Tennis Collaboration and Competition Project

Tiago Montalvão

June 23, 2020

1 Introduction

This is a report of the third project of Udacity Deep Reinforcement Learning Nanodegree. It describes the model architecture with chosen hyperparameters used in the agent, the experiments, and potential improvements upon the current results.

2 Implementation

2.1 Agent

The agent was implemented using Multi Agent Deep Deterministic Policy Gradient (MADDPG). This algorithm is basically an enhancement of Deep Deterministic Policy Gradient (DDPG) for dealing with multiple agents learning at the same time in an environment.

It is categorized as an Actor-Critic method, because there are two networks (actor and critic) collaborating to achieve the agents' results. Basically, the actor learns to pick actions from states, while the critic learns to calculate the state-action Q value function.

As described in the original MADDPG paper [1], one actor and one critic is trained for each agent, but the critics are trained in a **centralized** way, meaning they use observations from all the agents during training phase, while the actors use observations only from their corresponding agent. In this way, the critic assesses the actions in states with a stationary view of the environment.

The DDPG was then implemented with:

- **Experience replay buffer** to handle data correlation between consecutive samples.
- **Two neural networks** with identical architecture (local and target networks) for each actor and critic, as described in the following section (so there were 4 networks in total for each agent). **Soft update** was also implemented to updated each target network, instead of updating the whole network every n-th iteration.
- **Random gaussian noise** implemented to add noise to the actions, thus creating incentives to exploration.

It is worth mentioning that the implementation was based on previously implemented DDPG on the Nanodegree, but a few adjustments were made in the hyperparameters and in the code itself. The adjustments were:

- Ornstein–Uhlenbeck process **replaced** with simple **gaussian noise** and exponential decay to noise inside an episode.
- Replay buffer support for returning tuples with the shape (n_agents, batch_size, object_size), in which the object_size denotes the size of the object being returned (e.g. states, actions, rewards, etc.).

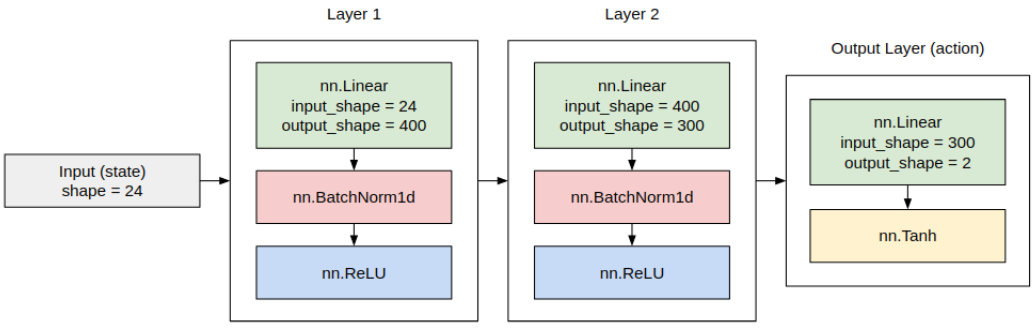
- Gradient clipping in the critic network update.
- Instead of sampling from the replay buffer and learning from these tuples every iteration, a counter was implemented to only update the network each UPDATE_EVERY iterations.
- Each UPDATE_EVERY iterations, tuples were sampled for each agent and this agent's networks were updated a few times in a row (N_UPDATES_PER_STEP hyperparameter).
- 1D Batch Normalization added to both actor and critic networks.

These adjustments gave a lot of stability for the learning process.
The chosen hyperparametes are described above:

