DDPG implementation for Tennis challenge

Jonas Stiernström

Algorithm

The algorithm used was a DDPG, same DDPG model used in the continuous learning project.. Some changes were made to hyperparameters.A smaller batch size was used, the Tau was increased and the learning rates were changed for both actor and critic. Dropout was also added to the actor and critic models to help with stability.

The actor in the algorithm estimates the best action to take, and the critic then use this estimation to create the optimal action value.

As for parameters experimentation led me to the some changes from the ones used in the previous project. The full set of hyperparameters are the following:

BUFFER_SIZE: 1e5 (as an int)

BATCH_SIZE: 64

Gamma 0.99

TAU: 2e-1

DDPG for Tennis Challenge

LR_ACTOR: 1e-4

LR_CRITIC: 3e-4

WEIGHT_DECAY: 0

And for the model, the actor consists of 3 fully connected layers. The first two using relu as the activation function, but the last one using a tanh activation function. The layer input sizes for the actor is as follows: 33, 128, 128, 4(output size)

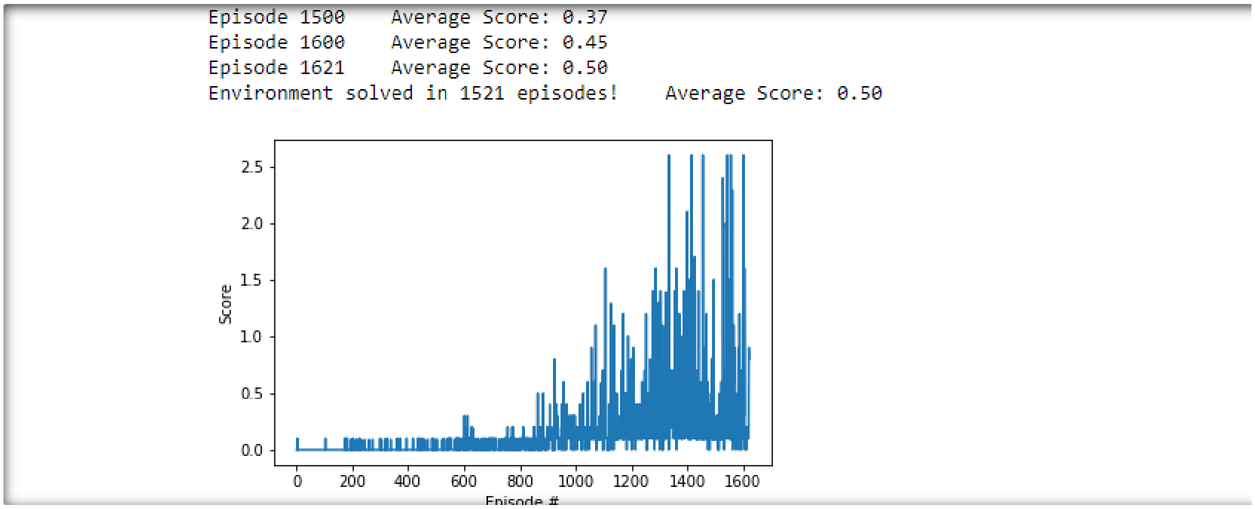
The critic is mostly the same, but using relu as the output activation function instead of a tanh. The critic differs only in that it in the second layer concatenates the input with the actions. And that while the actor has an output size of 4, the critic has an output size of 1.

Both the actor and critic use dropout for the second FC layer with a probability of 0.2, which helped the stability of the models.

Result

The result was in the end very pleasing - see the graph:

DDPG for Tennis Challenge



```
Episode 1500	Average Score: 0.37
Episode 1600	Average Score: 0.45
Episode 1621	Average Score: 0.50
Environment solved in 1521 episodes!	Average Score: 0.50
```

Experiments

While there were a lot of iterations and experiments when it came to finding the right hyperparameters for the task, the best experiments came after I added dropout and started seeing more stable results. And an increase of the Tau value combined with a lower batch size was a winner that sent me on the way. All I had to do after that was to find the right combination of learning rates.

Improvements

Future improvements could definitely be made. And something I read on the forums about was the implementation of Batch Normalization. I would be interested in adding this later as it could improve the model. And the course shows ways on how to do it so I could use that as a guide when it is time to improve on it further. I am also intrigued by testing a deeper network, as this could also help the model reach higher scores.