Towards a Life-Long Learning Soccer Agent*

Alexander Kleiner, Markus Dietl, and Bernhard Nebel

Institut für Informatik
Universität Freiburg
79110 Freiburg, Germany
{kleiner,dietl,nebel}@informatik.uni-freiburg.de

Abstract. One problem in robotic soccer (and in robotics in general) is to adapt skills and the overall behavior to a changing environment and to hardware improvements. We applied hierarchical reinforcement learning in an SMDP framework learning on all levels simultaneously. As our experiments show, learning simultaneously on the skill level and on the skill selection level is advantageous since it allows for a smooth adaption to a changing environment. Furthermore, the skills we trained turn also out to be quite competitive when run on the real robotic players of the players of our *CS Freiburg* team.

1 Introduction

The *RoboCup* context provides us with problems similar to those encountered by robots in real world tasks. The agents have to cope with a continuously changing environment, noisy perception and a huge state space [6]. Mid size robots are additionally confronted with a complex motion model and non-trivial ball handling problems. Programming robots overcoming all these difficulties is a tedious task. Furthermore, with changes in the environment or hardware improvements, previous solutions may not work any longer and it is necessary to reprogram the robots. *Reinforcement Learning (RL)* offers a rich set of adaptive solutions which have also proven to be applicable to complex domains [4]. However, before one can apply *RL*, it is necessary to reduce the state space. In particular, often one uses generalization techniques on the input space. We reduce the size of the state space by *tile coding* [1,2] which is a widely used method for linear function approximation in RL.

In addition, it is advantageous to decompose the task into *skills* that are selected on a higher level, instead of trying to learn a "universal" control strategy. For example, *dribbling*, *shooting*, and *taking the ball* are three different skills that can be learned individually. Once the robots have learned these skills, the robots can learn when to apply them – similar to *layered learning* [12].

While decomposing a task might simplify the learning problem, it can lead to problems when we want to adapt to new environmental conditions. Using a

* This work has been partially supported by *Deutsche Forschungsgemeinschaft* (DFG) and by *SICK AG*.

G.A. Kaminka, P.U. Lima, and R. Rojas (Eds.): RoboCup 2002, LNAI 2752, pp. 126–134, 2003.

"layered" approach and assuming that a new kicking device is used or that the carpet has changed, one would be forced to first adapt the basic skills to the new situation and then to adapt the selection strategy. However, it is not clear what the best setting would be to re-train the lower level skills in this case. As a matter of fact, we would like to confront our robots with the new situation and train both levels simultaneously. In other words, we want them to adapt their low level skills to the new environments as well as learning to decide which skill to apply in which situation [5]. The ultimate goal in this context is to build robotic soccer agents that improve their skills during their whole life and doing this as efficiently and quickly as possible.

In order to address these problems we decided to apply hierarchical RL based on Semi Markov Decision Processes (SMDPs), as introduced by Bradtke and Duff [3,7] and further developed by Sutton [13]. In contrast to Markov Decision Processes (MDPs), which are defined for an action execution at discrete time steps, SMDPs are providing a basis for learning to choose among *temporally abstract actions*. Temporally abstract actions are considered in standard SMDPs as *black box skills*, which execute a sequence of actions in a defined partition of the state space for an arbitrary amount of time.

RL methods have already successfully been applied to the simulation league in the *Karlsruhe Brainstormers* [8] and CMUnited [10] teams. This work is different from ours since both teams are focusing mainly on the multi-agent problem of robotic soccer and use different techniques for state space generalization. Stone and Sutton [11] have shown how RL trained agents can beat even hand-coded opponents in the *keepaway* scenario. Their skill selection has also been learned by SMDP techniques. However, their skills are hand-coded.

One team applying RL in the mid size league are *Osaka Trackies*, which use a method building self-organized hierarchical structures [14]. In contrast to other approaches which favor the decomposition to "standard" soccer skills, the resulting hierarchy consists of small, but very flexible skills. In contrast to our work that is build upon a world model, their system can be considered as behavior-based, because the state space is defined by uninterpreted images from the vision system.

The rest of the paper is structured as follows. In the next section we specify the SMDP learning model. In Section 3, we sketch how to apply our hierarchical RL approach to robotic soccer. In Section 4, we describe our experimental results, and in Section 5 we conclude.

