BRIDGING THE GAP: LEARNING IN THE ROBOCUP SIMULATION AND MIDSIZE LEAGUE

Thomas Gabel, **Roland Hafner**, **Sascha Lange**, **Martin Lauer**, **Martin Riedmiller**

University of Osnabrück, Institute of Cognitive Science and Institute of Computer Science, 49069 Osnabrück, Germany

Abstract: In this paper, we discuss the application of reinforcement learning for autonomous robots using the RoboCup domain as benchmark. The paper compares successful learning approaches in simulation with learning on real robots and develops methodologies to overcome the additional problems in real world.

Keywords: autonomous mobile robots, reinforcement learning, sensor fusion

1. INTRODUCTION

In autonomous robotics we are often faced with the task to make a robot interact with its environment in a complex way. In many cases, developing explicit control procedures is difficult and time consuming and we are not guaranteed to achieve optimal control.

In this paper, we want to discuss an alternative way of realizing robot behavior using reinforcement learning (Sutton and Barto, 1998). Doing so, we only need to specify a reward function that models whether the agent acts well or not while the robot autonomously learns how to act in order to achieve as much reward as possible. After convergence, the learned policy is optimal in the sense that with no other policy the robot can achieve more reward.

Unfortunately, this theoretical result is based on hard constraints like stochastic independence over time and finite sets of possible situations and actions. In practice on a real robot, reinforcement learning turns out to be a hard job since the theoretical assumptions are not fulfilled and high dimensional state and action spaces have to be dealt with. Moreover, repeating a robot experiment several thousand times is not possible which would be necessary to guarantee convergence.

While reinforcement learning techniques have so far been applied to a large set of artificial benchmarks like pathfinding in a maze, driving artificial cars in a hilly one-dimensional environment or controlling a simulated cart-pole system real-world applications are rare and its difficult to compare learned policies with controllers based on classical engineering. In contrast, *RoboCup* (Kitano *et al.*, 1997) provides a standardized environment where teams with completely different control approaches compete, so that we can directly measure the improvement that can be achieved using learning techniques.

In this paper, we want to present how reinforcement learning can be applied to autonomous robots considering the RoboCup domain as example. We will start with a description of learning in the context of the simulation league in which robots and their environment are simulated and will in the latter show how the experiences from simulation league can be used for omnidirectional real robots in the RoboCup Middle Size League as well and which pre-processing is necessary.

2. LEARNING SKILLS IN SIMULATION

We started using reinforcement learning algorithms to generate optimal skills in 1999 in the RoboCup Simulation League. There, the world is described by a stochastic state-transition system. The agents can chose between a set of parameterized actions and therewith interact with the environment. The system is organized as a cyclic process with a loop time of $100ms$, i.e. the environment takes an action from the agents periodically and provides information about the state of the game.

Hence, the simulation can be described as a Multi-Agent Markov Decision Process (MMDP) (Hu and Wellman, 1998) with a certain set of possible states and actions in which 22 agents act. This principle modeling can be exploited by the learning algorithm.

The reinforcement learning principle is based on the objective of accumulating as much reward as possible over time. The reward is provided if the agent succeeds in performing a given task, e.g. if we want the agent to learn dribbling, reward is provided when the agent controls the ball and negative reward is given whenever it loses the ball.

Using this principle we were able to implement a couple of skills as well as strategy-level behaviors in soccer simulation. As an example, we want to decribe how intercepting a rolling ball was learned (Gabel and Riedmiller, 2006). One fundamental difficulty in intercepting a ball is that the agent must be very precise in its movements. For example, small errors made in initial turn actions may increase the time needed to solve the whole task significantly.

Modeling this problem as a learning task we described the current situation by a six-dimensional state vector containing the velocity of the ball, the velocity of the agent and the position of the ball with respect to the agent's pose. The agent achieves a large positive reward whenever it gets control of the ball while it obtains a negative reward whenever it does not possess the ball.

We used the TD(1) algorithm (Sutton, 1988) to iteratively learn the value function that represents the expected accumulated reward considering a certain state and an optimal behavior in future. Since the value function is a function of the six-dimensional continuous state space an explicit table representation is not possible but we need a function approximator. We investigated several possible techniques like grid based approaches, memory based approaches and approximation based on multi layer perceptrons.

