ECE 276C Assignment 4 Report

Mingwei Xu A53270271

Fall 2019

1 Question 1 - DDPG

In this question, we implemented DDPG to solve the 2-DOF reacher environment. The DDPG parameters are the same as suggested in the original paper [1].

1.1 Network Construction

The actor network is a four layer fully-connected neural work with hidden layer size of 400 and 300. Activation functions are two ReLUs and one tanh at last. Weights of nodes are initialized from uniform distributions, with parameters same as proposed in the paper.

The critic network is a four layer fully-connected neural work with hidden layer size of 402 and 300, as the 2-dimensional action is introduced to the network from the second layer. Activation functions are two ReLUs, and there is no activation function after the output layer. Similarly, Weights of nodes are initialized from uniform distributions, with parameters same as proposed in the paper.

We initialize target actor and target critic networks as a copy of actor and critic networks respectively.

We use Adam as optimizer for all networks

1.2 Hyper Parameters

We set learning rate for both actor and critic networks as 10^{-3}, discount factor $\gamma = 0.99$ and batch size is 100. For the replay buffer, the maximum buffer size is 10000 and we initialize the buffer at first by filling 1000 (state, action, reward, next state, done flag) pairs taken from uniform random sampled actions.

When selecting actions, we add a 2-dimensional Gaussian exploration noise with zero mean and $diag(0.1, 0.1)$ covariance.

We train the policy for 200000 steps with non-randomly initialized environment. Then, we evaluate the policy using randomly initialized environment and no exploration noise.

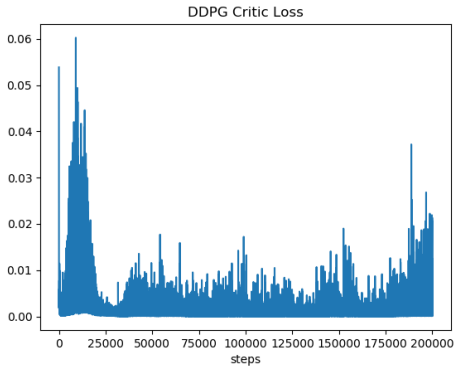
1.3 Result

We train the policy using three different random seed values: $10, 100, 1000$ respectively. The final policies are able to reach the goal in randomly-initialized environments in very few steps. The results are shown in below.

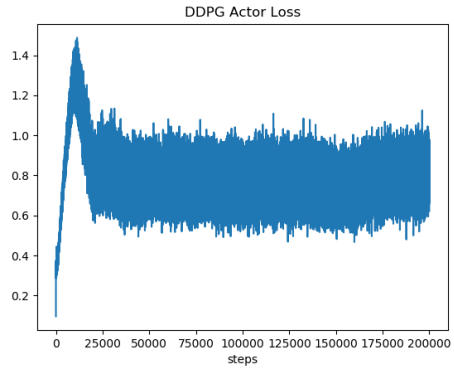
For the result GIF please see attached file.

Critic and Actor Loss Critic and actor loss of those three policies using different random seed values are shown in "Fig. 1", "Fig. 2" and "Fig. 3".

From the loss we can not observe too much information. Nevertheless, the actor loss shows convergence after about ten thousands steps, which means that over policy is being optimized.

