Report

Lifeng Wei

December 2018

1 Introduction

This report summaries my explorations for solving the **Reacher** environment. First it briefly discuss the algorithms applied. Then it shows some of my experimental results for visualization and the effect of some techniques applied in this implement. In the last some possible improvements are also discussed.

The description for the environment can be found in the Readme.md file and will not be repeated here.

2 Algorithm

2.1 DDPG

The full name of DDPG [2] is Deep Deterministic Gradient Policy. You can take it as the generalization of Deep Q-Learning (DQN) [3] in the continuous action space. It is one of the actor-critic method, where the actor choose an action with known state, and the critic predicts the future reward of this action taken by actor. It's assumed that the distribution of actions is a normal distribution (perhaps multivariate), and clipped if the action space is bounded. The actor decides the mean of this normal distribution.

2.1.1 Similarity to DQN

Just like DQN, DDPG also makes use of the replaybuffer from where it samples experience to train the agents. Both local and target networks exist for actor and critic to stabilize the distribution of experience. All these details are already explained when describing DQN. If you want, you can find another report about implementation of DQN in the 'Navigation' repository.

2.1.2 Network Structure

- Actor network
  The structure of actor is exactly the same as mentioned in the original DDPG paper [2]. It uses Relu [4] as activation and Batch normalization

[1] to accelerate training. The structure is:

FC(state size, 400)→Relu→Batch Normalization→FC(400,300)→Relu→Batch Normalization→FC(300, action size)

- Critic network
  Here we tested two different structures of critic. The main difference between them is the place to include action. The structures are:
  1. Structure 1 (activated by 'critic=1')
     FC(state size, 400)→Relu→Batch Normalization→Add action, FC(400+action size, 300)→Relu→FC(300,1)
  2. Structure 2 (activated by 'critic=2')
     FC(state size, 400)→Relu→Batch Normalization→FC(400, 300)→Relu→Batch Normalization→Add action, FC(300+action size,1)

In the experiment I found that the second structure does not learn at all. I guess the reason is that action directly participates in the final prediction of reward. There is not enough interaction and non-linearity relationship between state and action.

2.1.3 Network Updates

Here we illustrate in details how networks are updated in each learning step. First we explain our notations:

- From replaybuffer we get state S_i, action a_i, reward r_i, state after action S'_i. $i = 1, 2, \ldots, N$
- Actor networks $A(s|\theta^A_{local})$, $A(s|\theta^A_{target})$, $Q(s,a|\theta^Q_{local})$, $Q(s,a|\theta^Q_{target})$
- Reward discount rate γ. Soft update parameter τ. Learning rate for local actor η_1. Learning rate for local critic η_2.

First we update the critic network with TD method. You can do TD(λ) but here I just used TD(0).

$$\theta^Q_{local} = \theta^Q_{local} - \frac{1}{N}\nabla_{\theta^Q_{local}} \sum_{i=1}^{N}(r_i + Q(S'_i, A(S'_i|\theta^A_{target})|\theta^Q_{target}) - Q(S_i, a_i|\theta^Q_{local}))$$

$$\theta^A_{local} = \theta^A_{local} + \frac{1}{N}\nabla_{\theta^A_{local}} \sum_{i=1}^{N} Q(S_i, A(S_i|\theta^A_{local})|\theta^Q_{local})$$

After updating these two local networks, we update two target networks:

$$\theta^A_{target} = \tau * \theta^A_{local} + (1-\tau) * \theta^A_{target}$$

$$\theta^Q_{target} = \tau * \theta^Q_{local} + (1-\tau) * \theta^Q_{target}$$

2.1.4 Other details

In practice there are also some other details, including choice of hyperparameters, applied. I list them here:

- Network Initialization: The initialization of networks' weights are not totally random. As suggested in the DDPG paper, all weights in fully connected layers, except for the output layer, is initialized uniformly in interval $(-\frac{1}{\sqrt{f}}, \frac{1}{\sqrt{f}})$. Here f is the input size of the layer. For the output layer, actor's network is uniformly sampled from $(-10^{-3}, 10^{-3})$ while critic's network is from $(-10^{-4}, 10^{-4})$.
- Hyperparameters: $\eta_1 = 0.0001$, $\eta_2 = 0.001$, $\tau = 0.001$, batch size=128, replaybuffer size=1,000,000
- Learning speed: I did not update the networks every timestep. Instead, I updated the networks 10 times every 20 timesteps.
- Gradient Clipping: When the L2-norm of gradient of local critic network is larger than 1, I would shrink it to 1.
- Decreasing level of noise: The level of noise is always getting smaller. At the i^{th} episode, the noise is original noise $\times \frac{1}{\sqrt{i}}$

3 Result

3.1 Solve the environment

Below is the trajectories of episodic reward and average reward of past 100 episodes. You can see the agent learns fast and the average reward never drops below 30. Although the agent will learn some wrong moves, it correct itself quickly.

