Reinforcement Learning Approaches for the Swing-Up and Stabilization of the Cart Pole

Antonius Hohenhövel[1], Steffen Borchers-Tigasson[1]

[1] Hochschule für Technik und Wirtschaft Berlin
Wilhelminenhofstraße 75A, 12459 Berlin
E-Mail: steffen.borchers@htw-berlin.de

1 Introduction

In many applications, from architecture to robotics, reinforcement learning approaches for stabilizing, controlling, and optimizing systems have been considered as an alternative to classic control methods. Advantageously, reinforcement learning is a quite flexible framework for numerous control problems, allows including optimality conditions, and prior knowledge if available.

Treating the control problem in the formalism of Markov decision processes (MDPs), strong convergence results on optimality are available for reinforcement learning algorithms (see e.g.[11, 4]) thus making reinforcement learning algorithms attractive.

Even if significant progress has been achieved in the field of reinforcement learning over the past decade (see e.g. [7, 9]), solving real world control problems for nonlinear systems using reinforcement learning is still a very challenging and difficult problem. It is well known that reinforcement learning approaches are subjected to the curse of dimensionality, see e.g. [8]. Thus, for practical solutions of possibly nonlinear systems, it is required to find reasonable approximations instead of the exact solution of the underlying Hamilton-Jacobi-Bellmann equation.

In this contribution we apply and compare reinforcement learning approaches for the swing-up of the nonlinear cart-pole problem considering disturbances.

DOI: 10.58895/ksp/1000124139-8 erschienen in:
Proceedings – 30. Workshop Computational Intelligence: Berlin, 26. - 27. November 2020
DOI: 10.58895/ksp/1000124139 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000124139/

To this end, we use a classic agent-environment scheme (see e.g. [7]). The system, and in particular the swing-up process, is available as nonlinear ODE model derived from Lagrange formalism, which allows for efficient simulation. Particularly, we compare dynamic programming and temporal difference learning for the swing-up of the cart-pole system. We evaluate the influence of the most important metaparameters (e.g. learning rate), as well es the granularity of the discretization on the learning process.

We show applicability of reinforcement learning to the studied problem and discuss practical issues of considered approaches. In addition, we propose a strategy based on reward scheduling for solving advanced control problems, and discuss adaptive discretization as a tool to trade-off computational efforts and accuracy. Although we show that a simulation model speeds up the learning process significantly, the approaches studied here can be applied to a hardware only setup.

The paper is structured as follows: In Section 2, the studied system and swing-up process is described, and in Section 3, the reinforcement learning approach is motivated. In Section 4, the results are presented. The paper concludes with a summary and discussion in Section 5.

2 Studied system: Swing-up of the Cart-Pole

2.1 Model

As model, we consider the classic cart-pole inverted pendulum, derived from Langrangian mechanics. A scheme of the system is depicted in Figure 1, and is decribed by the ODE system:

$$\dot{x}_1 = x_2 \tag{1}$$

$$\dot{x}_2 = \frac{-m_2 l \sin x_3 x_4^2 + u + m_2 g \cos x_3 \sin x_3}{m_1 + m_2 - m\cos(x_3)^2} \tag{2}$$

$$\dot{x}_3 = x_4 \tag{3}$$

$$\dot{x}_4 = \frac{-m_2 l \cos x_3 \sin x_3 x_3^2 + u\cos x_3 + m_2 g \sin x_3 + m_1 g \sin x_3}{l(m_1 + m_2 - m\cos(x_3)^2)} \tag{4}$$

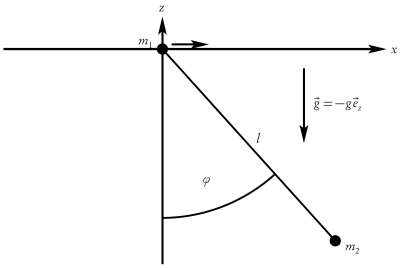

Figure 1: Scheme of the considered Cart-Pole system.

