Reinforcement Learning of a Pneumatic Robot Arm Controller

Sander Maas S.M.P.Maas@student.tue.nl
Eindhoven University of Technology, the Netherlands

Marco Wiering Marco@cs.uu.nl
Intelligent systems group, Utrecht University, the Netherlands

Boudewijn Verhaar Boudewijn.Verhaar@philips.com
Philips Applied Technologies, the Netherlands

Abstract

We applied Reinforcement Learning (RL) on a real robot arm actuated by two pneumatic artificial muscles that expose a highly non-linear behaviour. To facilitate learning, we developed an empirical model based on real robot observations. Using the learned simulation model, reinforcement learning was able to quickly learn good robot controllers.

1. Introduction

Advances in Artificial Intelligence have drawn the attention of Philips Applied Technologies. In the context of the Home Robotics Project, the merits of RL methods on difficult control problems are evaluated. To gain stronger evidence of their performance, the methods are benchmarked on a real robot arm. This arm is actuated by two pneumatic artificial muscles that expose a highly non-linear behaviour.

The problem is stated as follows. A robot arm with one degree of freedom has been constructed. The arm is actuated by two pneumatic artificial muscles, so-called McKibben actuators. Together they determine the position of the arm. We say the system is *overactuated*. See Figure 1 for a graphical overview.

Muscles. One of the muscles' desirable properties is their analogy with biological muscles, in terms of contraction rate and force. However, the muscles are hard to control, due to their highly non-linear behaviour.

Control problem. The muscles are controlled by adapting the air pressure inside the muscles. Inflation causes the muscle to contract, resulting in a higher pull force. Binary valves allow the controller to change the air pressure. These valves allow three possible actions per muscle: putting air in, letting air out or keep the amount of air unchanged. Since two muscles are controlled simultaneously, this results in *nine possible combined actions* at every timestep.

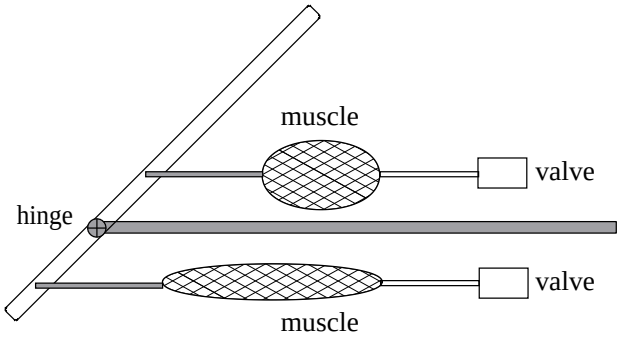

Figure 1. Schematic overview of the robot arm.

The state of the robot arm is monitored by three *sensors*: one pressure sensor per muscle and one potentiometer that measures the arm's angle.

The *goal* of the controller is to bring the arm as fast as possible in the desired position, ending with a velocity near zero.

2. Controller

We use Q-learning (Watkins, 1989) with neural networks as function approximators to solve the control problem. Every 30 ms we read the sensors and let the RL agent choose one of the nine possible actions. An executed action is rewarded if it leads to a goal state, otherwise it is punished. An episode is cut off if the agent does not reach a goal state within 200 actions.

Problems. Applying the RL agent directly to the robot arm proved to be troublesome. We identified three main problems: (1) It takes too much time, (2) The state is not known, and (3) Shaping is necessary.

The state is not known. The most recent sensor values alone do not contain enough information for the agent to base its action decisions on. This is due to the complexity of the system. For example, the air pipes impose a delay on the air flow, leading to pressure drops and pressure build ups. We need to know *how many* sensor values to include in the state.

Shaping is necessary. The eventual goal is to have a fast controller that is able to reach any goal angle, starting from any arbitrary state. Using a trial-and-error approach in a big continuous state space diminishes the probability of reaching a desired goal angle to almost zero. We want a way to relief the problem. For example, start with states close to a goal angle and gradually increase the difficulty. However, such an approach is impossible on a real robot arm since we have no controller yet.

3. Learning from a simulation model

A solution to the above problems is to construct a simulation model and learn by interacting with this model. Designing a model would be an option, however, the behaviour of the McKibben actuators is very difficult to model (Chou & Hannaford, 1996). Moreover, the use of system knowledge is contrary to our research aim. Therefore we focus on learning a simulation model from empirical data (**?**).

General idea. First, we store all experiences with the robot arm in a database. That is, all sequences of observed sensor values and executed actions. Next, we decide how many sensor values are included in the state. Once this is defined, we fit a function on the database. The input set of this function fit consists of all (state, action) pairs and the target set consists of the corresponding sensor values that were observed at the next timestep.

Quality and noise. The fit on the database can be performed by any function approximator. We use a feedforward neural network. By comparing the function fit with the database, we can assess the quality of the fit. The quality analysis shows whether or not the (state, action) pairs contain enough information to enable a good prediction for the next sensor values. Our analysis shows that the recent sensor values alone are not sufficient to enable a good prediction. This is an important result. If a powerful function fitter performs poorly, it is very strong evidence that this state will not enable any RL agent to reliably base its actions on. Extending the state with previous sensor values proved to be the solution.

The insertion of noise to the simulation's predictions might improve the transfer to the real system significantly (Gomez, 2003). In our work we use our analysis to base the amount of noise on.

Shaping. A simulation allows to instantly *select* the agent's state. This makes shaping possible. Initially we choose easy states. Depending on the agent's performance, we gradually confront the agent with more challenging states.

4. Results

The construction of the empirical simulation solved our three main problems. We achieved faster learning rates, a good state representation and it allows for shaping. Our experiments show successful transfer to the real robot arm. The use of the simulation makes the whole learning cyclus extremely fast. It takes two hours to fill the database and fifteen minutes to fit a simulation and train the agent. In our experiments, about 20,000 episodes are necessary to train the agent. The controller typically achieves a goal state within one second on the robot arm.

Of course, there are drawbacks as well. One serious drawback is gathering a representable part of the state space. This is not possible on general problems. Furthermore, this also implies that the method may not be very scalable to higher dimensional state spaces. Future work will adress scalability issues to multiple joints.

References

Atkeson, C., & Schaal, S. (1997). Robot learning from demonstration. *Proceedings of the Fourteenth International Conference on Machine Learning (ICML'97)* (pp. 12–20).

Chou, C.-P., & Hannaford, B. (1996). Measurement and modelling of mckibben pneumatic artificial muscles. *IEEE Transactions on Robotics and Automation*, *12*, 90–102.

Gomez, F. J. (2003). *Robust non-linear control through neuroevolution.* Doctoral dissertation.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* Doctoral dissertation, King's College, Cambridge, England.