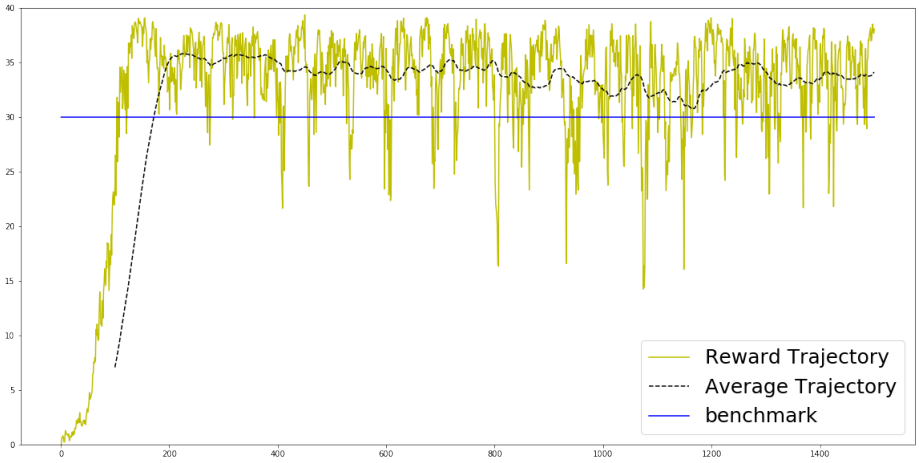

Figure 1: Trajectories of my agent. Solved the environment at episode 172

3.2 Importance of the techniques

In this part we will change some settings to see the functions of some techniques we used here. We can also see what are important for solving this environment.

1. Level of noise: If we keep the level of noise a constant, the training actually fails. This suggests that our original level of noise is too large to get a stable solution. A smaller level from the very beginning might solve this problem with the sacrifice of exploration.

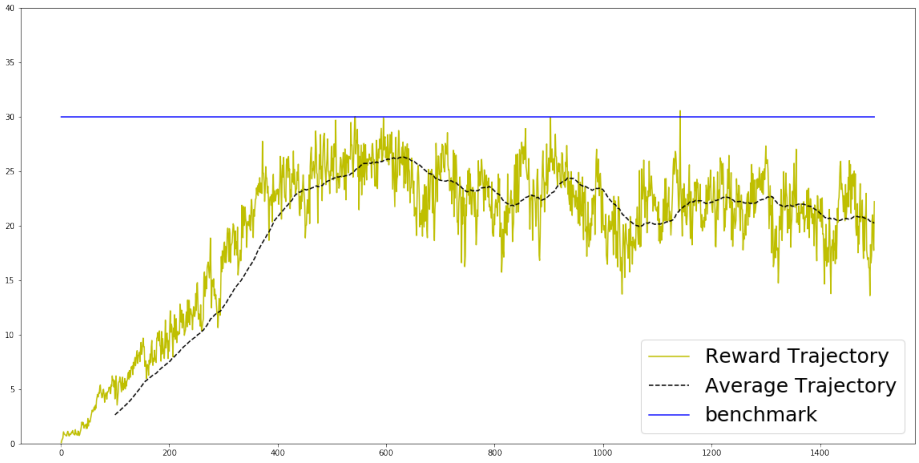

Figure 2: A constant level of noise never solves the environment

2. Batch Normalization: We can see that batch normalization is not neces-

sary to solve this problem, but it does accelerates the process of training. And it seems that as long as one network learns fast and stable, the agent can be trained well. But when both networks are not using batch normalization, the training gets much slower.

