Imitative Reinforcement Learning for Soccer Playing Robots

Tobias Latzke, Sven Behnke, and Maren Bennewitz

University of Freiburg, Computer Science Institute, D-79110 Freiburg, Germany
{latzke,behnke,maren}@informatik.uni-freiburg.de,
WWW home page: http://www.NimbRo.net

Abstract. In this paper, we apply Reinforcement Learning (RL) to a real-world task. While complex problems have been solved by RL in simulated worlds, the costs of obtaining enough training examples often prohibits the use of plain RL in real-world scenarios. We propose three approaches to reduce training expenses for real-world RL. Firstly, we replace the random exploration of the huge search space, which plain RL uses, by guided exploration that imitates a teacher. Secondly, we use experiences not only once but store and reuse them later on when their value is easier to assess. Finally, we utilize function approximators in order to represent the experience in a way that balances between generalization and discrimination. We evaluate the performance of the combined extensions of plain RL using a humanoid robot in the RoboCup soccer domain. As we show in simulation and real-world experiments, our approach enables the robot to quickly learn fundamental soccer skills.

1 Introduction

Reinforcement learning (RL) is an established machine learning technique. In a trial and error based procedure, an agent acquires knowledge about the consequences of its actions and strategies to attain a certain goal [1]. While RL methods have been successfully applied to complex problems in simulated environments [2, 3], they have rarely been used for real-world scenarios. The high costs of obtaining enough training examples for real systems often prohibits the acquisition of successful behavior by means of plain trial and error. The central intention of our work is the reduction of training expenses for RL methods so that they are applicable to real-world scenarios. In this paper, we propose three approaches to achieve this goal:

1. Speeding-up the exploration through imitation of a teacher
2. Repeated reevaluation of past experiences
3. Application of function approximators for better generalization

Imitation allows for knowledge transfer by observation between sufficiently similar agents. Imitation learning is a well established concept in robotics [4–9]. The idea is that the learning agent observes the actions of an experienced agent as well as the corresponding consequences. These observations give the learner clues about successful strategies to reach the goal. Through the imitation of the teacher, the exploration of the

huge search space is guided to regions that are promising. The learning agent no longer depends on random exploration but rather on meaningful indications which actions to choose.

In addition to the guidance by other agent's experiences, the extensive exploitation of own experiences is crucial for learning in real-world systems. Many learning algorithms rely on the online processing of experiences at execution time and discard them immediately afterwards. Often however, the full merit of an experience can be assessed only later in the learning process when more information is available. We reduce the amount of training data that has to be collected by storing and reusing experiences. The extensive use of both, own and observed experiences, provides the necessary knowledge to choose promising actions that lead to a successful performance.

In complex domains, however, it is highly unlikely that exactly the same situation is encountered twice. Generalization to similar situations is therefore essential. On the other hand, generalization should not prevent discrimination between different situations. The function approximation technique presented in this paper combines the advantages of quick generalization and accurate long-term discrimination.

We present extensive experiments to evaluate the performance of a combination of the proposed extensions to plain RL. In simulation as well as in real-world experiments with a humanoid robot in the RoboCup soccer domain, we demonstrate how the robot is able to quickly learn fundamental soccer skills.

This paper is organized as follows. In the following section, we briefly introduce Q-learning, which is a popular RL algorithm. In Section 3, we describe function approximation and its application to Q-learning. In Section 4, we present our approach to function approximation that allows for quick generalization as well as for sufficient discrimination. Section 5 explains our use of imitation and memory to guide the exploration. Section 6 describes the robot hardware as well as the application of RL to a specific task and Section 7 presents the experimental results obtained in simulation and with a real robot. Finally, in Section 8 we discuss related work.

2 Q-Learning

The framework underlying RL is that of Markov decision processes (MDPs), which describe the effects of actions in a stochastic environment and the possible rewards at the different environmental states. The goal of the agent is to maximize the expected (discounted) future reward, without knowing the MDP or the reward function in advance.