Hereby, x_1 denotes the carts position, x_2 the carts velocity, x_3 the angle φ, and x_4 the angular velocity $\dot{\varphi}$, and finally u denote the systems input in terms of a force applied to the cart in positive x-direction. The input is chosen for simplicity from the binary input set $u \in \mathbb{U} : \{-1, 1\}$.

2.2 System Simulation

In order to apply the reinforcement learning algorithm successfully, we need to simulate the system repeatedly. More precisely, we need to compute the future systems states (fixed time horizon $h = 0.2s$) given varying initial conditions and inputs. This is achieved here using the Python-based OpenAI gym engine [3], considering the inverted cart-pole as described in [1], see Figure 1. The OpenAI gym provides an efficient numerical solution as well as a visualisation of the systems dynamics.

2.3 Goal/Reward

Classically, in terms of optimal control, for the described non-linear cart-pole system we aim to find an (optimal) input sequence so as to transfer the system from its the lower, stable equilibrium point ($s^* := \{x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0\}$) toward the upper equilibrium point (i.e. $x_1 = \text{const.},\ x_2 = 0,\ x_3 = 180^\circ,\ x_4 = 0$). Hereby, the horizontal position is constrained throughout the process by $-l \leq x_1 \leq l$.

For the reinforcement framework, it is crucial to define a meaningful reward function, which is granted after each episode (try-out). Generally, the reward will depend on achievements made in an episode (and thus on the decisions made for the input), and is subsequently propagated backwards so as to learn from the current episode.

Here, a successful run (swing-up) is simplified to the condition $175^\circ \leq x_3 \leq 185^\circ$. Note that it is straightforward to add further (and more rigorous) conditions defining success, however for simplicity of presentation this is omitted here. Only in case of success as defined above, the reward $R = 0$ is granted, otherwise $R = -1$.

3 Reinforcement Learning Framework

The reinforcement learning approach considered here is treated as episodic agent-environment interaction scheme as described by [7]. As key algorithms we consider the temporal difference learning, particularly TD(0) 1-step-Q-learning, see e.g. [9]. Figure 2 depicts how the agent (i.e. the AI choosing the current input) interacts with its environment (cart-pole system).

3.1 initialization()

At the beginning of each learning episode we initialize the algorithm by defining the metaparameters (learning rate α, exploitation rate ε, discount factor λ) as well as the simulation constraints (maximum time steps, epsiode number, discretization, ...). Note that we consider a finite uniform state space discretization rather the infinite continuous state space. Particularly, we consider 4800 discrete states.

3.2 run()

With the function run() we start the interaction of the agent with the environment. First, the environment is reset (reset()), and the initial state s^* together

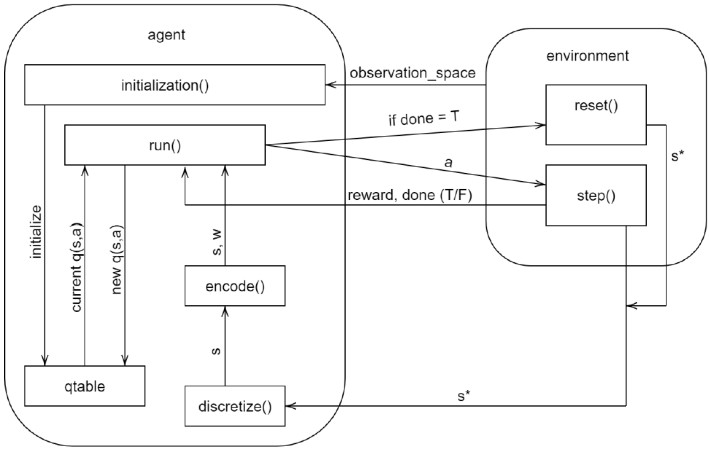

Figure 2: Reinforcement structure scheme.

with an initial input chosen by the agent is propagated to the environment. A simulation step is performed (step()), whereas the time-discretization of (1)–(4) with fixed step size $h = 0.2s$ is considered (discretize()). The function encode() is optional and used in a modified algorithm not further considered here. Next, the current state and the current input is updated, another step is performed and a successor state is reached. The process is repeated until either success, a constraint violation, or the maximum number of time steps is reached.

