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

Application of Reinforcement Learning Methods

Group 19 - Final Project Report

Yannik Frisch · Tabea Wilke ·
Maximilian Gehrke

the date of receipt and acceptance should be inserted later

1 Introduction

We shortly present two state-of the art reinforcement learning algorithms, the *Deep Deterministic Policy Gradient* and the *Natural Actor Critic*. Both algorithms are evaluated on the Quanser Robots platforms on the simulated quanser systems platforms *BallBalancerSim-v0*, *CartPoleSwingShort-v0* and *Qube-v0*. We furthermore present the results of training both algorithms on the *BallBalancerSim-v0* and evaluating it on the pyhiscal *BallBalancerRR-v0* platform. Finally, we let a pretrained *Natural Actor Critic* agent continue learning on the physical version of *CartPoleSwingShort-v0*. We close with a discussion of the results.

Address(es) of author(s) should be given

2 Deep Deterministic Policy Gradient

The *Deep Deterministic Policy Gradient* approach [1] is an application of the *Deep Q-Learning* algorithm [2] to actor-critic methods [x] in combination with the *Deterministic Policy Gradient* [3]. It is a model-free and off-policy algorithm, learning a deterministic policy. A replay buffer with a total size of 1e6 samples is used to sample independently and identically distributed mini-batches, randomly selected to temporarily decorrelate them. Target networks are used, which are constrained to slow changes to improve the stability of learning.

2.1 Evaluation on BallBalancerSim-v0

The Quanser Robots *BallBalancerSim-v0* environment consist of a plate whose angles can be controlled by the input actions. The goal is to balance a ball on the plate, receiving a maximum reward of [?] per time-step for balancing it in the middle of the plate. The environment ends after a maximum of 1000 time-steps.
We started our evaluations with using the same network structures for the actor and critic as [x] did. We used 2 hidden layers with 100 and 300 hidden neurons for the actor and the critic networks and their targets. The learning rates are also set to $\alpha_{actor} = 1e-4$ and $\alpha_{critic} = 1e-3$. In figure [x] one can find our first acceptable results. The discounting is set to $\gamma = 0.99$, we did small target updates ($\tau = 1e-4$), used a mini-batch size of 64 and a total replay buffer size of 1e6. We slightly increased the noise to $\sigma_{OU} = 0.2$ and $\theta_{OU} = 0.25$ as the environments action space has an higher amplitude compared to the *Pendulum-v0*.

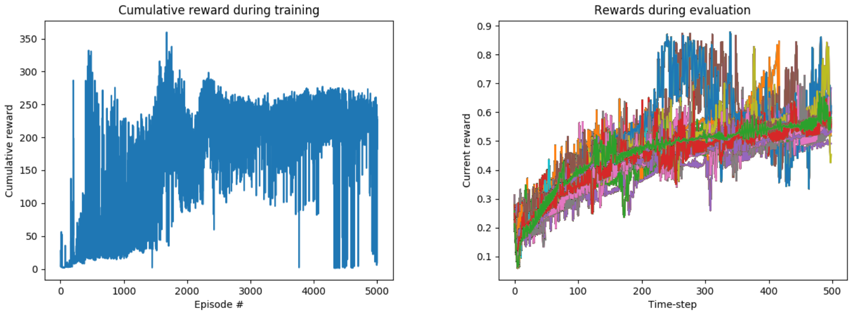

Fig. 1 The left figure shows the cumulative reward per episode during the training process. The right one displays current reward per time-step for 25 episodes of evaluation [EDIT: DO 100 EPISODES!]

The algorithm did learn to balance the ball, but was not very stable, which can also be read from the learning process plot. To further increase the sta-

bility, we increased th mini-batch size used to sample from the replay buffer, and reduced the noise again. Using weight regularization did not seem to be helpful, so we set it to zero.
Figure [x] shows the training process where discounting is set to $\gamma = 0.2$ compared to $\gamma = 0.99$. One can see discounting is crucial to solve this environment.