After learning the value function we can directly derive an optimal policy for the agent by choosing

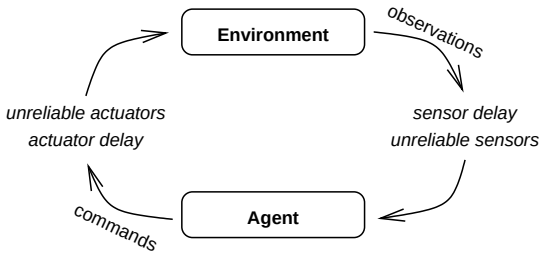

Fig. 1. Adaptation of the classical world-agent interaction loop to the situation with which we are typically faced with in autonomous robotics

the action which leads to the successor state that maximizes the value function. Since the simulation model is known, we can perform this calculation directly.

We evaluated the performance of an agent acting according to the learned policy on a comprehensive set of test situations measuring the average number of steps that was needed by the agent to intercept the ball.

While grid-based and memory-based approximators where only able to learn and represent suboptimal policies the use of neural network approximators resulted in an almost optimal policy with an average performance of 10.57 steps. In comparison, the optimal policy which can be calculated exploiting the known simulation model has an average performance of 9.73 steps.

The learned policy is a rather good approximation to the optimal solution which is typically unknown. Moreover, using techniques like active learning and reward shaping the learned policy can be further improved. In the best case, the learned policy needed on average only 10.01 steps on the set of test situations.

3. CARRYING OVER TO REAL WORLD

Since the soccer simulator is build in an ideal manner like a Markov decision process reinforcement learning can be applied directly. Unfortunately, reality does not behave as nice and therefore additional difficulties occur. A Markov decision process uses a stochastic model of reality: it is a time-discrete system, it assumes stochastic independence over time and it assumes complete knowledge of the world. Additionally, most reinforcement learning algorithms are based on an environment with a small number of distinctive states and actions. All of these assumptions do not hold with real robots.

In contrast, we are faced with a high dimensional continuous state space. We can only make use of a small number of sensors to get information on

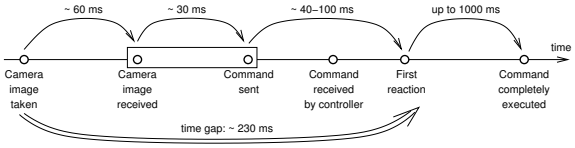

Fig. 2. Illustration of the time gap due to sensor and actuator delays.

the current state. Sensors are noisy and unreliable and it is not possible to observe all state variables directly, e.g. sensors typically used are not able to measure the velocity of other objects but they only yield a snapshot of the environment. Furthermore, other objects may be occluded by obstacles. In a stochastic sense, we are therefore faced with censored sensory input.

Figure 1 depicts the situation with which we are typically faced. This is clearly not a Markovian process since commands are executed with a certain delay and sensory input also refers to a certain point in past (Behnke *et al.*, 2003). Hence, the effect of an action taken by an agent can be observed no earlier than several hundred milliseconds later (cf. fig. 2). E.g. the system latencies of our soccer robots sum to approximately $230ms$ which is seven times larger than the length of the cycle interval of the control loop that we use. Since our robots are driving with up to $3\frac{m}{s}$ they meanwhile cover a distance of $60cm$.

To overcome the problem of latencies and to create a framework that is more similar to a Markov decision process we propose to bridge the time gap using sensor processing and predictive models. Thereto we carefully need to measure the latencies of all sensors and actuators and provide all measurements and actuator commands with timestamps to which they refer. Notice, that different sensors may have different delays and different motor commands may lead to different latencies even for the same motors. E.g. we observed that executing a command to turn the robot on a spot is realized quicker than a command to drive forwardly.

Using predictive models, we can replace the state estimated from the latest sensory input by the expected state at the time of command execution. Doing so, we replace outdated state information by a predicted state that is more similar to the true state at time of command execution assuming adequate predictive models. Hence, reinforcement learning becomes possible.

4. ESTIMATING PREDICTIVE MODELS

We use two types of sensors on our robots: an omnidirectional color camera on top and wheel encoder (Hafner *et al.*, 2006). Both types of sensors are heavily affected by noise and imprecision

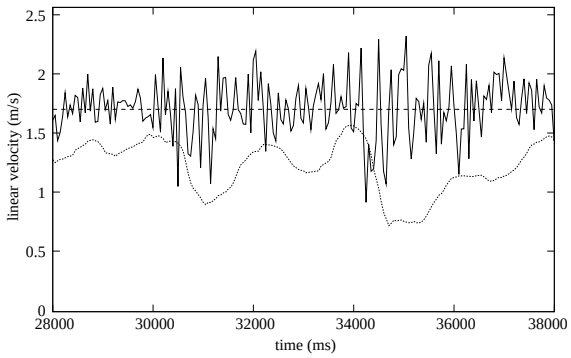

Fig. 3. Comparison of three velocity measures: desired linear velocity (dashed line), odometry (solid line) and estimated linear velocity (dotted line). The robot was given motor commands with constant linear velocity and varying angular velocity.

and their measurements have to be pre-processed to get meaningful information. To illustrate the noise level we depicted in figure 3 a comparison of desired velocities and measured velocities using wheel encoders (odometry). The noise of odometry is caused by wheel slippage and kinetic effects. Actually, the true velocity of the robot is below the desired one which means that the odometry overestimates the robot velocity.