2 Learning in (S)MDPs

The framework of MDPs provides a formal description for time discrete interactions between an agent and its environment. It is assumed that the agent chooses at discrete time steps t an action a_t according to the state s_t previously received from the world. An MDP is defined by the tuple *(S,A,T,R)*, where S is the set of world states, A is the set of actions, $T : S \times A \times S' \Rightarrow [0, 1]$ is the transition model and $R : S \times A \Rightarrow \Re$ is the reward function. The transition model T is

defined by $p(s_{t+1}|a_t, s_t)$ which returns for every world state s_t and action a_t the probability distribution over world states s_{t+1}. Furthermore, the reward function *R(s, a)* defines real valued rewards returned by the environment according to the agent's last state and action taken.

An MDP is solved when the agent has identified a policy π which maximizes rewards received over time. By RL methods this can be achieved by identifying the optimal value function $V^*(s)$, indicating the maximal future (discounted) rewards to be expected from state s. Once the optimal value function is found, the agent can behave optimally by selecting actions greedily according to $V^*(s)$.

There are well known methods to approximate the optimal value function by successive steps through the state space. One widely used method is known as *Q-Learning* [16] which allows learning without the transition model T. Rather than learning a mapping from states to values, this method learns an approximation for $Q^*(s, a)$ which maps from state-action pairs to values. The update rule for one-step Q-Learning is defined as

$$Q_{k+1}(s_t, a_t) := (1-\alpha)\, Q_k(s_t, a_t) + \alpha \left[R(s_t, a_t) + \gamma \max_{a \in A} Q_k(s_{t+1}, a_{t+1}) \right], \quad (1)$$

where α denotes the *learning rate*, and γ a *discount factor*.

Convergence speed of Q-Learning and other RL methods can be improved by considering eligibility traces [2]. The idea is, roughly speaking, to keep track of previously visited states and update their value when visiting states in the future. This yields the effect that a whole trace can be updated from the effect of one step. The influence of states on the past can be controlled by the parameter λ. Q-Learning with eligibility traces is denoted by $Q(\lambda)$.

In Q-Learning the value $Q^\pi(s, a)$ of a state s is the approximated utility for selecting a in s and following the greedy policy afterwards. Therefore the traces have to be cut off when selecting a non-greedy-action for execution (e.g. for exploration). However, when replacing $max_a Q_k(s_{t+1}, a_{t+1})$ by $Q_k(s_{t+1}, a_{t+1})$ in equation 1 and selecting a_{t+1} according to the policy selecting actions, we get the update for an on-policy method, known as *Sarsa* [9] that allows updates of the whole trace.

In SMDPs, actions are allowed to continue for more than one time step. An SMDP is an extension to the definition for MDPs and defined by the tuple *(S,A,T,R,F)*. F is defined by $p(t|s, a)$ and returns the probability of reaching the next SMDP state at time t when the *temporally abstract action* a is taken in state s. Q-Learning has been extended for learning in SMDPs [3]. The method is guaranteed to converge [7] when similar conditions as for standard Q-Learning are met.

The update rule for *SMDP Q-Learning* is defined as

$$Q_{k+1}(s_t, a_t) := (1-\alpha)\, Q_k(s_t, a_t) + \alpha \left[r + \gamma^t \max_{a \in A} Q_k(s_{t+1}, a_{t+1}) \right], \quad (2)$$

where t denotes the sampled time of execution and r the accumulated discounted reward received during the execution. Like the transition model T, the time model F is sampled by experience and has not to be known in advance.

In recent work, a unification of MDPs and SMDPs has been proposed [13]. It also has been shown that there is a clear advantage of interrupting temporally abstract actions during their execution and switch among them if the change is profitable. Our work, however, is based on the standard framework for SMDPs.

3 Applying Hierarchical Reinforcement Learning to Robotic Soccer

The learner's state space is based on the world model of a *CS Freiburg* player [17]. The world model is continuously updated in 100 msec intervals and consists basically of positions and velocities of the ball and robots on the field. Each sensor has a specific field of view which means that the world model can be incomplete.

The robot's trajectory is controlled by a differential drive. Furthermore, the robot is equipped with a custom manufactured kicking device for handling the ball. This device consists of a kicker to shoot and two "fingers" mounted on the left and righthand side of the kicker to control the ball. We noticed that effects of the kicking device and differential drive might vary on different robots of our team. It is one of the goals of our work to cope with these differences.

Experimental results presented in this paper have firstly been achieved using a simulation of our robots. The simulation is implemented as a client-server architecture and executes asynchronously to world model updates. Since the world models generated on the real robots and generated from the simulation are the same, the learner can switch between them on the fly.

