Continuous Control Project
Deep Reinforcement Learning Nanodegree

Borja González
January 12th, 2018

Overview

The goal of this report I briefly summarize the learnings and final modeling decisions taken as part of the Navigation project.

Neural Network Architecture

For the first attempts I used a 3-layer architecture as I did with the Navigation project with 128, 64 and 64 neurons, however the agent was not able to improve from an average reward between 3 and 4. Then, checking Udacity's bipedal environment solution with DDGP I tried a 2-layer network with leaky ReLUs for the critic Network. This didn't make any improvements on its own, however, adding Batch Normalization after the first layer as suggested at the student hub showed better results, although still far from solving the environment.

After adding all the improvement, I mention in the section below I tried with a network with two hidden layers of 256 and 128. This one was another big step; however, it didn't seem able to solve the problem. Therefore, I tried the same configuration but with a 256/256 architecture. This one did solve the challenge.

Further Improvements

First attempts showed that the Neural Network and hyperparameters alone weren't the only problem so I looked for more.

An improvement suggested in the benchmark implementation in the project was adding gradient clipping. Adding this in the early stages didn't showed an immediate benefit, but it gave slightly improvements and I left for the final version.

Similar happened with another suggestion from the student hub about initializing the weights of the local and target critic to the same values.

The key addition was to change the training with an idea from the benchmark implementation, which was about doing the training only every 20 episodes. This almost solved the problem with the 256/128 network and solved it with the 256/256 configuration.

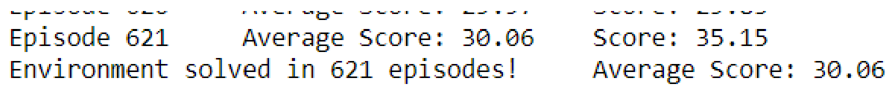
Choice of hyperparameters

Final choice of hyperparameters was:

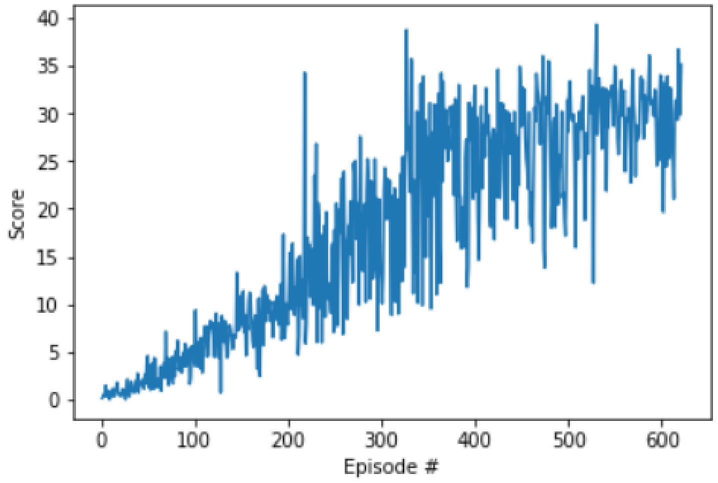
```
BUFFER_SIZE = int(1e6)          # replay buffer size
BATCH_SIZE = 128                # minibatch size
GAMMA = 0.99                    # discount factor
TAU = 1e-3                      # for soft update of target parameters
LR_ACTOR = 2e-4                 # learning rate of the actor
LR_CRITIC = 2e-4                # learning rate of the critic
WEIGHT_DECAY = 0                # L2 weight decay
N_LEARN_UPDATES = 10            # number of learning updates
N_TIME_STEPS = 20       # every n time step do update
```

Results

The agent was able to solve the environment after 621 episodes, which means after 31 ten-learning step updates



```
Episode 621     Average Score: 30.06     Score: 35.15
Environment solved in 621 episodes!      Average Score: 30.06
```



Further Work

First, I would like to do further tweaks to try to solve the environment in less episodes. Then I will try the 20-agent problem and I will work with different algorithms such as A2C or PPO. Then I will move to the crawler environment.