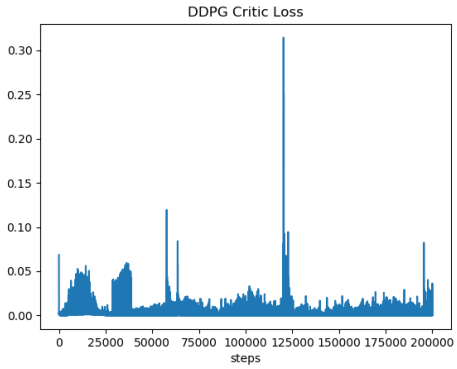


(a) DDPG Critic Loss

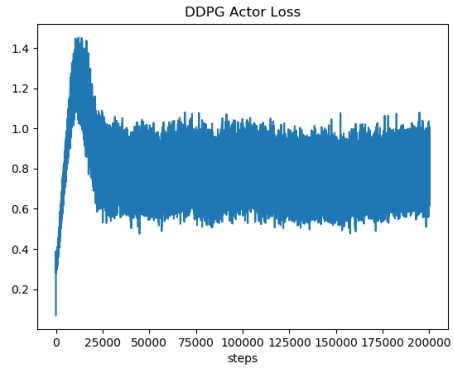


(b) DDPG Actor Loss

Figure 1: Random Seed 10

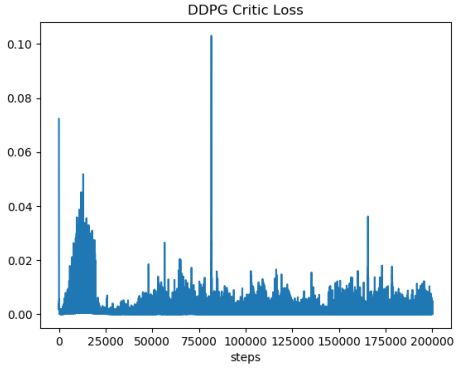


(a) DDPG Critic Loss

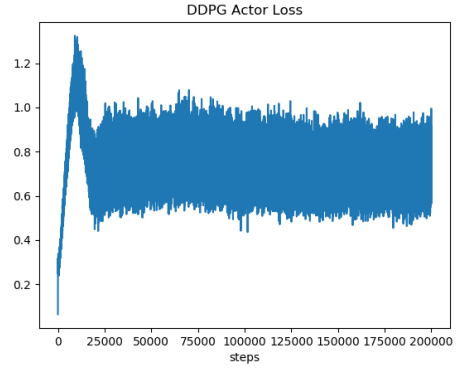


(b) DDPG Actor Loss

Figure 2: Random Seed 100



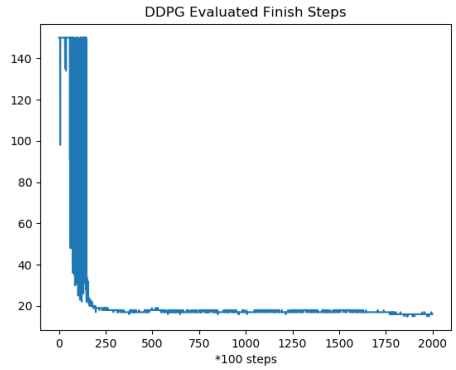
(a) DDPG Critic Loss



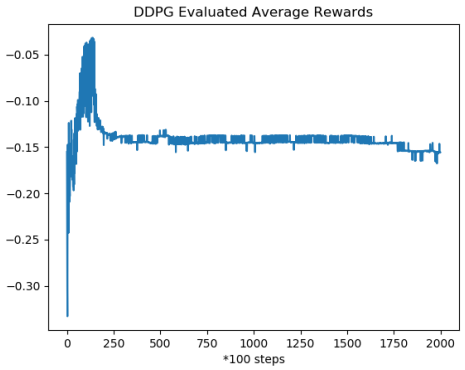
(b) DDPG Actor Loss

Figure 3: Random Seed 1000

Finish Steps and Average Rewards During Training An intuitive way to visualize our training progress is to evaluate our policy after very 100 steps, and check the finish steps and average rewards of each policy evaluation. The results are shown in "Fig. 4", "Fig. 5" and "Fig. 6".



