Report of Deep reinforcement learning Nano-degree Project 2: Continuous Control

2019-2-19

This report summarizes the brief detail of my implementation of this project.

Environment

In this environment, the subject is to train 1 or 20 double-jointed arms to follow the moving target point in space. A reward of +0.1 is provided if the agent hand is in the target point location. The required goal is to train the agent to achieve more the 30 average points at least. In this project, two versions of the environment are ought to be chosen, one has only 1 agent, the other has 20 agents training at the same time.
Here I resolved the first version with only 1 agent.

DDPG

I borrowed the typical Deep Deterministic Policy Gradients (DDPG) framework of the former lessons, the neural networks of the actor-critic have the structures as follows:

Actor:
Full connected hidden layer 1: (input states, 256)
ReLU function
Batch Normalization of layer 1
Full connected hidden layer 2: (256, 128)
ReLU function
Output layer (128, actions)
Tanh function

Critic:
Full connected hidden layer 1: (input states, 256)
ReLU function
Batch Normalization of layer 1
Full connected hidden layer 2: (256+action_size, 128)
ReLU function
Output layer (128, 1)

Linear function

Hyperparameters of the agent:

```
BUFFER_SIZE = int(1e5) # replay buffer size
BATCH_SIZE = 128              # minibatch size
GAMMA = 0.99                    # discount factor
TAU = 1e-3                        # for soft update of target parameters
LR_ACTOR = 1e-4                # learning rate of the actor
LR_CRITIC = 1e-4              # learning rate of the critic
WEIGHT_DECAY = 0              # L2 weight decay
OUNoise: mu=0., theta=0.15, sigma=0.05
```

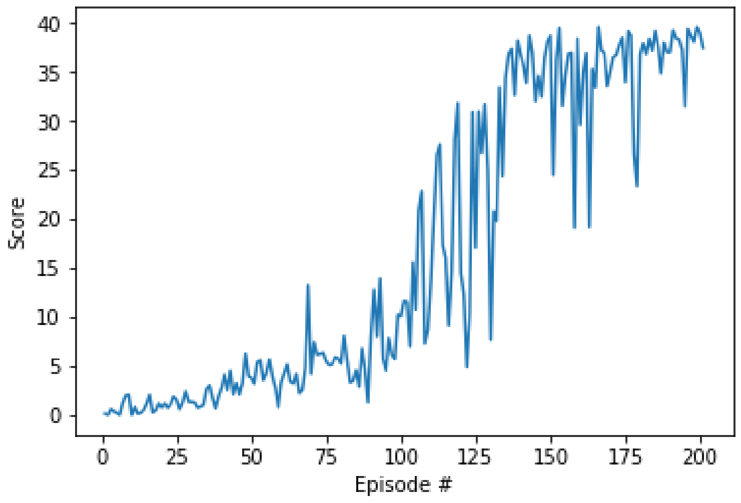
Reflection and Results

In this project, I encountered the many times of failure in training the agent. I applied the

torch.nn.utils.clip_grad_norm(self.critic_local.parameters(), 1)

at first, as the benchmark implementation suggested. Also, I add the batch normalization as I read about in the slack discussion, yet the average score can only reach 1-2 without the ability to learn.

Later on, as I review the DDPG code while doing the parameter tuning, I tried to increase the layer dimension, or the buffer size, it doesn't make much difference. However, when I turn down a little of the OUNoise sigma, the agent seems to reach higher score, but will still drop after several episode. Thus, I further tuned down the sigma (from 0.2 to 0.05) and changed the uniform random distribution into a normal distribution. The agent starts to learn and reached the goal in around 200 episodes, as shown in the following figure.



It seems the noise would interrupt the agent from learning and a much smaller noise must be used in this situation.

Perspective

In the next, the implementation for 20 agents can be tried out, with other algorithms such as PPO, A3C, or D4PG. It would be interesting to learn about the implementation of these algorithms and try to compare with each other.