The action selection according to the current state is called the agent's policy. RL methods use an estimate of the expected cumulated future reward, the utility function, to derive a policy in order to maximize the long-term reward. In our work we use an ϵ-greedy policy, i.e. the agent chooses a random action with probability ϵ and the action with the highest utility expectation otherwise.

Temporal-difference (TD) methods [10] like Q-learning perform an update of the utility estimate after each transition (s, a, s'), where s is the state in which action a was executed and s' is the resulting state. As the cumulated future reward is not known in advance, TD methods use an estimate of the utility of s' to update the utility function.

In Q-learning, the utility function is called Q and maintains utility values for each state-action pair. The update rule after a transition is given by

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha \cdot \left(r_{t+1} + \gamma \cdot \max_{a' \in \mathcal{A}} Q_t(s',a') - Q_t(s,a) \right) . \tag{1}$$

Here, r is the immediate reward, $0 < \alpha \leq 1$ is the learning rate and $0 < \gamma \leq 1$ is the discount factor.

3 Function Approximation

A tabular representation of the Q-function that stores one estimation for each state-action pair is only useful for small problems. In practice however, the number of possible states is often very large or even infinite, making it impossible to maintain a table of all Q-values. Furthermore, the completely isolated evaluation of each state-action pair is not appropriate since it does not happen often that exactly the same situation is encountered twice. In practice, many situations appear similar and are discriminable only in detail. Experiences should therefore be generalized. Function approximation is a method that is often used for generalization.

The general notation of function approximation is that given we have input-output pairs (x, y), we want to compute a function f that is an approximation of the unknown function f^* that produced (x, y). First, a prototype for f has to be defined, i.e. a class of functions, that allows to express f^* with a suitable parameterization. The approximation f is represented by a parameter vector $\boldsymbol{\theta}$, which can be seen as instantiation of the function prototype.

3.1 Function Approximation Using Gradient Descent Methods

The idea of gradient descent methods is to modify the parameter vector $\boldsymbol{\theta}$ toward the direction that yields the greatest error reduction for the example under consideration. This is done by calculating the gradient of the local error term with respect to $\boldsymbol{\theta}$

$$\begin{aligned} \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \frac{1}{2}\eta \nabla_{\boldsymbol{\theta}_t} (y - f_t(x))^2 \\ &= \boldsymbol{\theta}_t + \eta (y - f_t(x)) \nabla_{\boldsymbol{\theta}_t} f_t(x) \ . \end{aligned} \tag{2}$$

Here, η is called the step size parameter and $\boldsymbol{\theta}_t$ and f_t indicate the parameter vector as well as the approximation of f^* at time t.

In this paper, we consider the special case of linear gradient descent. We extract a set of features from the input x and represent them as the feature vector $\boldsymbol{\phi}_x$. The number of parameters n is equal to the number of features and the function prototype is given by the weighted sum of the features

$$f(x) = \sum_{i=1}^{n} \boldsymbol{\theta}(i) \boldsymbol{\phi}_x(i) \ . \tag{3}$$

The components of $\boldsymbol{\theta}$ are called the weights of the corresponding features. The advantage of this prototype is that the gradient $\nabla_{\boldsymbol{\theta}} f(x)$ corresponds to the feature vector $\boldsymbol{\phi}_x$.

However, care has to be taken that suitable features are extracted from x to provide linear correlation between $\boldsymbol{\phi}_x$ and y in the case that the correlation between x and y is non-linear.

3.2 Q-Learning with Function Approximation

A training example (x, y) in RL consists of the previous state s and the executed action a as input and the target for Q-learning $r_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_t(s', a')$ as output. The update rule for Q-learning with linear gradient descent is then given by:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta(r_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_t(s', a') - Q_t(s, a))\, \boldsymbol{\phi}_{s,a} \ . \tag{4}$$

The update rule contains the feature vector $\boldsymbol{\phi}_{s,a}$, which is the function's gradient. This implements responsibility assignment: Those features, that have been most active in the last decision, are most responsible for the current local error. Thus the corresponding weights have to be adjusted most.

