Noname manuscript No.
(will be inserted by the editor)

Technical Report: Implementation of DDPG and MPO on virtual and existing environments

Magnetic Levitation, Furuta Pendulum and Ball Balancer

Luca *Hier könnte ihre Werbung stehen* Dziarski · Theo Gruner

Received: date / Accepted: date

Abstract The model-free algorithms *Deep Deterministic Policy Gradient* (DDPG) and *Maximum A Posteriori Policy Optimization* (MPO) have been implemented and tested on a variety of tasks. First they were tested in an virtual environment after which they were evaluated using real robots. This paper contains the implementation details, specific variations from the original algorithms and a detailed evaluation section. Evaluations have been carried out with focus on the hyper-parameter settings for each individual algorithm and experiment. This paper supplements the reports of many other research groups at the TU Darmstadt to found an extensive summary of RL algorithms on a variety of popular RL environments.

Keywords Deep Reinforcement Learning · DDPG · MPO

1 Introduction

- General:
  - Robot learning
  - curse of high dimensionality
  - difficulties of
- continuous action and state-space
  -
- model free algorithms
- advantage of DDPG [2] and MPO [1] briefly

Jan Peters
first address
Tel.: +123-45-678910
Fax: +123-45-678910
E-mail: fauthor@example.com

S. Author
second address

2 Explanation of the used Algorithms

The reviewed algorithms are model-free and off-policy.

2.1 Deep Deterministic Policy Gradient

DDPG [2] is a policy search algorithm which optimizes the policy based on gradient descent. It is assumed that the gradient of the policy gradient and the gradient of the objective function $\mathcal{J}(\theta)$ w.r.t. the policy parameters θ are in the same direction which results in following policy update.

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta \mathcal{J}(\theta) \tag{1}$$

The evaluation of $\nabla_\theta \mathcal{J}(\theta)$ is based on the Deterministic Policy Gradient [5].

$$\nabla_\theta \mathcal{J}(\theta) = \mathbb{E}_{s\sim\mu^\pi(s)} \left[\nabla_\theta Q(s,a)|_{a=\pi(s)} \right] \tag{2}$$

The Q-function needed for the calculation of the policy gradient is represented by a neural network. The parameterized policy is also approximated by neural networks. In order to fit the parameterized Q-function to the true one, the Q-function is improved by minimizing the TD-difference between the expected Q-function $Q(s_t, a_t)$ in state s_t and the reward $r(s_t, a_t) + Q(s_{t+1}, a_{t+1})$

$$\min_\phi L = \min_\phi \mathbb{E}_{s\sim\mu^\pi(s)} \left[\left(Q^\phi(s_t, a_t) - r(s_t, a_t) + Q^{\phi'}(s_{t+1}, a_{t+1}) \right)^2 \right] \tag{3}$$

The expectation can be approximated by sampling random tuples of $(s_t, a_t, r(s_t, a_t), s_{t+1})$. In order to be off-policy, a Gaussian noise is added to the deterministic policy. As will be shown in the later sections, choosing an appropriate noise is key for the performance of the algorithm.

Useful Hyperparameter Tuning: DDPG lacks the stability of robust Trust-Region algorithms and is heavily dependent on the hyperparameter settings. We focused primarily on the usage of different noises and on the batch size. The default noise used in the original paper was an Ornstein-Uhlenback (OU) noise, but recent advances favor the Adaptive-Parameter noise [4]. The first adds noise to the action while the latter adds noise to the parameters of a perturbed actor.
Another important hyperparameter tends to be the mini-batch size. In order to estimate the mean the batch-size should grow for more complex environments.

2.2 Maximum A Posteriori Policy Optimization

Big disadvantages of off-policy algorithms, like DDPG, are that convergence is not guaranteed, especially for high dimensional problems. This is due to an information loss [3] occurring when the difference between the old and updated policy is too large. On-policy algorithms, like REPS or PPO, cope with

this problem by introducing an additional Kullback-Leibler constraint which regularizes the policy update w.r.t the old policy. On-policy algorithms tend to converge more consistently than off-policy algorithm but are instead data inefficient since the sampled data can only be used once.
MPO [1] uses the KL constraint but is still and off-policy algorithm which leads to an algorithm which is supposed to converge fast with high data efficiency. The optimization is split into three distinct steps. First, the Q-function is updated, then the dual function of the constraint optimization problem is solved which is similar to REPS and in the third step the policy is updated. Except for the hyperparameters, the evaluation of the Q-function seemed to be to be crucial.
Concerning MPO, one can choose between a parametric or non-parametric additional q-function which perform differently. Both ways have been implemented in the context of this work.

