

Collaboration and Competition

November 6, 2018

1 Methods

I leveraged many components of the codes provided by Udacity to solve OpenAI Gym’s Pendulum environment. In this project I used a policy-gradient actor-critic algorithm which is described in [1]. The Actor-Critic learning algorithm represent the policy function. The policy function is called the actor, and the value function called the critic. Given the current state of the environment, the actor produces an action and the critic produces a TD error signal given the state and reward. In order for the critic to estimate the action-value function $Q(s,a)$, it requires the output of the actor. The output of the critic drives learning in both the actor and the critic. In this project I used neural networks to represent the actor and critic structures.

A major challenge of learning in continuous action spaces is exploration. An advantage of off-policies algorithms such as DDPG is that we can treat the problem of exploration independently from the learning algorithm. I constructed an exploration policy μ' by adding noise sampled from a noise process \mathcal{N} to our actor policy

$$\mu'(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N} \quad (1)$$

\mathcal{N} can be chosen to suit the environment. As detailed in the supplementary materials we used an Ornstein-Uhlenbeck process [2] to generate temporally correlated exploration for exploration efficiency in physical control problems with inertia [1]. The detail of algorithm is provided as follow1. A recurring problem in multi-agent reinforcement learning is the environment non-stationarity due to the agents changing policies. This is particularly true in competitive settings, where agents can derive a strong policy by overfitting to the behavior of their competitors. Such policies are undesirable as they are brittle and may fail when the competitors alter strategies. Since the game requires two agent, each agent provides the state, action, reward and next while interacting with the environments

Hyper Parameters used in this study are as follow.

Neural network with following architecture was used for DDPG actor and DDPG Critic.

Table 1: Hyper Parameters

Parameter	Value
Replay Buffer Size	1e6
Mini Batch Size	1024
Discount Factor	0.99
Soft Update Target	1e-3
Learning Rate Actor	1e-3
Learning Rate Critic	9e-4
L2 Weight Decay	0

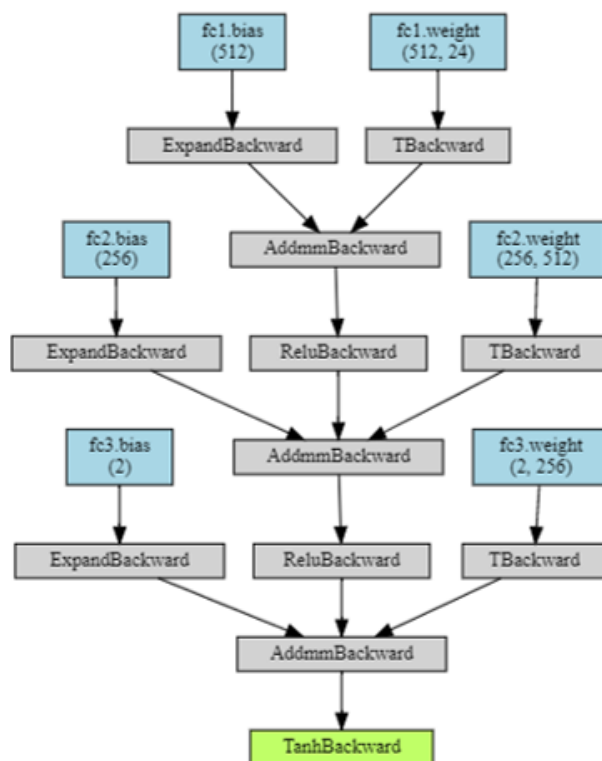


Figure 1: Neural network Architecture

Algorithm 1 DDPG algorithm [1]

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
 Initialize a random process \mathcal{N} for action exploration
 Receive initial observation state s_1
 for t = 1, T **do**
 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
 Execute action a_t and observe reward r_t and observe new state s_{t+1}
 Store transition (s_t, a_t, r_t, s_{t+1}) in R
 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
 Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

end for
end for

2 Results

The algorithm was converged in 1779 iterations with hyper-parameters noted in the method section. Figure 2 shows the score versus episode numbers.

3 Future Direction

- Hyper parameter tuning of algorithm to improve the performance.
- Tuning the structure of the Neural network to improve the performance.

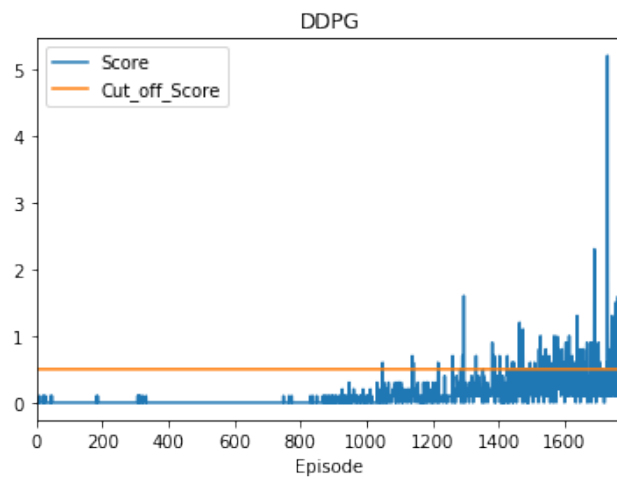


Figure 2

References

- [1] Timothy, P., Jonathan, J., Alexander, P., Nicolas, H., Tom, E., Yuval, T., David, S., and Daan, W. “Continuous control with deep reinforcement learning”. *arXiv preprint arXiv:1509.02971*.
- [2] Uhlenbeck, G. E., and Ornstein, L. S., 1930. “On the theory of the brownian motion”. *Physical review*, **36**(5), p. 823.