Of particular interest is the final or terminal state $\mathbf{s}_T$ at the end of an episode, which decides upon the reward granted. Only in case of success a positive reward is granted, in all other cases (e.g. violation of constraints, time's up,...) the reward is negative. Depending on the RL algorithm, at the end of each episode, the reward is back-propagated (new q(s,a)) so as to update the Q-table. Hence it is required to store the visited states within an episode. The here considered TD(0) algorithm however allows to update the Q-table at each step considering bootstrapping.

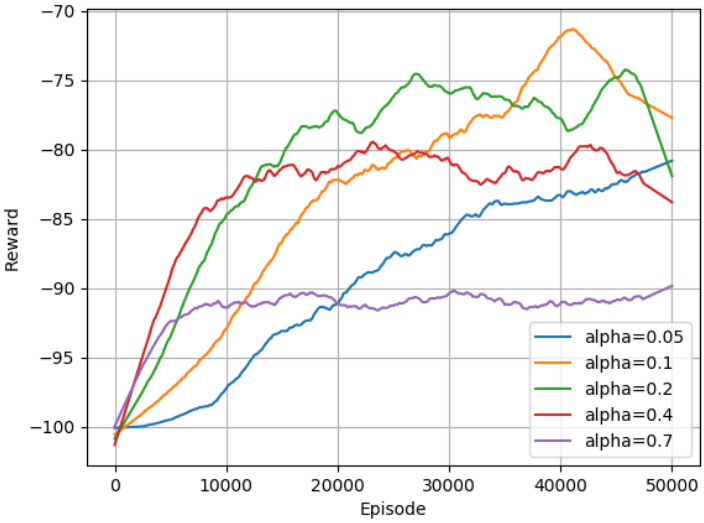

Figure 3: Influence of the learning rate α. Exploration rate $\varepsilon = 0.1$ is kept constant.

4 Results

We successfully applied the TD(0) algorithm to the cart-pole. To evaluate the influence of the learning rate, we evaluated the RL-approach considering different learning rates and compared the results see Fig. 3. Figure 3 shows the influence of the learning rate α onto the average reward given obtained by averaging the reward given five independent runs.

4.1 Influence of the learning rate

As general conclusion, high learning rate is advantageous only at the beginning, in the long run however a lower learning rate leads to better performance. A plausible cause here is the stochastic nature of the process due to the state space discretization. This is because the same action from a certain state may cause different successor states. Starting the process with high learning rate,

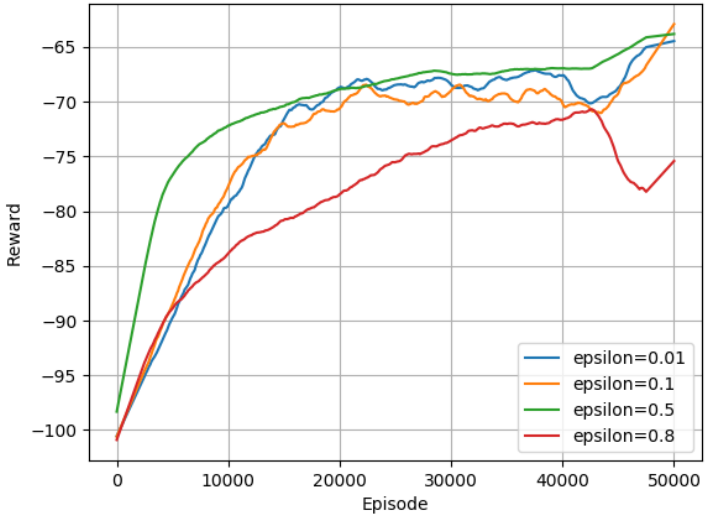

Figure 4: Influence of the exploration rate ε. Learning rate $\alpha = 0.2$ is kept constant.

the Q-table is quickly updated, however gets stuck in local optima. Given a smaller learning rate, the Q-Value of that action at that state changes more slowly thus averaging all possible transitions. This in turn leads, though more slowly, to a more complete overall picture of the swing-up process.