Given a constant cycle time, the interface can be used for learning in MDPs. Within cycle c_t the learner receives the current world state s_t, and consequently returns the selected action a_t which causes s_{t+1} to be emitted in the successive cycle c_{t+1}. Since the world model provides reliable "high-level" features of the robot's perception, simulated and real perception can be considered as almost equivalent.

The application of RL to our players has been carried out in a straightforward manner, similar to the way humans would train soccer. Firstly, basic skills, namely *shootGoal*, *shootAway*, *dribbleBall*, *searchBall*, *turnBall*, *approachBall* and *freeFromStall* [18,17] have been trained in simple, static scenarios. Secondly, the appropriate selection of these skills and their embedding into the task has been trained in a realistic scenario which was a game against a *CS Freiburg* player from 2001. Finally, the simulation-trained soccer player has been executed on one of our real robots.

Early experiments have shown that *Sarsa(*λ*)* performs better than Q(λ) when learning the skills above. We assume that this is mainly caused by the necessary cuts of eligibility traces after non-greedy actions when applying the off-policy method. Therefore, skills have been trained by *Sarsa(*λ*)*. We set $\gamma = 1.0$ (due to the presence of an absorbing state), $\alpha = 0.1$ (small, due to the non-determinism of the environment), $\epsilon = 0.05$ (small, since high exploration could lead to failures) and $\lambda = 0.8$ (a common value when learning with n-step updates).

The *state space* of each skill consists of features extracted from the world model, e.g. *distance* and *angle* to the *ball* or to the *opponent*. These features

were chosen according to their relevance for the behavior to be learned. The automation of this selection will be subject of future work. The *action space* of each skill is given by two discretized scalar values for the translational and rotational velocity and one binary value for kicking. For each skill, terminal states have been defined depending on the skill's natural goal. The skill *approachBall*, for example, terminates when either the ball has been approached successfully or could not be seen by the robot anymore. We defined a reward of 100 when the goal state was reached, a reward of -100 when the robot failed its task and a reward of -1 for each action taken.

For the learning of the skill-selection SMDP we decided to allow a high degree of exploration to guarantee the recurrent selection of all skills. Exploration on the skill selection level supports the selection of skills with low expectation of future rewards. This leads to more training of those skills and therefore to their improvement which might lead to higher expectations. Hence, we applied $Q(\lambda)$, since it is an off-policy method that learns the optimal policy, regardless of performing explorative actions[1]. During learning, goals made by the learner were rewarded with 100, goals made by the opponent with -100 and steps taken in a skill with -1. The state space of the SMDP was defined by the scalars *angle* and *distance* to ball, *angle* and *distance* to opponent, *angle* and *distance* to goal, and the binary values *ball is visible* and *opponent is visible*.

4 Experimental Results

The results presented in this section are selected from a series of experiments and are representative. Each experiment has been carried out with a learner that was equipped with previously trained skills that had been learned in simple scenarios. All graphs shown in this section are smoothed by averaging over 100 episodes.

In the first series of experiments the task of the learner was to learn the selection of skills when playing against a static goalkeeper. We intentionally chose a simple scenario in order to give a first impression of the learner's performance. During each episode the goalkeeper was placed on an arbitrary position in front of the goal, whereas learner and ball were placed anywhere on the field. We compared two learners, one and that was focusing on the learning of skill selection, and a second that was additionally allowed to improve its skills further. Figure 1 shows the progress of the two learners.

The baseline indicates the average of the accumulated rewards a CS Freiburg Player achieves during one episode. In both settings a good action selection strategy was learned after 500 episodes. Learning within skills, however, leads to a noticeably better performance. Although the scenario in the experiment was similar to the one used to pre-learn the skills, the learning within the skills enables the learner to adapt more flexibly to the game-playing situation.

[1] Note, due to the much smaller number of steps during an SMDP episode, the negative effect of Q-Learning on eligibility traces is of minor importance.

