2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Learning to Dribble on a Real Robot by Success and Failure

Martin Riedmiller and Roland Hafner and Sascha Lange and Martin Lauer
Dept. of Mathematics and Informatics
Institute of Computer Science and Institute of Cognitive Science
University of Osnabrück

Abstract— **Learning directly on real world systems such as autonomous robots is a challenging task, especially if the training signal is given only in terms of success or failure (Reinforcement Learning). However, if successful, the controller has the advantage of being tailored exactly to the system it eventually has to control. Here we describe, how a neural network based RL controller learns the challenging task of ball dribbling directly on our Middle-Size robot. The learned behaviour was actively used throughout the RoboCup world championship tournament 2007 in Atlanta, where we won the first place. This contistutes another important step within our Brainstormers project. The goal of this project is to develop an intelligent control architecture for a soccer playing robot, that is able to learn more and more complex behaviours from scratch.**

I. INTRODUCTION

The idea behind the RoboCup initiative is to provide a testbed for the development of autonomous, intelligent systems. By playing soccer in different leagues, the ability of real and virtual systems to act reasonably in a highly dynamical, noisy and competitive environment is tested.

The Brainstormers's project was founded in 1998, with the goal to create an intelligent software architecture, that is able to acquire more and more of its behaviours by machine learning techniques. Especially, we focus on Reinforcement Learning methods, where the agent has to improve its behaviour by only being informed about its success or failure. Particularly important for us is the fact, that we do not only want to demonstrate that the methods work in principle, but they are actually applied in our competition teams [2], [1].

Starting in the simulation league, we were the first team to learn a very strong kicking behaviour by RL methods (applied for the first time in the competitions in 2000 and then ever since). Other behaviours, like quickly intercepting the ball, running to a given position, dribbling around an opponent where learned in the following years. Many of the learned behaviours made it into the competition team, since they were superior to all hand-coded and hand-tuned methods known at that time. Starting in 2002, we extended the RL approach to the question of multi-agent cooperation. Our complete attack was guided by a neural network, that has learned when and where to pass, where to dribble or where to go for an open position [6], [7]. However, in simulation one has the big advantage, that learning experience can be collected with virtually no limitation. Therefore, it is no problem to do hundreds of thousands of episodes, which where sometimes needed to achieve a highly optimizied behaviour.

The story is different, when one deals with real robots. Of course, one possibility is always to make a good simulator of the robot first, and then to train the behaviours by using this simulator. However, many effects that occur on the real robot are difficult to model and much effort has to be put into the development of a good simulator. Even then, simulation and real world may differ substantially. We entered the Middle-Size league in 2003 and our first behaviour to be learned was to intercept the ball. In these early approaches, we actually developped a simulator first and after several adaptations, the behaviour learned in simulation also behaved well on the real robot [4]. This intercept behaviour was used for the first time in our 2006 competition team, which won the worldchampionship in Bremen.

Dribbling is an example, where it is much more difficult to achieve a good simulator, since the interactions between ball and robot are very difficult to model. This is an example, where the advantage of being able to learn on a real robot become obvious. However, in order to do so, much more effective RL methods are needed.

In 2005, we proposed an off-line RL method called Neural Fitted Q Iteration (NFQ) [5]. By storing transitions and reusing them for every update of the Q function, this method drastically reduces the number of interactions needed with the system to be controlled. On our soccer robot, this method was successfully applied to learn the motor speed controllers on the real robot directly [3]. In the work presented here, we applied this method to the challenging task of dribbling. In only about 20 minutes of interaction with the real robot, we acquired enough information to learn from scratch a highly effective neural dribbling behaviour offline.

II. TASK DESCRIPTION

Dribbling here means to keep the ball in front of the robot, while heading to a given target. Since by the rule of the Middle-Size league only one-third of the ball might be covered by a dribbling device, the dribbling behaviour must carefully control the robot such that the ball is not lost or running out of the device when the robot changes direction. In previous years, we used a hand-coded and hand-tuned routine for this, but it showed drawbacks in making quite large movements when turning and by loosing the ball from now and then.

