DDPG implementation for Reacher challenge

Jonas Stiernström

Algorithm

The algorithm that eventually succeeded was a DDPG model based on the one used in the course. Main changes was made to enable the use of 20 agents. For example the Agent now takes in an additional parameter called agents, which decides how many agents you are using.Which is then reflected into the OUNoise function.

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 following changes from the ones used in the course:

- Learning rate for the actor was lowered to 1e-3 from 1e-4
- FC nodes were for both actor and critic lowered to 128

The full set of hyperparameters are the following:

BUFFER_SIZE: 1e5 (as an int)

DDPG for Reacher Challenge

BATCH_SIZE: 128

Gamma 0.99

TAU: 1e-3

LR_ACTOR: 1e-3

LR_CRITIC: 1e-3

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.

Result

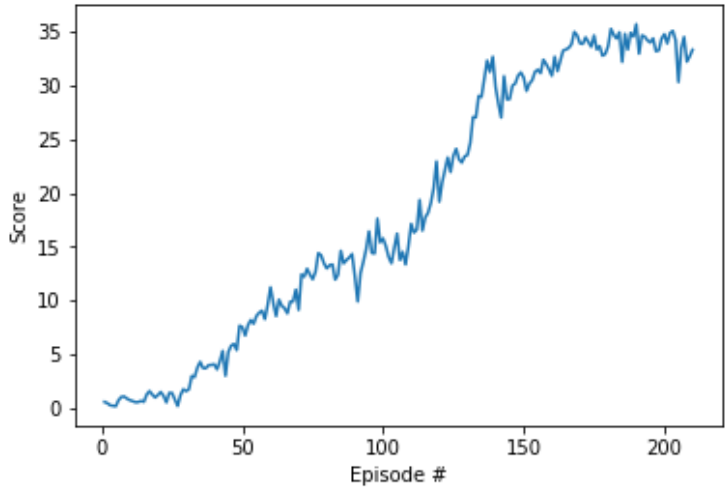
The result was in the end very pleasing - see the graph:

DDPG for Reacher Challenge

In [7]:
```
scores = ddpg()
```

```
Episode 100     Average Score: 7.05
Episode 200     Average Score: 28.19
Episode 210     Average Score: 30.06
Environment solved in 110 episodes!     Average Score: 30.06
```

In [8]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
plt.plot(np.arange(1, len(scores)+1), scores)
plt.ylabel('Score')
plt.xlabel('Episode #')
plt.show()
```



Experiments

To reach the last model I had to do some experimenting. Here are some examples of what I went through:

Experiment 1: Base DDPG used in the course. Result: 0.67 after 50 episodes. Next few experiments then focused on the learning rate to see how my results changed.

DDPG for Reacher Challenge

Experiment 5: A learning rate of 1e-3 for the actor was seemingly giving me better results (0.65 after 30 episodes). It was time to start looking at the model.

Experiment 6: I changed all FC layer nodes to 256 from 300/400. Result was an increase in the score to about 1.19 after 50 episodes, but it stagnated and never took speed again. But the increase intrigued me and I decided to half them again.

Experiment 7: With FC nodes at 128, the result was 1.99 after 50 epochs. And speed was much better. For some reason everything kind of stopped after that however. For the next experiments I decided to try 20 agents.

Experiment 12: After some great issues making 20 agents work and/or make any difference at all - a friendly person in the community helped me reset the workspace. This helped and now things picked up speed. I had changed a lot of parameters during the last 5 tests and resetted them back to where they were at experiment 7 now when things were working again.

Experiment 13: This was the version that I submitted, and I was shocked by how much faster the learning was with 20 agents, using the same parameters I used to train one agent back at experiment 7!

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.