Tennis with MADDPG

In this report my efforts on solving the Tennis environment made by Unity are described. The algorithm of choice is MADDPG (Multi Agent Deep Deterministic Policy Gradient), which is very similar to solving a single agent environment with “ordinary” DDPG.

The main difference is that in the case of the Tennis environment, two agents need to be created, which happens in the class MultiAgent. This class also creates a single replay buffer, which is accessed by both agents. Of course, it is also possible to create a replay buffer for each agent, which can be useful if the perspective on the environment differs strongly between the agents.

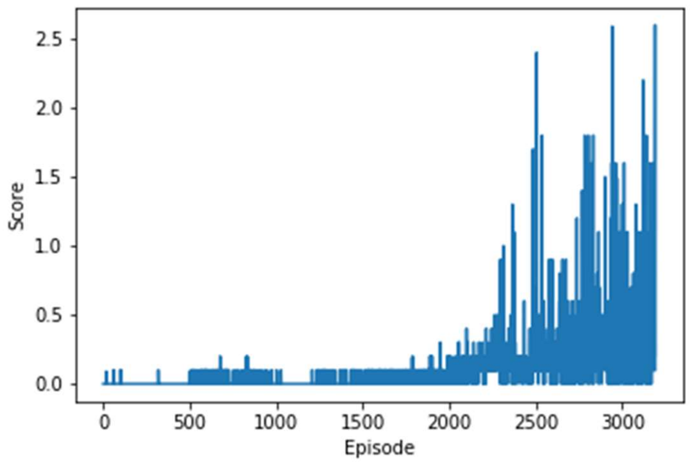
The class MultiAgent has two functions, act() and step(). If one of these functions is called, for both agents the function act() or step() function of the class DDPG are called. In the case of the step() function, after every four steps made, a sample is taken from the replay buffer and the model weights are updated.

The DDPG model has an Actor and a Critic network, both consist of a local and target network. The Actor is a straightforward feed forward network with two hidden layers (128 neurons each).
The Critic has two inputs, states and actions, and returns a single value (Value of the actions that were done). The hidden layers consist of 68 (64 + action_size), 32 and 16 neurons.

The other hyperparameters of the model are:

BUFFER_SIZE = int(1e6)
BATCH_SIZE = 512
DISCOUNT = 0.99
TAU = 1e-3
LR_ACTOR = 1e-4
LR_CRITIC = 3e-4
UPDATE_EVERY = 2

Figure 1. Scores of individual episodes of the Tennis environment.



The environment is solved in 3192 episodes. This is probably not the fastest solution ever for this problem. As you can see, the agents start learning effectively only after there have been 2000 episodes, and until the end the learning process remains unstable. This might have to do with the implementation with two distinct Actor networks, one for each agent. I have tried to tweak the network architectures (number of neurons) and properties of the replay buffer, but that did not really help improving the training process.

Recommendations

Things that may improve learning are the use of prioritized experience replay, in which more important past experiences are preferred over random sampling. Also sharing the training process by a shared critic might give the training process a boost. I have not spent much time on tweaking the learning rates of the Actor and the Critic, which might also be recommended in further research.