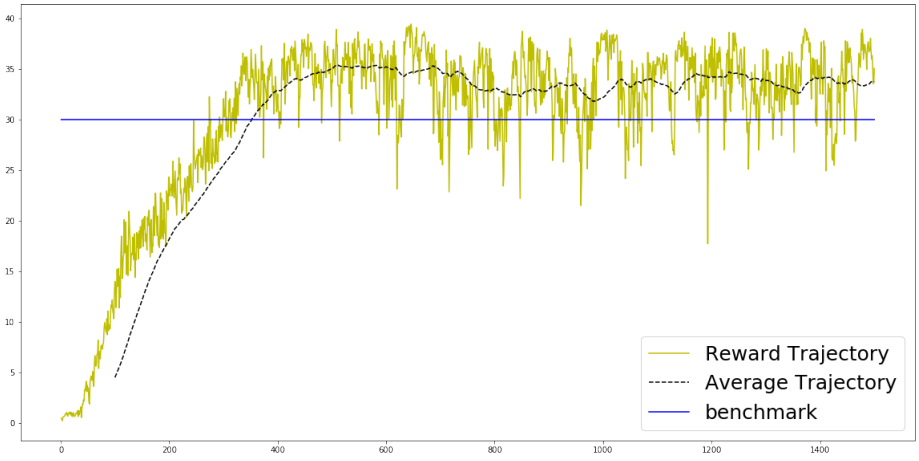

Figure 3: Removing batch normalization in actor requires 353 episodes

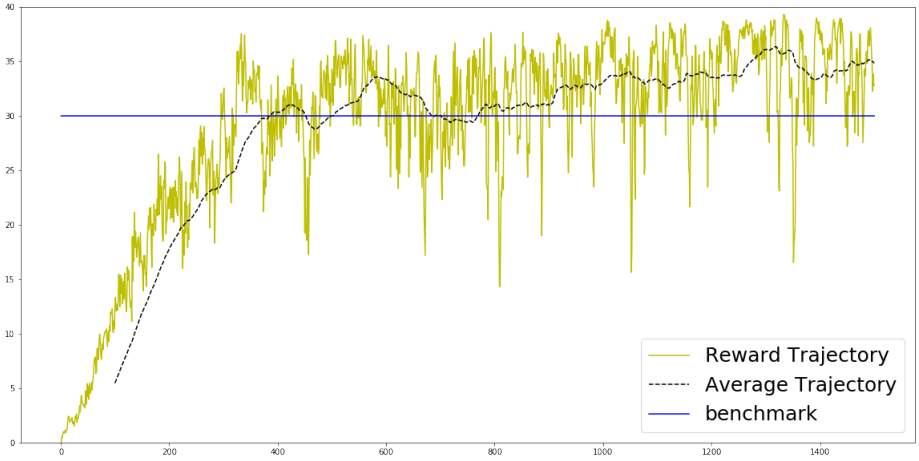

Figure 4: Removing batch normalization in critic requires 385 episodes

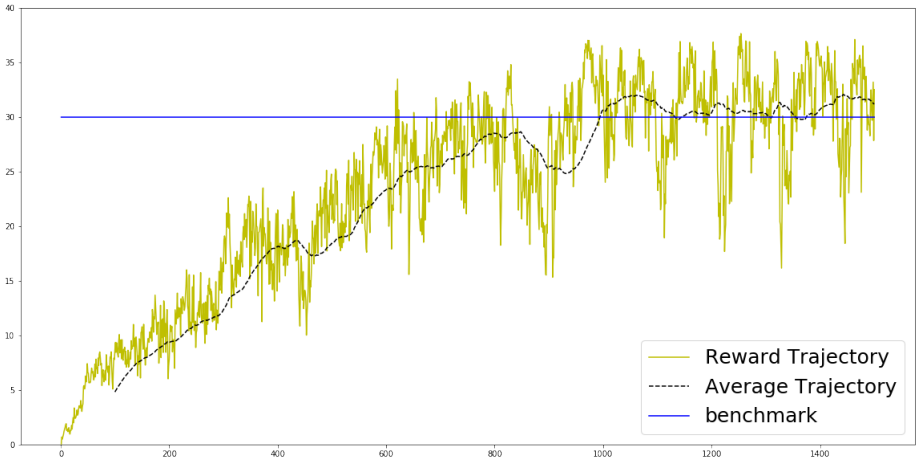

Figure 5: No batch normalization at all requires 995 episodes

3. Batch Size: Below is the plot with a batch size equaling 64. It solves the environment at episode 222. You will find that in the end the rewards get more unstable. Change of batch size has little influence.