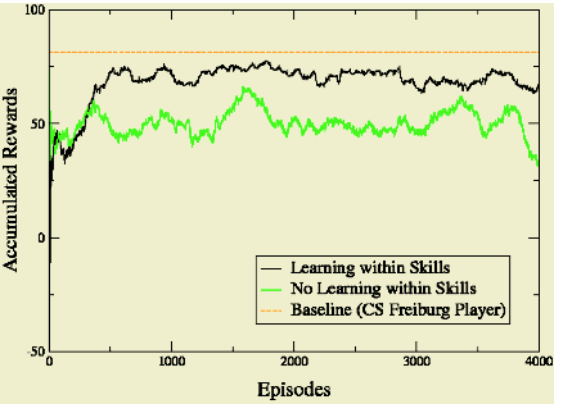

Fig. 1. Learner versus static goalkeeper with and without learning within skills

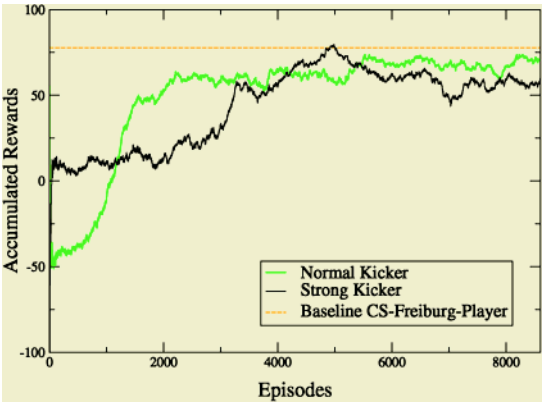

Fig. 2. Learner versus a *CS Freiburg* player with a normal and with a strong kicker. The initial skills were trained with a strong kicker. Both players reach an adequate level of play after some time of training

The results presented in Figure 2 demonstrate how the system reacts on a significant change of the environment. We compared learners with a normal and a strong kicker. The strong kicker was able to accelerate the ball much faster, but less precise. The task now was to compete against a *CS Freiburg* player that was slowed down to a third of its normal velocity. At the beginning of each episode the learner and ball were placed randomly into one half, facing the opponent.

Again, the baseline indicates the average of the accumulated rewards a CS Freiburg Player with normal velocity achieves during one episode. Due to the fact that skills were pre-trained with a strong kicker the learner using the normal kicker reaches less reward during the first 1000 episodes. After 6000 episodes, however, playing with a normal kicker turns out to be more successful than playing with the strong one. The learner with the normal kicker develops a different way of playing: He is dribbling more often to the front of the goal and

Table 1. Selected skills if learner was ball owner (in %)

	DribbleBall	ShootGoal	ShootAway	TurnBall
Normal Kicker	48.9	37.0	4.8	9.3
Strong Kicker	30.3	57.6	4.5	7.7

performs a rather precise shoot from small distance. Table 1 shows the frequency of selecting particular offensive skills.

The learner with the normal kicker tends to select *dribbleBall* more often, whereas the player with the strong kicker continues to shoot from further distances. Finally, both learners reach a similar level of performance that is winning against their opponent.

The previous experiments evaluated the learner's overall performance. It is also important, however, how the skills themselves improve over time. Figure 3 documents the learning progress of the skill *shootGoal*.

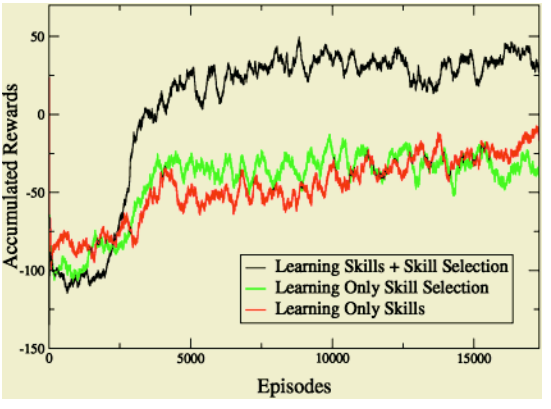

Fig. 3. Learning of the skill *shootGoal* while playing against a *CS Freiburg* player

The result shows that the simultanous learning of skills and their selection leads to higher rewards. The learner improving skills and skill selection reached an accumulated reward of nearly 50, whereas the other learners could not reach more than −25. Without learning the skill selection, skills are executed randomly, and thus also in inadequate situations. For example, *ShootGoal* could be chosen when facing the own goal. Certainly, it is possible that *ShootGoal* learns to handle such situations as well, but this could take a very long time of learning. In fact, the slowly improving curve for learning without skill selection indicates this learning process. On the other hand, without learning inside the skills, skills are executed as they were trained for static scenarios. Figure 3 shows that adaption of skills to the task at hand benefits the overall result.