This in turn motivates a modification of the RL algorithm by realizing the learning rate α as a linearly decaying function over time. In the remainder, α decays from 0.5 to 0.01 in the first 25000 episodes.

4.2 Exploration vs exploitation

It is well known that learning following RL strategies depend crucially on the balancing of exploration of the state action space as well as exploiting the results already obtained. Fig. 4 depicts the influence of the exploration rate ε on the learning process. In general terms, while high exploration rates are theoretically advantageous since each state-action pair is visited often, and

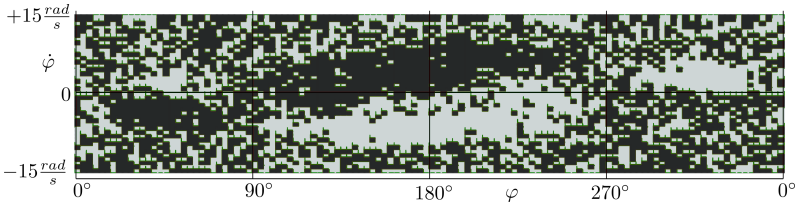

Figure 5: Visualisation of the Q-Table. The carts horizontal position and velocity are omitted. Dark grey corresponds to the action $u = -1$, light grey to $u = +1$.

thus convergence to the optimal policy is guaranteed, in practice however high exploration rates are numerically prohibitive.

Since learning not only depends on rigorous exploration of the state-action space, rather it is significantly boosted when a first success is achieved. So, exploitation is relevant so as to focus on most promising trajectories leading to an early success which can be exploited thereafter. This relation is in general very difficult to translate in reasonable ε parameter values. In our case, a exploration rate of 0.5 is well balanced.

4.3 Visualisation of the policy

The resulting Q-table with the greedy action is shown in Fig. 5. Clearly visible are the expected symmetrical properties. Each pixel corresponds to a discrete state. Omitted are the horizontal position and velocity of the cart for simplicity.

5 Discussion

The here presented RL algorithm is easily applied to four dimensional non-linear dynamic system. We have shown applicability of the algorithm for swing-up process of this cart-pole. The approach can be easily adapted so as to treat a variety of stabilisation goals such as keeping the cart-pole at the upper, instable stationary point. The presented temporal difference algorithm can be applied for this purpose directly, however a functional approach (i.e. learning

the control parameters of a state feedback using reinforcement learning) may be computationally more advantageous.

The presented approach is directly applicable to the hardware only case. However, in this naive setting with not prior knowledge, a rigorous exploration of the state space is mandatory. Thus, a simulation model is an advantage to speed up the training significantly. Prior knowledge can be encoded e.g. within the reward function or by adding additional constraints.

Regarding the computational tractability, discretizing the state space has to be considered carefully. While the available computational power limits the overall number of discrete states, it is advantageous to distribute the discrete states by taking the studied process into consideration. It is advantageous to constrain the state space to an meaningful observable area, so as to subsume large angular and horizontal velocities into a few (forbidden) states and thus reducing complexity.

Furthermore, it is possible to consider an adaptive dicretization scheme, starting with only few discrete states and subsequently splitting those states given an appropriate measure based on the transition probabilities as obtained from the reinforcement learning algorithm. Further research will apply the RL framework to different stabilization problems occurring in production, e.g. adaptive picking of objects with a robot arm, or placing objects in optimal fashion.

References

[1] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, UK, 2004.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAai Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Peter Dayan and Terrence J Sejnowski. Td (λ) converges with probability 1. *Machine Learning*, 14(3):295–301, 1994.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Santanu Pattanayak, Pattanayak, and Suresh John. *Pro Deep Learning with TensorFlow*. Springer, 2017.

[7] Stuart Russell, Peter Norvig. Artificial intelligence: a modern approach, 2002

[8] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Reinforcement learning with soft state aggregation. In *Advances in neural information processing systems*, pages 361–368, 1995.

[9] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 2017.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[11] John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.