other agents, as explained before, the actor only has access to local information.

The role of the actor in this method is to estimate the stochastic policy that returns the probability distribution over the actions. The critic is used to calculate the Q-values of the actions so it can be updated using the Bellman equation to find the approximation of $Q(s, a)$ and minimizing the MSE objective. As explained by [1], the actor is updated by applying the chain rule to the expected return obtained by the critic network from the actor parameters. As this method is off-policy, we can use the same tricks used by the DQN model, presented by [3], as the replay buffer and the use of local/target networks to help stabilize the parameters of both actor and critic.

I used a replay buffer size of $100,000$ experiences, mini-batches of 200 instances and a $\gamma = 0.99$ to discount the rewards. The learning rate of the actor was 0.0001, to the critic was 0.001, and the

[1] Source: `https://bit.ly/2OOkxKo`

networks were updated every two steps. As suggested by [1], the target networks slowly tracked the local networks using $\tau = 0.001$. For detail on how each one of these parameters was used in the model, check the papers cited or the code provided along with this report.

Different from the original paper, I used a separated actor and critic networks for both agents. Also, both local and target networks presented the same architecture, described before. Using this configuration, I was able to solve the environment in 1764 episodes. The figure 1 presents the results of the simulation. The left panel exposes the rewards obtained in each episode and, in the right panel, I plotted a moving average of the last 100 episodes of this rewards. The shaded area corresponds to one standard deviation σ of the rewards of those 100 episodes.

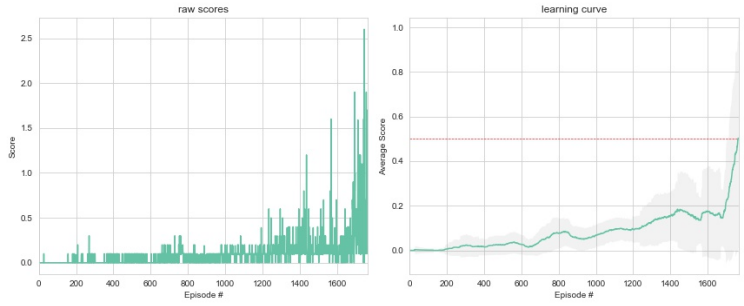

Figure 1: MADDPG - Learning Curve(left panel) and Moving-average(right panel)

Finally, as some possibles extensions, the parameters of the models could be better tuned, and another model could be added to the analysis, as the Proximal Policy Optimization (PPO) method, introduced by [4]. The authors suggested a new policy gradient method to deal with discrete action space environments. Due to the high-dimensional action space of the task explored in this project, we could adapt the model described n the paper to use an actor-critic approach. The policy network would play the role of the actor. The critic could be implemented as in MADDPG, including the states and action of both agents as the input to the networks. It would allow the agents using PPO to deal with the action continuous space and the multiagent environment. According to the original paper, the model strikes a balance between ease of implementation, sample complexity, and ease of tuning. So, we could compare the PPO against the model implemented in this project to check if these observations hold in the Tennis environment.

References

[1] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv.org*, September 2015.

[2] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6379–6390. Curran Associates, Inc., 2017.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[4] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv.org*, July 2017.