The task to build a predictive model is to calculate reliable and accurate estimates of the current situation in the robot's environment and to create measures of the dynamics in which the environment changes over time. This has to be done taking into account the limited computation time of at most $25ms$ per camera image. In a soccer environment, this means:

- estimating the robot's pose (self-localization)
- estimating the robot's velocity
- estimating the ball position and its velocity
- estimating the position of other robots and their velocities

Except of the velocity of other robots, we developed robust and efficient algorithm for all of these subtasks.

The basic idea behind our approaches to estimate dynamic models is to fit observations to a parameterized model. In contrast to approaches based on probabilistic robotics (Thrun *et al.*, 2005) that have become very popular in recent years, the classical idea of fitting model parameters has the important advantage of combining high accuracy with computational efficiency which is crucial in the context of an autonomous robot application.

E.g. for the self-localization subtask, we use a methodology that fits the model of white field markings on the soccer field with white line segments observed in the camera image (Lauer *et*

al., 2005). Using efficient numerical algorithms we are able to calculate the robot's pose within $6ms$ while a probabilistic self-localization approach based on Particle filtering (Doucet, 1998) needs more than three times as long for the same task returning estimates of worse precision.

To estimate the ball velocity and the robot velocity, we use specialized parameterized motion models of the ball and the omnidirection robot, respectively (Lauer *et al.*, 2006). For the ball, we assume a linear motion sometimes interrupted when the ball is being kicked or collides with an obstacle. The estimation of the ball velocity thus becomes the task to solve linear regression. Hence we are very efficient and avoid the difficulties in optimizing the parameters of probabilistic techniques like Kalman filtering and Particle filtering.

The linear and angular velocity of the robot is estimated using a motion model of an omnidirectional robot (Lauer and Lange, 2006). The motion model itself is non-linear but, with an appropriate mathematical decomposition, we can derive the linear and angular velocity of the robot analytically with little computational effort. The dotted line in figure 3 shows the estimated linear velocities calculated with such an estimator.

Reliable estimates of dynamic models are not only helpful to generate a good behavior of the robot and to be able to interact with moving objects, e.g. intercepting and dribbling a ball, but they can also be used to bridge the time gap by predicting how a situation will develop in future. Thereto we can directly apply the motion models and perform a step-by-step simulation. Considering interactions of objects like the ball colliding with an obstacle further improves the predicted states.

Since the interval to bridge over is approximately $200ms$, several new motor commands are sent since the point in time to which the dynamic models refer to. Hence, the dynamics of the objects may change in between. Considering this change in a model of robot acceleration may also reduce the prediction error. Therefore, we plan to use supervised learning techniques to build such an acceleration model.

5. LEARNING ON REAL ROBOTS

Similar to the simulation league, we started learning skills in the midsize league using reinforcement learning. Again, the learning problem is modeled as a Markov decision process with a set of possible actions that can be taken by the robot, a description of the relevant objects in the environment and their dynamics, and a reward function that describes the task that should be learned. Instead of the unknown true state of the environment we use the predicted state that we get from the sensor fusion and simulation process described in the previous section.

As an example, let us again consider the ball intercepting problem already discussed in context of the simulation league (Müller, 2005). We can model the problem using a five-dimensional state space and a finite set of actions. In contrast to the simulation league, we have been able to reduce the dimension of the state space by one choosing an appropriate orientation of the coordinate system in which we describe the geometric and dynamic parameters. To simplify the problem, we decided to let the robot learn only its linear velocity while the heading of the robot is controlled using a hand-crafted policy.

In contrast to the simulation league, we do not know the true system dynamics of the problem. Thus, the TD(1)-algorithm cannot be applied here. Instead, we used the Q-learning (Watkins, 1989) algorithm that learns an optimal policy considering only samples of state-action-pairs, a successor state and the reward achieved. No knowledge of the true transition model is needed.

Like in the simulation league, the state space was too large to be represented in a table. We used a piecewise linear function approximator to represent the Q-function that tells how much reward can be achieved taking a certain action in a certain state and afterwards following the optimal strategy. Exploiting the learned Q-function greedily, i.e. choosing the action for which the Q-function yields the largest value, leads to an optimal behavior that maximizes the achieved reward.