- Briefly explain background of algorithm
- Difficulties in implementation
-

3 Environments used for Evaluation

The algorithms are evaluated on three different environments which all represent a balancing task in which the desired state is an unstable equilibrium. The considered experiments are *Magnetic Levitation*, *Ball Balancer* and *Furuta Pendulum*. All environments are characterized by a state s. For each state s, the agent can choose an action and receives a reward from the environment. Since the algorithm can be implemented in a continuous environment, state and action space are continuous.

3.1 Magnetic Levitation

The experiments are ordered based on their complexity. For Magnetic Levitation, a ball is placed in the middle of a magnetic field which is generated by an electromagnet. By controlling the coil voltage, the direction and strength of the magnetic field can be regulated which applies a force onto the ball. The desired task is to minimize the deflection of the ball from its neutral position while using the least amount of current to fulfill the task. Therefore, the states are sufficiently described by a distance measure in the vertical direction and the current of the coil.

3.2 Furuta Pendulum

The pendulum consists of two connected arms. The first one is attached to an motor at its end and can rotates around the vertical axis. The end of the first

Table 1 General evaluation criteria

	DDPG	MPO
episode length	300	3000
Actor net	$400-300$	$100-100$
Critic net	$400-300$	$200-200$
mini-batch size	64	64
additional actions	-	64
γ	0.99	0.99
learning rate	0.0005	0.0005
τ	0.001	-
ε	-	0.1
ε_μ	-	$5 \cdot 10^{-4}$
ε_Σ	-	10^{-5}

and second arm are aligned perpendicularly. The task is to balance the second arm upright by choosing the right momentum. The system is fully described by the angles of the arm and pole as well as the angular velocity of the arm. For numerical reasons, the sines and cosines of the respective angles are stored resulting in 6 observed states.

3.3 Ball Balancer

observation space is 8 action space is 2

4 Evaluation

The following section summarizes the performance of the previously described algorithms (section 2) for the three environments. Thereby, the choice of parameters as well as the difference between the evaluation on the virtual and real environment will be of importance. The evaluation procedure in this section is as follows:
First, the learning procedure as well as the evaluation is carried out in the virtual environment which serves as a baseline. *Secondly*, the learned models will be evaluated on the real robots. *Thirdly*, learning and evaluation will be carried out on the real robot. The following table represents hyper-parameter settings which are constant for all environments:

4.1 Magnetic Levitation

4.2 Furuta Pendulum

4.3 Ball Balancer

4.4 Evaluation of the algorithms trained in simulation

Training results as well as evaluation results on virtual environment

4.5 Performance of the learned algorithms on the real robots

Using the learned model from the virtual environment on the real robots

4.6 Training and Evaluation on the real robots

Learning and evaluation on real robot

5 Conclusion

Acknowledgements I am tremendously grateful for my always loving mother, Renate the *GRANATE*. Without her I wouldn't be where I am now (pun intended). Special thanks to the best imaginable flat-sharing community, *WG DA ihr Hurn*, who always believed in me and pushed me to my absolute limits. $\sim$ *Luca Dziarski*

References

1. Abdolmaleki, A., Springenberg, J.T., Tassa, Y., Munos, R., Heess, N., Riedmiller, M.: Maximum a Posteriori Policy Optimisation (2018). URL http://arxiv.org/abs/1806.06920
2. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning (2015). DOI 10.1561/2200000006. URL http://arxiv.org/abs/1509.02971
3. Peters, J., Katharina, M., Alt, Y.: Relative Entropy Policy Search the KL in the other (2008)
4. Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R.Y., Chen, X., Asfour, T., Abbeel, P., Andrychowicz, M.: Parameter Space Noise for Exploration pp. 1–18 (2017). DOI 10.1063/1.334137. URL http://arxiv.org/abs/1706.01905
5. Silver, D., Lever, G., Technologies, D., Lever, G.U.Y., Ac, U.C.L.: Silver14 pp. 1–9 (2014). URL papers://d471b97a-e92c-44c2-8562-4efc271c8c1b/Paper/p652

Author, Article title, Journal, Volume, page numbers (year)