

Yet Another Continuous Control Agent Implementation

by

Carlos M. MATEO

Abstract: This project implements the method Deep Deterministic Policy Gradient DDPG (an end-to-end method) for resolve the problem of generate a control policy of a continuous control problem. The results show that this implementation using the hyper-parameters summarized in this report obtain an average score of at least 30 points after training during 108 episodes. Finally, it is done a brief discussion about which improvement and future directions can be taken to improve this repo. The use of the associated code is well explained in the README file.

Keywords: Deep Reinforcement Learning; DDPG; Robot Control

Special thanks to Udacity community for helping with this implementation.

1 Method

Model architecture. This repo presents the implementation of a Deep Deterministic Policy Gradient DDPG algorithm to teach an agent to follow a continuous moving target. Concretely, here is implemented an Actor-Critic method, thus two different networks are used. On the one hand, actor network is composed with 2 fully connected layers with 128 units and 64 respectively. Plus an output fully connected layer with 4 unit for coding 4 angular velocities of the robot. On the other hand, the critic network is composed by 3 fully connected layers of 124 (128 minus action dimension), 64 and 16 respectively. And also an output fully connected layer of 1 unit representing action value function (Q-function). While after each hidden layer is used a rectified linear function in both networks for clipping negative values. Actor network output layer triggers an hyperbolic tangent function to clip output values in the range of $[-1, 1]$. To prevent overfitting is used dropout after each hidden layer in both networks and batch normalization.

While the error function used with the critic function approximation is the mean squared, the actor deep networks used the negative mean. Here, ADAMS method is used for optimize the network parameters, with different DN learning rates. To prevent that oscillations during training and ensure its convergence, I decided to use a clipping strategy after each backward propagation.

Double DQN. As it was discussed in [2] works, use a double network strategy, where one is updated each step and the other is used as a target, is mandatory for achieving stability during RL when is used a function approximation. Double

Table 1: Hyper-parameter configuration

Hyper-parameter	Value	Comments
buffer size	1000000	Replay buffer size.
batch size	128	Minibatch size.
γ	0.99	Discount rate.
τ	0.001	Proportional increment used for soft updating target parameter.
actor learning rate	0.0005	Learning rate used in actor ADAM optimization.
critic learning rate	0.001	Learning rate used in critic ADAM optimization.
C	1	Every 1 step is updated local parameters.
Clipping	1	Gradient clipping.

DQN [3] keeps update this two equal deep q-networks DQN (for critic as well as actor network), a local and a target one, with different weights and at different rates. But, here instead to update target DQN each C steps, I decided to use a soft update strategy after each step, merging τ -times the new parameter values. Note that τ is bounded between 0 and 1. The parameter parameters are softly update according with,

$$\theta_{target} = \alpha * \theta_{local} + (1 - \alpha) * \theta_{target}$$

Experience Replay. Mnih et al. in [2] postulate the needed of using experience replay for improving learning stability. Here, it is used an uniform sampling strategy for selecting experience sample from replay buffer.

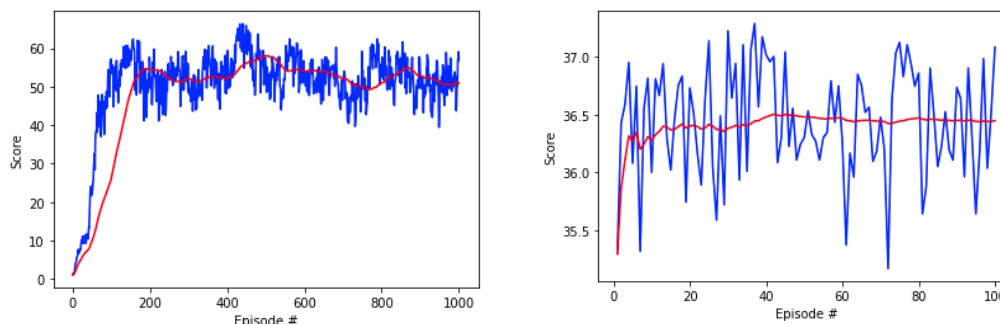
Exploration strategy. As a difference with respect to discrete action space problem, use ϵ -greedy policies in continuous action space problems is not possible. Instead, to solve here the problem of exploration, it is used Ornstein-Unlenbeck process to generate noise over the predicted action values.

2 Experiments

Training Details Here, it is used a multi environment set up with 20 random instances of the problem. Each instance has its own random seed to enhance the domain problem exploration (noise generation) during the training process. Instead of using an strategy of early stop when score reach a desired level (30 points over last 100 episodes), I decided to fix a constant number of episodes (1000 episodes). This strategy let us observe if the learning does not drop the performance after reaching the desired level which really often happened. Moreover, to increase the number of good samples each episode has a fixed number of steps (in my implementation 2000) instead of stopping after the agent is done in any environment instance. The rest of hyper-parameters are summarize in table 1.

Evaluation Details. The agent is executed during 100 episodes using an greedy policy without noise in action prediction. The cumulative discounted reward is averaged over the last 100 episodes, reaching an average of 36.45, a maximum score of 37.29 and a minimum of 35.17. In all cases always I get score values bigger than 30. In addition, say that instead of fixing the number of step per episode, during evaluation an episode finish when any of the instances is done.

Figure 1: Training and evaluation results. This figure presents training plot (left) and evaluation results (right). While the blue curve is current score, red line is the average score during last 100 episodes. Training reach 30 points (averaged during last 100 epi.) in the 108 episode.



3 Discussions and Conclusions

Results show that this method reach an average of cumulative reward of at least 30 points. Also an interesting behaviour observed during the training is that the convergence of this method is as faster as satisfactory examples is experimented during the exploration. This leads to think that a clever solution is to fix the big number of steps for episode, because this increase the chances of reach desired actions. Figure 1 shows that the training after reaching 50 points keeps constant score values without dropping performance during time. The evaluation presents a pretty constant behaviour having values around 36 points.

Some interesting improves that can be contemplated for this task are the implementation of prioritised experience replay as well as the dueling DQN for the critic network. Also, the implementation of Rainbow [1] which extend Actor critic strategies is another interesting improvement. Add LSTM layer to the network architecture is also considered for future improvements. Because, I have shown here that this problem is resolve using sequential input data, LSTM exploit time sequential input data.

References

- [1] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improve-

ments in deep reinforcement learning. *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3215–3222, 2018.

- [2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [3] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double Q-Learning. In *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pages 2094–2100, 2016.