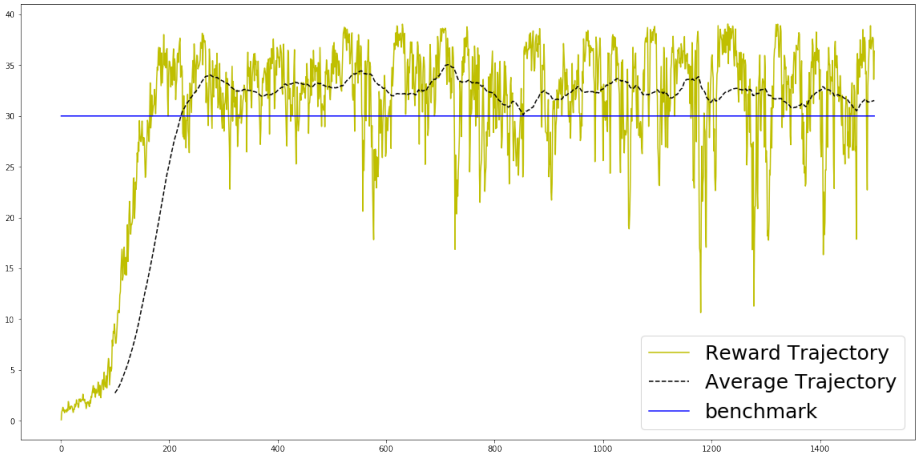

Figure 6: Batch size 64 makes little difference. Successes at episode 222

4. Gradient Clipping: The plot below is the trajectories for training an agent without clip the norm of gradient for critic.

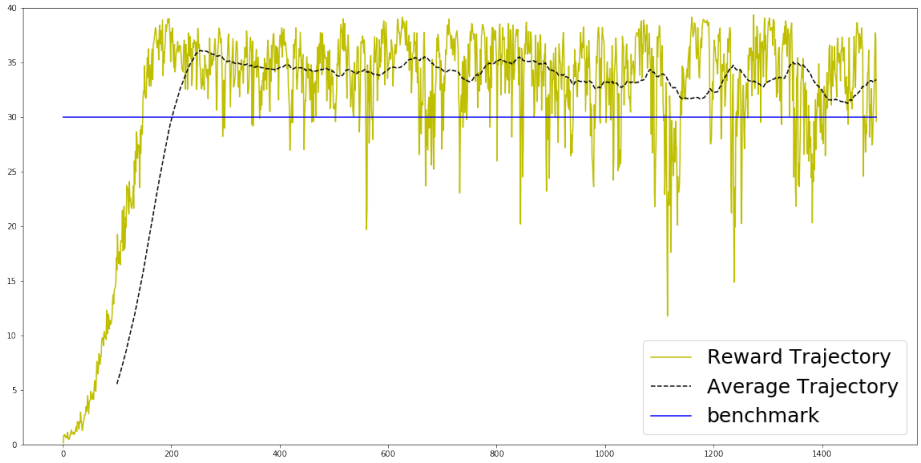

Figure 7: Training without gradient clipping makes almost no obvious difference. Solves environment at episode 202.

5. ReplayBuffer Size: Below I set the size of ReplayBuffer from 1,000,000 to 100,000. This makes the ReplayBuffer remember most recent 5 episode instead of 50. We can see the whole process get less stable. The agent make worse mistakes with a higher frequency.

Figure 8: Remember 5 episodes instead of 50 is a bad idea. Still solves the environment at episode 162.

6. Another critic structure: Below is the plot using the second structure of critic network. We can see the second network structure does not seem to be learning at all. I think the main reason is that the representation

of the network is not enough. There lacks interaction of action and state features.

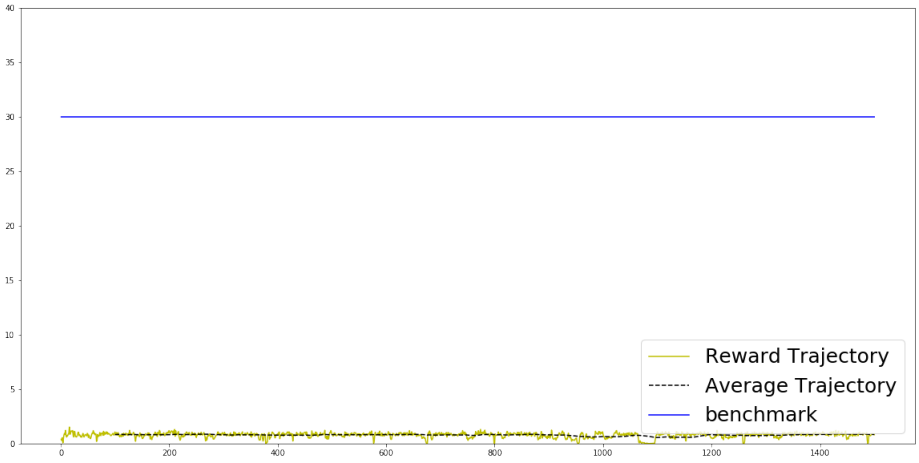

Figure 9: The other structure of critic cannot learn at all.

7. Using one agent to train: Testing. Waiting for results
8. Initialization of networks. Working on it. Waiting for results.

4 Future Work

Of course I will complete the previous chapter once the results are there. Running 1500 episodes is somehow time consuming. Around 8 hours each time.

Future work on this project includes trying PPO, TRPO and A2C methods. I will also try to work on the **Crawler** environment and solve it.

References

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[2] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[4] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.