(a) DDPG Finish Steps



(b) DDPG Average Rewards

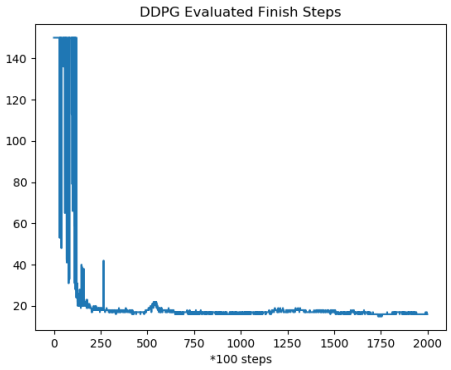
Figure 4: Random Seed 10

From the results we can see after around 15k steps, our policy is good enough to reach the goal in under 20 steps, regardless of the random seed value. The policy improves dramatically around the steps where the actor loss converges.

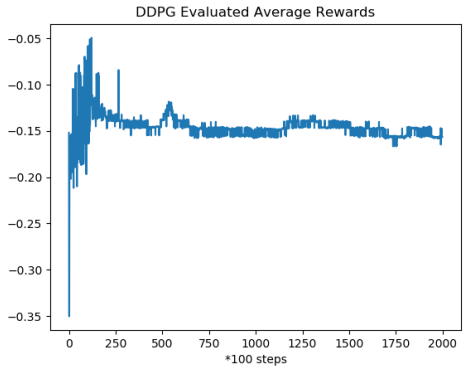
The results show that DDPG is very effective in solving the 2-DOF reacher environment.

Comparison Between Assignment 3 In assignment 3, our policy gradient method can get comparable result using batch size 1000 and 200 iterations at best, which is equivalent to 20k steps in DDPG in terms of samples. DDPG can get similar result using 15k steps, which means the DDPG is more sample-efficient.

Also, the policy gradient method is less stable and sometimes gets inconsistent result. Sometimes we

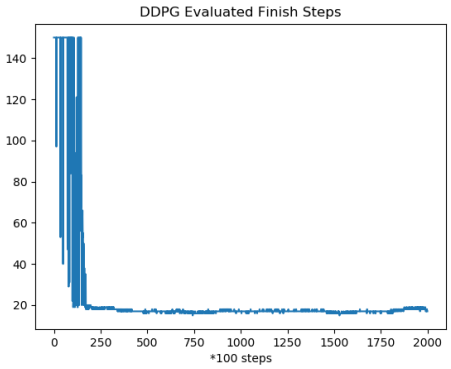


(a) DDPG Finish Steps

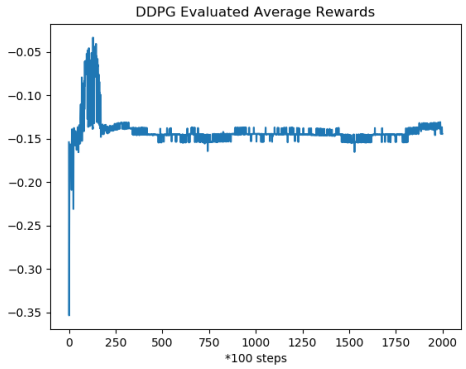


(b) DDPG Average Rewards

Figure 5: Random Seed 100



(a) DDPG Finish Steps



(b) DDPG Average Rewards

Figure 6: Random Seed 1000

need to increase batch size to 1500 or it needs more iterations to converge. In contrast, DDPG is getting stable result in different random seed values, which proves that the actor and critic network helps for training stability.

2 Question 2 - TD3

In this question, we implemented TD3 to solve the 2-DOF reacher environment. The TD3 parameters are the same as suggested in the original paper [2].