An additional problem that occurs when learning algorithms are applied to real robots is that the number of experiments that can be carried out is limited to a few dozens. In contrast, to learn intercepting a ball in the simulation league, we carried out 100 000 experiments in simulation which is far from being realizable on a real robot.

To overcome this problem we used a special simulator that simulates a RoboCup robot and that has the same command interface as the real robot (Kleiner and Buchheim, 2003). Unfortunately, we could not use the simulation environment from simulation league since there an abstract interface is used that is very different from a physical meaningful interface.

After carrying out approximately one million experiments in the simulator we transfered the resulting policy to the real robot. It turned out that the learned strategy was able to intercept a rolling ball also in reality but the failure rate was larger than in simulation. This shows that the complexity of a task on a real robot is much more difficult than in simulation and that simulators - although

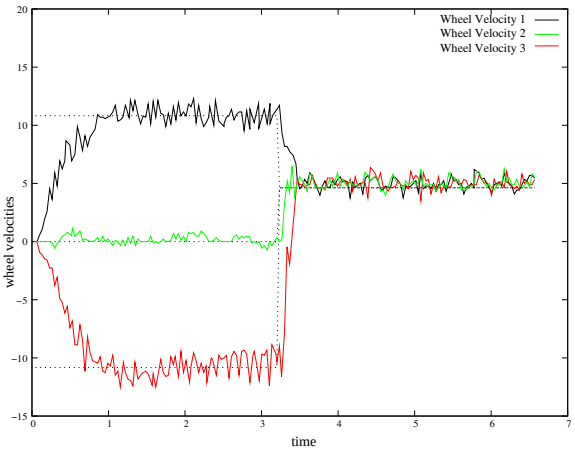

Fig. 4. An exemplary trajectory of the robot using the learned motor controllers. The solid lines show the achieved velocities of the three wheels while the dashed lines show the desired wheel velocities.

based on a physical modeling - do not cover all the sources of disturbance and noise that occur in practice. E.g. on the real robot we found out that due to kinetic effects a combination of a linear velocity sideways and an angular velocity results in a harder turning than expected and almost no linear movement. Effects like this typically are not modeled in simulators used in RoboCup although they are relevant for real robots.

An alternative approach of dealing with the lack of experiments that can be carried out on real robots was proposed in (Riedmiller, 2005). We used it to let a robot learn to drive to a certain position. The idea is to avoid simulation but to use the data acquired from experiments as intensive as possible by reusing them. The idea is related to the idea of bootstrapping in statistics. It is combined with an approximation of the Q-function using multi layer perceptions (Bishop, 1995).

The task to drive to a certain position was modeled in a very simple manner: three possible actions (drive forwardly, drive backwardly, stop) and a two dimensional state space (distance to the target position and velocity of the robot) have been used. The control of the tangential velocity and the robot heading was realized by a handcrafted controller.

With this modeling, the robot was able to learn driving to the desired position within 10 experiments. Optimal behavior was attained after 60 experiments. Hence, the optimal policy was learned within one hour on the real robot.

Beside the relatively simple task to drive to a position we also applied the improved reinforcement learning algorithm to the control of individual motors (Hafner and Riedmiller, 2006). So far we used PID controllers to control the motor

Table 1. Comparison of the tasks that have been learned in our simulation and midsize league team

level	simulation league	midsize league
cooperation	-7 vs. 8 positioning -attacking strategy -4 vs. 3 attacker positioning	–
skills	-ball intercepting -goto position -dribbling -precise kicking -1 vs. 1 positioning	-ball intercepting -goto position -approaching ball
low level control	–	-single motor control

voltages. But since the motor and robot dynamics contain nonlinearities caused by friction, stiction and the power amplifier, PID controllers are not optimal. The problem is described by four dimensional state variables and a one dimensional action set, namely the modification of motor voltage. We discretized the action to nine distinct voltage modification steps between which the reinforcement controller could choose.

Again we needed only 50 experiments on the real robot to learn the task. Each experiment needed five seconds of interaction with the robot and some additional seconds to update the Q-function. Totally, learning was realized within one hour. An exemplary trajectory that was generated with the learned controller is depicted in figure 4.

6. LEARNING ON DIFFERENT LEVELS

So far we discussed learning approaches used to acquire individual skills of a single robot. However, learning algorithms can be applied to all levels of strategy: on the level of individual skills like intercepting or dribbling the ball, on the level of motor control and on the level of multi agent cooperation. We realized learned behaviors on all of these three levels starting in the simulation league with individual skills like kicking, intercepting and dribbling the ball (Riedmiller and Merke, 2003).