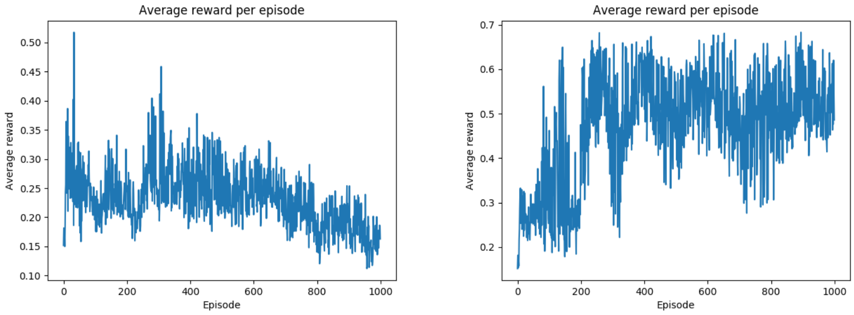

Fig. 2 The left figure shows the cumulative reward per episode during the training process with γ set to 0.2. The right one displays the process for $\gamma = 0.99$. Using discounting close to 1 was very important.

We tried to reduce the computational effort by only using a single hidden layer with 100 hidden neurons instead of two layers. The impact on the performance is shown in figure 3. The learning suffered from instabilities, so we decided to weight the stability of using two hidden layers higher than the performance loss.

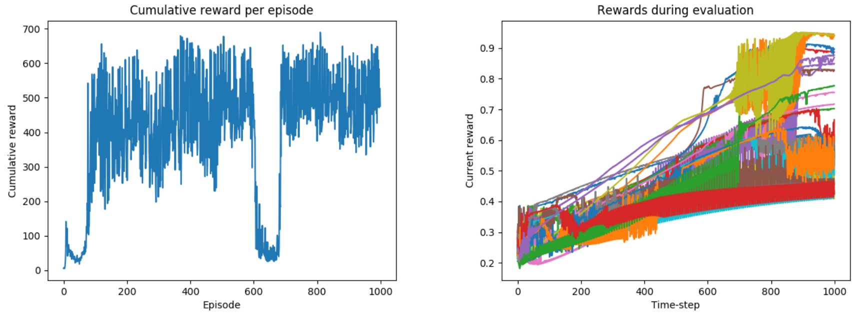

Fig. 3 The left figure shows the cumulative reward per episode during the training process using only a hidden layer and the right one again displays the performance during evaluation.

Our best results can be found in figure 4 where we set the OU action noise equal to the one used in the original paper with $\sigma_{OU} = 0.15$ and $\theta_{OU} = 0.2$.

We used slightly harder updates with $\tau = 1e - 3$, and achieved an average cumulative reward of about 650 for 25 episodes of evaluation. The learning process took about 3 hours for 1000 episodes.

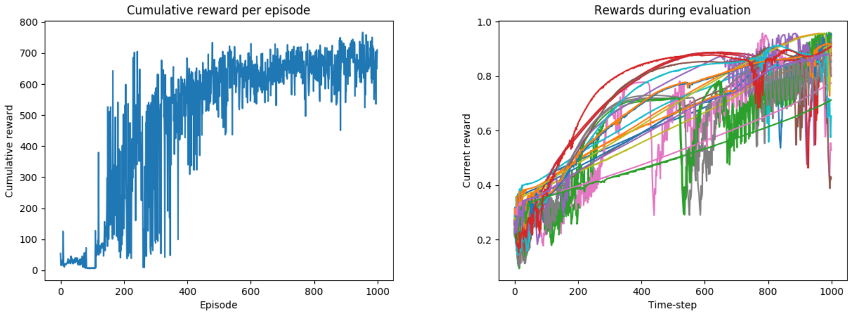

Fig. 4 The left figure shows the cumulative reward per episode during the training process. The right one displays the current reward per time-step for every evaluation episode.

