Udacity - Collaboration and Competition

Student's name: Paulo Cotta

Abstract

I apologize for the English, but it is not my native language and if you have any problems with writing, please feel free to point out the errors.

In this report, I briefly summarize the learnings and final modeling decisions made as part of the collaboration and competition project. After several attempts with different hyperparameters and models, I was able to find a configuration that solves the environment with about 1477 steps. Several optimizations were implemented, but the main criterion for resolving the environment in my experience was long enough training time (enough episodes).

Hyperparameters used:

- BUFFER_SIZE = int(1e5)
- BATCH_SIZE   = 128
- GAMMA   = 0.99
- TAU   = 1e-3
- LR_ACTOR   = 2e-4
- LR_CRITIC   = 2e-4
- WEIGHT_DECAY = 0

Parameters used in the call function used (def ddpg_tennis):

- n_episodes=8000
- max_t=1000

Object Model

The learning algorithm is a gradient of deep deterministic policy from various agents. DDPG is an actor-critical algorithm and mainly uses two neural networks, one for the actor and one for the critic. These networks calculate action vectors for the current state and generate a time difference error signal at each stage of time.

The DDPG uses a stochastic behavioral policy for good exploration and a deterministic target policy for estimating. The current state is the input of the actuator network and the output is a single value that represents the action. The deterministic theorem of the policy gradient provides the update rule for the weights of the network of actors. This for each agent instantiated to play.

The critic's output is simply the estimated Q value of the current state and the action taken by the actor. The critical network is updated from the gradients obtained in the TD error signal. The basis of the algorithm was from the previous exercise, only with a big difference that there were two agents competing with each other and in the other the environment was uniform. Hence the decision to use the same architecture.

Model Architecture

Throughout the experimentation phase, the basic network architectures have not changed much in terms of the number of layers and units: there have always been 2 hidden layers totally connected with ReLu activations for actors and critics. I tried several combinations for the two hidden layers and used 128/128 in my final configuration and the Adam optimizer.

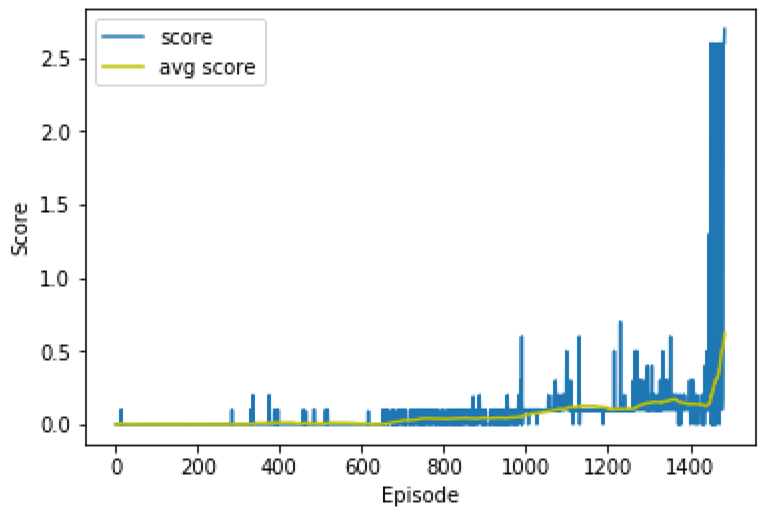
In this work, the ddpg.py and model.py files.

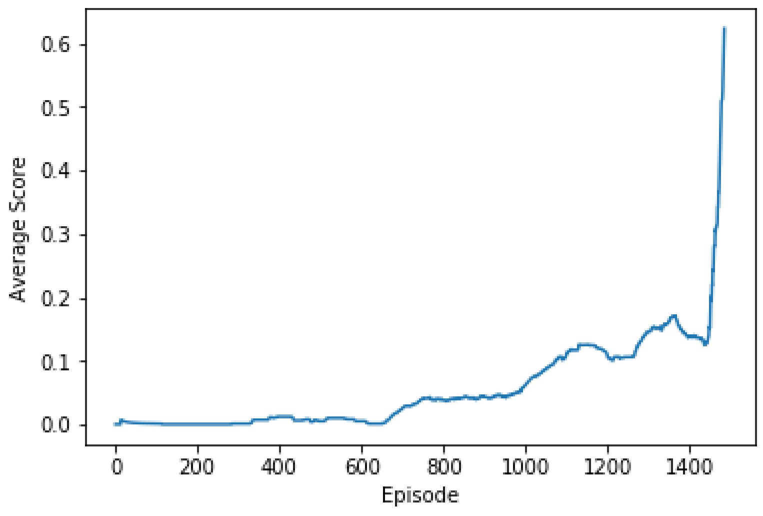
Final result - Conclusion

The model reached the goal in episode 1477, where it managed to achieve the goal as shown below:

- Episode 100 Score 0.10 Average Score 0.00
- Episode 200 Score 0.10 Average Score 0.00
- Episode 300 Score 0.10 Average Score 0.00
- Episode 400 Score 0.20 Average Score 0.01
- Episode 500 Score 0.20 Average Score 0.00
- Episode 600 Score 0.20 Average Score 0.00
- Episode 700 Score 0.20 Average Score 0.03
- Episode 800 Score 0.20 Average Score 0.04
- Episode 900 Score 0.20 Average Score 0.04
- Episode 1000 Score 0.60 Average Score 0.06
- Episode 1100 Score 0.60 Average Score 0.11
- Episode 1200 Score 0.60 Average Score 0.11
- Episode 1300 Score 0.70 Average Score 0.15
- Episode 1400 Score 0.70 Average Score 0.14
- Score of 0.509400007613 achieved in 1477 episodes

Below are the graphs after execution:





References

Book - Reinforcement Learning, Sutton