On the basis of the individual skills we were able to learn multi agent coordination tasks like optimal positioning of several agents and optimal attack. To our surprise, the robots autonomously learned higher level concepts like playing give and go passes although we provided them only with the individual skills mentioned above.

In the midsize league we again started with individual skills like driving precisely to a given target position and intercepting a ball. Currently, we are working on a learned low level motor control that will be used to replace non-optimal PID controllers. A survey of all tasks for which learning has been applied is given in table 1.

7. DISCUSSION

Reinforcement learning techniques in the domain of autonomous robots provide the possibility to avoid hand-coded behavior and allows to create near optimal skills. We showed how this technique can be used in simulation and we compared it to the theoretically optimal policy. The learned policy turned out to be very close to the optimum.

Applying reinforcement learning to real world tasks needs adaptations to make the world look like a Markov decision process. We proposed several algorithms to estimate dynamic models of the environment that allow to predict how the environment of an autonomous robot will look like in future at the point in time when a selected action becomes active. This technique can be used to bridge the time gap that occurs in real world and thus enables reinforcement learning techniques to be applied in reality.

We demonstrated how learned policies can be used on real robots and which techniques have to be used to overcome the problem of little experiments that are possible in practice and high dimensional state spaces.

The final proof of concept could be supplied in the RoboCup tournament: the Brainstormers simulation league team became world champion in 2005 using skills and behaviors that have been learned with reinforcement learning and could thus outperform alternative approaches like handcoding and classical control techniques.

AKNOWLEDGEMENTS

This work was supported by the German Research Foundation (DFG SPP 1125).

REFERENCES

Behnke, Sven, Anna Egorova, Alexander Gloye, Raúl Rojas and Mark Simon (2003). Predicting away robot control latency.. In: *RoboCup 2003: Robot Soccer World Cup VII*. pp. 712–719.

Bishop, Christopher M. (1995). *Neural Networks for Pattern Classification*. Oxford University Press.

Doucet, Arnaud (1998). On sequential simulation-based methods for Bayesian filtering. Technical Report CUED/F-INFENG/TR.310. University of Cambridge.

Gabel, Thomas and Martin Riedmiller (2006). Learning a partial behavior for a competetive robotic soccer agent. *Künstliche Intelligenz, in press*.

Hafner, Roland and Martin Riedmiller (2006). A neural RL controller for speed control of a real robot. Submitted to: Robotics Science and Systems RSS06.

Hafner, Roland, Sascha Lange, Martin Lauer and Martin Riedmiller (2006). Brainstormers Tribots teamdescription. Submitted to: Robocup Symposium 2006.

Hu, Junling and Michael P. Wellman (1998). Multiagent reinforcement learning: theoretical framework and an algorithm. In: *Proceeding of the Fifteenth International Conference on Machine Learning*. pp. 242–250.

Kitano, Hiroaki, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa and Hitoshi Matsubara (1997). RoboCup: A challenge problem for AI. *AI Magazine* **18**(1), 73–85.

Kleiner, Alexander and Thorsten Buchheim (2003). A plugin-based architecture for simulation in the F2000 league. In: *Robocup 2003*. pp. 434–445.

Lauer, Martin and Sacha Lange (2006). Vision based techniques of modeling the environment for an autonomous robot application. submitted to: International Journal of Computer Vision.

Lauer, Martin, Sascha Lange and Martin Riedmiller (2005). Calculating the perfect match: an efficient and accurate approach for rob ot self-localization. In: *Robocup-2005*.

Lauer, Martin, Sascha Lange and Martin Riedmiller (2006). Moton estimation of moving objects for autonomous mobile robots. *Künstliche Intelligenz* **20**(1), 11–17.

Müller, Heiko (2005). Fahrtplanung eines Roboters mithilfe von Reinforcement Learning. Master's thesis. Universität Dortmund.

Riedmiller, M. and A. Merke (2003). Using Machine Learning Techniques in Complex Multi-Agent Domains. In: *Adaptivity and Learning* (I. Stamatescu, W. Menzel, M. Richter and U. Ratsch, Eds.). Springer.

Riedmiller, Martin (2005). Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In: *Proc. of the European Conference on Machine Learning*.

Sutton, Richard S. (1988). Learning to predict by the method of temporal differences. *Machine Learning* **3**, 9–44.

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement learning : an introduction*. MIT Press.

Thrun, Sebastian, Wolfram Burgard and Dieter Fox (2005). *Probabilistic Robotics*. MIT press.

Watkins, C. (1989). Learning from delayed rewards. PhD thesis. Department of Computer Science, Cambridge University. Cambridge, UK.