2.2 Evaluation of the Pretrained Model on the Real Ball Balancer System

Evaluating our best model trained in simulation on the real ball balancer was not successful. The chosen actions were too big and the plate tilted. Also, it was not possible for us to reset the environment between the evaluation episodes. The result of a single episode of evaluation can be found in figure [x].

2.3 Evaluation on CartPoleSwingShort-v0

The Quanser Robots *CartPoleSwingShort-v0* environment consists of a movable car with a singular pendulum attached. The car can be controlled by input actions. The reward is depending on the angle, with a max reward of 2 per time-step for balancing the pendulum straight upright.
We achieved some progress using [x]. The results are displayed in [x].

2.4 Evaluation on Qube-v0

The Quanser Robots *Qube-v0* environment implements the Furuta Pendulum, which consists of a motor controlling one horizontal arm. One end of the joint is attached to the motor, the other end is attached to another vertical arm, which can only be controlled indirectly by controlling the first arm. See [x] for more details.
The goal is to balance the second arm in upright position, receiving a maximum reward of 0.02 per time-step. The environment stops after 300 time-steps. We did not get any useful results re-using the parameters we found

for *BallBalancer-v0*. Using an Ornstein Uhlenbeck seemed not to help to deal with with the local optimas of the environment. Setting the randomness σ_{OU} to small resulted in to less exploration. Choosing a higher σ_{OU} resulted in better exploration but much less stability. Examples are displayed in figure 5

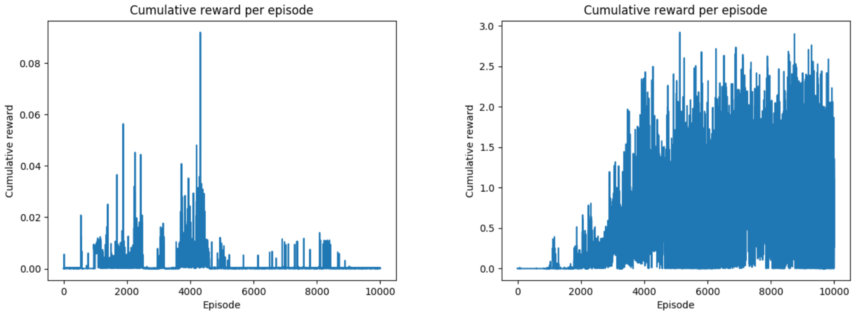

Fig. 5 The left plot shows the cumulative reward per episode during training with $\sigma_{OU} = 0.2$, the right one with $\sigma_{OU} = 2.2$. The other parameters were equal to the ones achieving our best results on *BalllBalancerSim-v0*.

No stable training process could be achieved. Instead we started using a consistent gaussian noise. Unfortunately this did also not help the training process, and seemed to be even more unstable. Figure 6 shows a result of using gaussian noise. The algorithm seemed to learn well for a period of time, but tended to overwrite these progress. Lowering the target update rate τ did not help, nether did lowering the actor and critic learning rates.

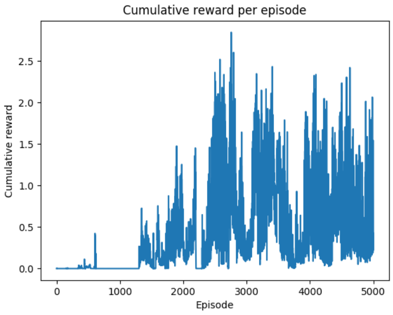

Fig. 6 X

Further evaluations are needed to find the right set of hyper-parameters for this environment.

3 Natural Actor Critic

3.1 Evaluation on CartPoleSwingShort-v0

3.2 Evaluation on the BallBalancerSim-v0

3.3 Evaluation of the Pretrained Model on the Real Ball Balancer System

3.3.1 Learning from the Physical Cart Pole System with pretrained parameters

4 Discussion

References

1. Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2015) Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971
2. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning. arXiv preprint arXiv:13125602
3. Silver D, Lever G, Heess N, Degris T, Wierstra D, Riedmiller M (2014) Deterministic policy gradient algorithms. In: ICML