III. The RL controller

a) Goal: The goal of the dribbling controller is to turn the robot as quickly as possible to a given target direction without loosing the ball.

b) Controller structure: Control signals influencing the behaviour of the robot are: speed in relative x direction, the speed in relative y direction and the rotation speed. In our approach, we used a hybrid controller architecture: The rotation speed is set automatically by using a simple control law, based on the the difference between actual rotation angle and target direction. The learning part of the controller decides upon the correct speed in x and y direction, such that the ball is not lost, while the robot is turning.

c) Inputs: Based on the camera input and the internal sensors of the robots, the following state information is computed: velocity of the robot in x and y direction, rotation speed, heading direction relative to a given target direction. This state information serves as input to the controller. Additionally, the information whether the ball is within the desired dribbling area of the robot ('robot owns ball') or not ('ball is lost') is computed. This information is used to compute the reinforcement signal for learning.

d) Actions: Actions are target speed vectors for the x and y direction of the robot. Overall, 4 action pairs with different combinations of x and y speeds are used.

e) Description of the learning task: The task of the RL controller is to choose the x and y speed such that while turning, the ball is not lost. The controller is punished by a large negative reward, as soon as the ball left the dribbling device. When it has successfully turned to the target direction without loosing the ball, it is rewarded by 0 and the episode is considered stopped. Every intermediate time step yields a small constant negative reward. This formulation expresses our desire for a quickly turning policy, that does not loose the ball.

f) Learning procedure: The Neural Fitted Q Algorithm (NFQ) requires samples of transition behaviour as inputs. In principle, two extremes are possible to collect training data:

- collect data by always greedily exploiting the most recent policy. After each episode, data collection is stopped, and the neural Q function is updated by NFQ.

- collect the data completely randomly for several episodes without updating the Q function. The Q function is only updated at the end, using the NFQ procedure.

The first method always exploits the information contained in the data by imroving the neural Q value function before collecting new one. Therefore, data can be collected in a rather 'goal-oriented' manner. However, when the action set becomes large, one NFQ iteration can take quite a long time, so the human assistant, who has to survey the learning procedure, will have a very boring job.

Therefore we decided to go for the second method here (that's the one used in the video): At the beginning, a completely random behaviour was used to collect transition samples. This is done for about 10 minutes, where the human assistant just brings the ball back to the field, when the ball was lost. Using this sample set, NFQ can now be applied completely offline, without any further interaction with the real robot. After about 100 NFQ iterations, the resulting controllers were already very successful. They then were used, to do another 10 minutes of data collection, now with an already well behaving policy. Then the newly acquired transition samples were added to the previous samples and then again 100 NFQ offline iterations were performed. After that, a highly effective dribbling behaviour resulted, which was used in our 2007 world champion Middle-Size Team. You can see examples of the reliable and highly effective dribbling behaviour on the video (e.g. look at the space-efficient turns).

g) Neural controller used: The neural network that was used to learn and approximate the Q-function is a multilayer perceptron with sigmoid activation functions. It uses 5 input neurons (4 for the state information and 1 for coding the action, i.e. an x/y speed-pair). The hidden layers are two layers of 20 neurons each, and theres one output neuron, coding the Q-value. Action selection then is done by choosing the x-y pair, that results in the lowest Q-value for the current state.

IV. Concluion

Being able to learn directly on a real system without using an analytical model or a simulation, allows the controller to be tailored exactly to the system that eventually has to be controlled. We believe that this is a key feature for many (industrial) real world applications.

The NFQ method was successfully applied to learn to dribble a ball directly on our omni-directional soccer robot. Dribbling is a crucial behaviour for the success in a robot soccer game. Since our team can not kick very hard, this is particularly true for our team. Becoming world champion in 2007 with a neural behaviour that was learned directly on the real robot is therefore an important milestone in the RL history of our Brainstormers team.

References

[1] T. Gabel and M. Riedmiller. Learning a Partial Behavior for a competitive Soccer Agent. *Künstliche Intelligenz*, 2:18–23, 2006.

[2] T. Gabel and M. Riedmiller. On Experiences in a Complex and Competitive Gaming Domain: Reinforcement Learning Meets RoboCup. In *Proceedings of the IEEE Symposium on Computational Intelligence and Games*, Honolulu, USA, 2007.

[3] R. Hafner and M. Riedmiller. Neural Reinforcement Learning Controllers for a Real Robot Application. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 07)*, Rome, Italy, 2007.

[4] H. Müller, M. Lauer, R. Hafner, S. Lange, and M. Riedmiller. Making a Robot Learn to Play Soccer Using Reward and Punishment. In *Proceedings of the German Conference on Artificial Intelligence, KI 2007*, Osnabrück Germany, 2007.

[5] M. Riedmiller. Neural Fitted Q Iteration - First experiences with a data efficient neural Reinforcement Learning Method. In *Proc. of the European Conference on Machine Learning, ECML 2005*, Porto, Portugal, October 2005.

[6] M. Riedmiller and A. Merke. Using machine learning techniques in complex multi-agent domains. In I. Stamatescu, W. Menzel, M. Richter, and U. Ratsch, editors, *Adaptivity and Learning*. Springer, 2003.

[7] D. Withopf and M. Riedmiller. Effective methods for reinforcement learning in large multi-agent domains. *it - Information Technology Journal*, 5(47):241–249, 2005.