4 Feature Construction

A typical approach for feature construction is the use of input features that indicate whether or not the input is inside a certain region of the state space. Such a region is called the receptive field of the corresponding feature. We construct the features following the principles of coarse coding [11]. The essential idea is to use multiple large receptive fields, which may overlap, so that an input can activate multiple features. Two inputs are similar in the features that are present for both or absent for both and they differ in the features that are present for only one of the inputs. The simultaneous consideration of similarity and difference is one important instrument to deal with the conflict between generalization and discrimination, which typically occurs in learning systems. In this work, we use a variant with continuous features that have an activation value between 0 (i.e. the input is outside the receptive field) and 1 (i.e. the input is in the center of the receptive field). The activation function Φ is called the feature's shape.

In the one-dimensional case the receptive field is an interval $[b_0, b_1]$. Let $c = \frac{b_0 + b_1}{2}$ be the center of the receptive field. Then the activation function is:

$$\Phi(x) = \begin{cases} 1 - \frac{2|x-c|}{b_1 - b_0} & \text{if} \quad x \in [b_0, b_1] \\ 0 & \text{else} \end{cases} \ . \tag{5}$$

For the transfer to n dimensions we propose the product of the activities on each dimension respectively:

$$\Phi(x^{(1)}, \ldots, x^{(n)}) = \begin{cases} \prod_{i=1}^{n} \left(1 - \frac{2|x^{(i)} - c^{(i)}|}{b_1^{(i)} - b_0^{(i)}}\right) & \text{if} \quad \forall i \leq n\colon\ x^{(i)} \in [b_0^{(i)}, b_1^{(i)}] \\ 0 & \text{else} \end{cases} \ , \tag{6}$$

where $x^{(i)}$ and $b_{0/1}^{(i)}$ denote the input and the borders of the receptive field in the dimension i.

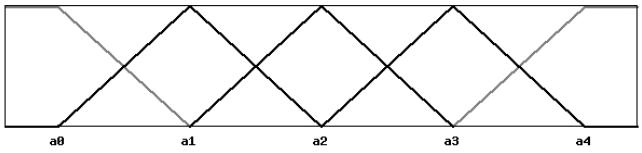

Fig. 1. 1-dimensional generic feature.

The whole state space is covered completely by a group of features. We call this group a generic feature. The dimensions are partitioned into uniform intervals with the borders $a_0, \ldots, a_e$, where e is the total number of intervals on this dimension (the resolution). In one dimension each receptive field covers two adjacent intervals with its peak exactly on their border. For each pair of intervals there is one feature plus two extra features at the border of the domain. Fig. 1 shows the generated features for a resolution of $e = 4$. For each input value there are two active features with a total activation of 1. Each feature contributes to the approximation according to its activation. This can be interpreted as the feature's responsibility for a certain input. The systematic overlapping within a generic feature corresponds to a smooth shift of responsibility between two adjacent features. The resulting approximation is a linear interpolation of the adjacent weights.

5 Imitation and Memory

Imitation is an important mechanism for the transfer of knowledge and skills between similar agents. An imitating agent can observe the behavior of an experienced agent and thus draw conclusions about the consequences of certain actions.

In this paper, the imitating agent has full access to experiences of a teacher. These experiences are provided as sequences of states and actions in the learning agent's representation along with the corresponding rewards. Our imitation approach relies on the evaluation of these sequences according to own criteria. One possibility to do so, is the application of the Q-learning algorithm with the stored sequence of actions. The imitation effect can be increased by processing the sequences in temporally inversed order. This is possible because the whole sequence of transitions and rewards is known in advance.

One advantage of storing and evaluating such sequences is the fact that the same algorithm can be applied to processing own experiences. This can be seen as a form of episodic memory, which allows to "revive" own experiences when their utility is easier to access. If, e.g., the agent chooses a very good action by chance, but does not see immediately that it was a good action or why it was good, then it can be helpful to revive that experience later in the learning process when increased knowledge allows for a better assessment of this observation.