The parameters are similar to DDPG, except in TD3 we utilize a twin critic network to learn the two Q-functions as proposed in the paper. Additionally, the 2-dimensional action is introduced to the critic network from the first layer, instead of second layer as in DDPG. We also has a different target update rate $\tau = 5 \times 10^{-3}$ vs $\tau = 10^{-3}$ in DDPG

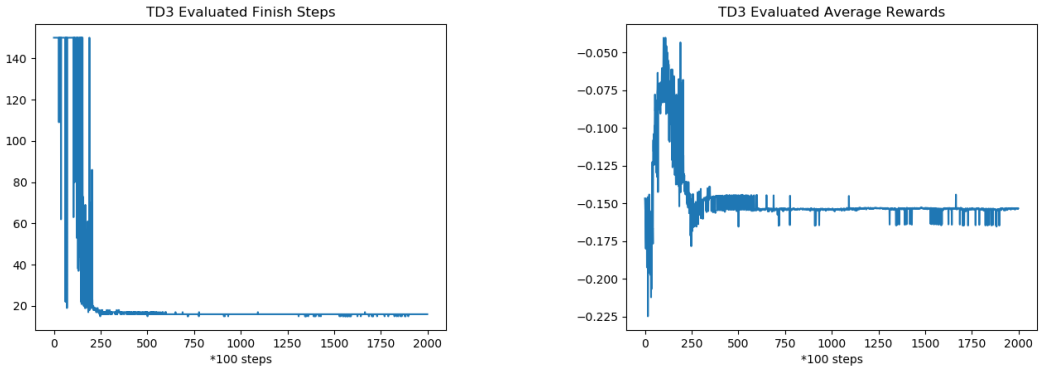
Still, we learn the policy for 200000 steps with non-randomly initialized environment. Then, we evaluate the policy using randomly initialized environment and no exploration noise. Random seed is 1000 in this case.

2.1 Result

Below here shows the results of TD3. Due to the fact that TD3 perform one policy update for every two Q-function updates, and has a different critic network structure, the loss is not comparable with DDPG, hence we do not show the loss here.

The final policy is able to reach the goal in random-initialized environments within very few steps. Please see attached GIF for the result.

Finish Steps and Average Rewards During Training Policy evaluation result during training are shown in "Fig. 7".



(a) TD3 Finish Steps

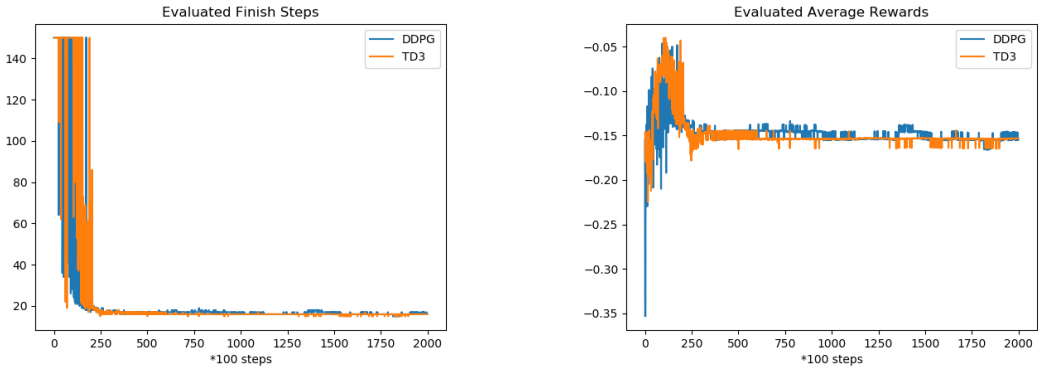
(b) TD3 Average Rewards

Figure 7: Random Seed 1000

Comparison Compared to DDPG, TD3 is supposed to have a smoother and more stable policy update, thanks to its twin-Q function approximation and delayed policy updates. It outruns DDPG in cases where DDPG could overestimate the Q-values.

However, in simpler cases like the 2-DOF reacher environment, DDPG has good performance, while TD3 may learn the policy slower due to its smoothing tricks and slower policy update.

In the comparison below shown in "Fig. 8", we can see DDPG converges a little bit earlier than TD3, but they obtained the same performance after the convergence till the end.



(a) Finish Steps

(b) Average Rewards

Figure 8: Random Seed 1000

References

[1] arXiv:1509.02971 [cs.LG]

[2] arXiv:1802.09477 [cs.AI]