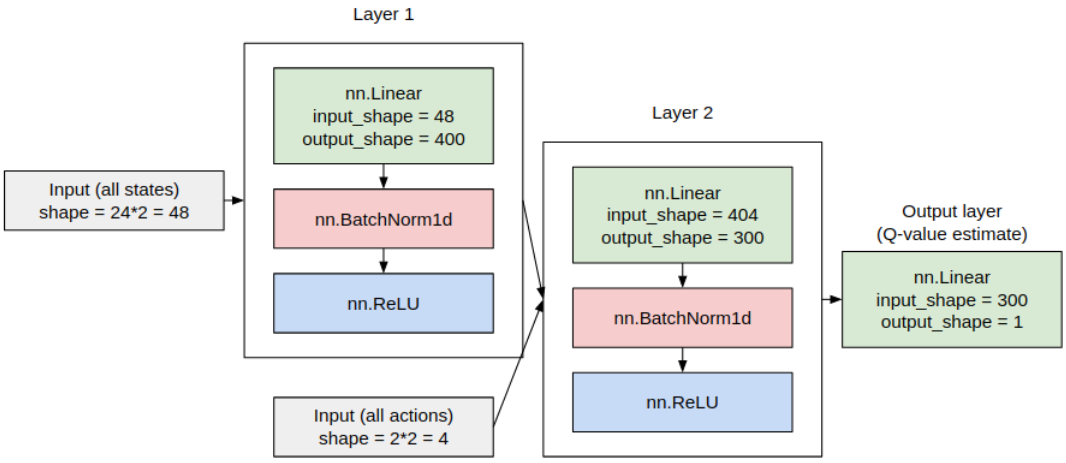
```
BUFFER_SIZE = int(1e6)     # replay buffer size
BATCH_SIZE = 256           # minibatch size
GAMMA = 0.99               # discount factor
TAU = 1e-3                 # for soft update of target parameters
UPDATE_EVERY = 5           # how often to update the network
N_UPDATES_PER_STEP = 5     # number of updates to perform at each learning step
INITIAL_NOISE = 1.0        # initial value for noise multiplier
NOISE_DECAY = 0.99925      # value to multiply noise after each step
LR_ACTOR = 5e-4            # learning rate of the actor
LR_CRITIC = 1e-3           # learning rate of the critic
WEIGHT_DECAY = 0           # L2 weight decay
```

2.2 Model architecture

The architecture for the actor network is as follows:



The architecture for the critic network is as follows:

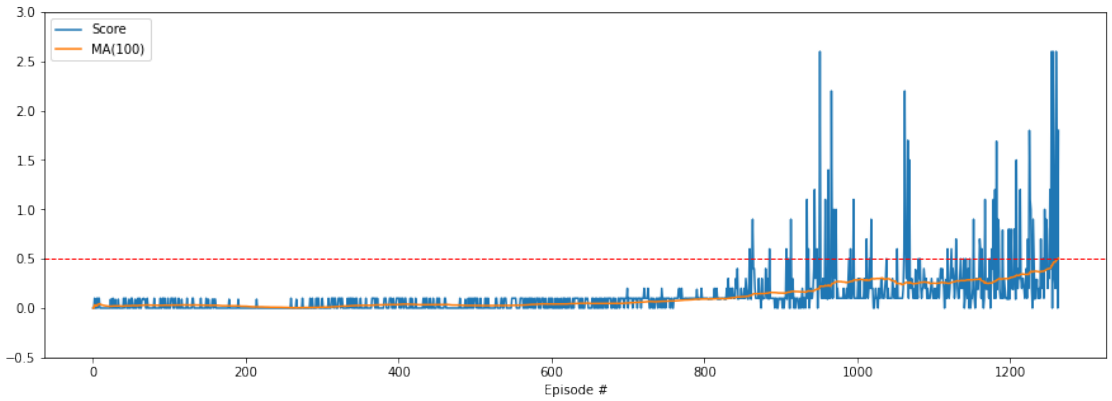


In the project notebook, there is a cell which executes a command that gives the model summary (the same info above with some numbers). Based on this, we can get the information that there are 132302 trainable parameters in the actor network and 142801 trainable parameters in the critic network.

3 Results

The environment was solved in 1264 episodes, and it took it about 66 minutes to train the agents. All the details can be checked in the project notebook. It can be seen that thes agents struggled a bit in early episodes, but could learn well from around episode 900 onwards, as can be seen in the GIF in the repository [1] README.md.

The following image shows the individual episode score (in blue), the moving average of the last 100 episodes' scores (in orange), and the threshold of +0.5 for the moving average (in red), so that the environment is considered solved.



[1] https://github.com/tiagomontalvao/DRLND-CollaborationAndCompetition

4 Ideas for Future Work

MADDPG is one of the most classic actor-critic algorithms for multi agent environments. I would like to train other algorithms, like MAPPO, to compare convergence results. I could also try to apply single agent DDPG for each agent to assert that it gets very unstable and is not able to learn well.

Another improvement could be the addition of **Prioritized Experience Replay**, giving more weight in experience sampling from replay buffer for tuples that have larger TD errors.

References

[1] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments, 2017.