Finally, we started a first experiment with our best learned skills and skill-selection on a real robot. The task was, as also evaluated in the simulation, a static scenario with a goalkeeper in front of the goal. The learner started an episode from the center of the field, whereas the ball was placed randomly. As

we expected, the learner started to chose reasonable skills, such as *searchBall* to locate the ball on the field, *approachBall* to get close to it and *turnBall* to get the right direction. To our surprise, most skills executed impressively well. The learner was robustly playing the ball without losing quality in skill selection or execution.

While playing for one hour the learner was able to score 0.75 goals per minute. In contrast, the hand-coded CS Freiburg player scores 0.94 goals per minute when limited to one third of its maximal velocity and 1.37 goals per minute when playing with maximal velocity. Although the performance of the player is superior, the result is remarkable, since the learner was completely trained in the simulation. Note that the learners result was achieved by far less time for design and parametrization.

In order to evaluate how the total performance improves over time, more than one hour of playing will be necessary. A long-term evaluation will be presented in future work.

5 Conclusion and Discussion

We studied the applicability of hierarchical reinforcement learning to robots in the mid size league. For our particular setting, RL methods perform remarkably well. Our learner was able to compete with one of our hand-coded players, even when environmental conditions were changed. Additionally, the low amount of learning time indicates that there is still potential for learning in hierarchies with more than two levels.

The experiments show that learning inside skills improves the overall performance significantly. Thus, the results lead to two important conclusions: Firstly, the whole system achieves higher adaptivity to changes in the environment while acting stable without tending to aimless behavior. Secondly, based on the fact that skills adapt themselves to the global task, it seems to be possible to reuse these skills for a different task, such as ball passing or general team cooperation.

Our final experiment on a real soccer robot has shown that knowledge learned in our simulation can be reused for a real-world task. It can be assumed that the hierarchical structure supports the stable behavior of the robots.

In future work we will investigate how the process of adaption can be accelerated further by using more flexible hierarchies. Furthermore it will be interesting, whether our implementation can be scaled-up to the game of more than one robot, particularly, in which way single players are able to adapt their skill selection when they are exposed to the multi-agent problem.

References

1. J. S. Albus. A theory of cerebellar function. In *Mathematical Biosciences*, volume 10, pages 25–61.
2. A. Barto and R.S. Sutton. *Reinforcement Learning – An Introduction*. MIT Press, Cambridge, 1998.

3. S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time Markov decision problems. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 393–400. The MIT Press, 1995.
4. R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 1017–1023. The MIT Press, 1996.
5. Thomas G. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *Fifteenth International Conference on Machine Learning*. Morgan Kaufmann, 1998.
6. H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The RoboCup synthetic agent challenge,97. In *International Joint Conference on Artificial Intelligence (IJCAI97)*, 1997.
7. R. Parr. *Hierarchical Control and Learning for Markov decision processes*. Ph.d. thesis, University of California at Berkeley, 1998.
8. M. Riedmiller and A. Merke. Karlsruhe Brainstormers – a reinforcement learning approach to robotic soccer ii. In Veloso et al. [15]. To appear.
9. G. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR, Cambridge University Engineering Department, 1996.
10. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, Berlin, Heidelberg, New York, 2000. Springer-Verlag.
11. P. Stone and R.S. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
12. Peter Stone and Manuela Veloso. Layered learning. In R. Lopez de Mantaras and E. Plaza, editors, *Eleventh European Conference on Machine Learning (ECML-2000)*. Springer-Verlag, 2000.
13. R. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. In *Artificial Intelligence*, volume 112, pages 181–211, 1999.
14. Y. Takahashi and M. Asada. Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 395–402, 2000.
15. M. Veloso, T. Balch, and P. Stone, editors. *International RoboCup Symposium 2001*, 2002. To appear.
16. C. J. C. H. Watkins. *Learning with Delayed Rewards*. Ph.d. thesis, Cambridge University., 1989.
17. T. Weigel, A. Kleiner, F. Diesch, M. Dietl, J. S. Gutmann, B. Nebel, P. Stiegeler, and B. Szerbakowski. CS Freiburg 2001. In Veloso et al. [15]. To appear.
18. Thilo Weigel, Willi Auerbach, Markus Dietl, Burkhard Dümler, Jens-Steffen Gutmann, Kornel Marko, Klaus Müller, Bernhard Nebel, Boris Szerbakowski, and Maximilian Thiel. CS Freiburg: Doing the right thing in a group. In P. Stone, G. Kraetzschmar, and T. Balch, editors, *RoboCup-2000: Robot Soccer World Cup IV*, pages 52–63. Springer-Verlag, Berlin, Heidelberg, New York, 2001.