The evaluation of observations in temporally inversed order is related to another technique for faster information propagation in temporal-difference methods. Eligibility traces [10, 12] use the *last* local error to update the value not only for the current state but for the recently visited states as well. So there is no individual local error used to

update the previous states of the stored sequence. In contrast to that, we calculate a *new* local error for *each* transition in the stored sequence. Thus, we treat each step of the sequence as if it was the actual observation. Convergence proofs for table-based Q-learning (e.g. [13]) require to visit all states and to choose all actions infinitely often, but they do not make any assumptions about the ordering of these observations. Thus, the guarantees on convergence remain unchanged under the appropriate conditions.

6 Task and Implementation

The proposed concepts of function approximation, imitation, and memory are evaluated for a humanoid toy robot called RoboSapien, which has been augmented with a camera and a Pocket PC as to allow for autonomous behavior as proposed in [14]. The task of the agent is to dribble the ball from different positions into the empty goal. Note that dribbling is achieved by walking against the ball. There is no explicit movement to kick the ball. The exact task setup is taken from the scoring test defined in [14]. In this test, the robot stands on the most distant point of the center circle, facing the empty goal. The ball is placed at ten different positions on the other half of the center circle in the robot's field of view. One advantage of the adoption of this existing scoring test is the possibility to compare the performance to an existing hard coded behavior.

According to the given task, suitable state variables have to be chosen that contain all relevant information for learning a good policy. Obviously, the positions of the ball and the goal are necessary to perform the task. While the ball position can be expressed by two variables (e.g. angle and distance), this is not sufficient for the goal position. Since the goal has a significant width, an additional variable is required (e.g. left post angle and right post angle instead of one angle). These five variables define the unambiguous position of all relevant objects and their orientation to each other. There are many possibilities to represent this information.

The previous paragraph implicitly assumes an egocentric representation of the information, i.e. the relative positions of the ball and the goal from the agent's point of view. Another possibility is the transformation to an allocentric representation, i.e. the absolute positions of ball and robot on the field (including the robot's orientation). Further possibilities are e.g. egocentric Cartesian coordinates (x and y) instead of polar coordinates (angle and distance).

In this work, we use a combination of these representations and additional variables like the square roots of the distances to the ball and to the goal. In total we use 27 variables to represent the state space. This redundant representation allows the simultaneous consideration of different aspects of the situation. Each single representation is well-suited for the detection of some properties and similarities of situations whereas others can hardly be distinguished. The combination of several representations allows to benefit from the advantages of each representation. The additional state variables do not cause significant additional costs of computation since they just provide another perspective on the same data. As our function approximation method relies on many partitionings of the state space anyway, there is no difference (concerning the number of features) in using two partitionings of the same representation and using one partitioning of two different representations.

The action space of our robot consists of four possible actions: *walk_forward*, *walk_backward*, *turn_right*, and *turn_left*, that are executed for 3.2s each and separated by a short break of 0.4s. This pause is required to ensure that the actions' effects do not depend on previous actions but only on the current state.

As explained above, we use Q-learning with linear gradient descent to learn the policy. The function approximator consists of multiple generic features (see Section 4) that consider only a small subset of the available state variables each. First, a minimal representation is chosen from the redundant state representation, i.e. five independent variables that unambiguously describe the positions of all relevant objects and their orientation to each other. Then a generic feature is created for all possible combinations of two and of three dimensions out of these five. This is done for several minimal representations resulting in a total number of 140 generic features, that only consider the state space. These features are important to estimate the value of the current situation. However, they do not distinguish between the different possible next actions. Hence, another 140 features are added that additionally consider the action space.

To reduce the complexity of the search space, we exploit symmetry. This technique is widely used in search or optimization problems [15, 16]. Here, we use mirroring along the horizontal axis of the field. The value of turning left is equivalent to the value of turning right in the mirrored situation. Thus, it is sufficient to learn the value of turning right. Similarly, for walking forward or backward it is sufficient to learn the value for only one half of the state space.

7 Experimental Results

We evaluate the performance of the proposed concepts as follows. The task is described as an episodic RL task. Each action introduces an immediate reward of -1 per time step. The slight punishment of each action is sometimes called the costs of the actions. Each episode has three possible outcomes with the rewards:

- *Success:* The Ball is inside the opponent goal, reward: +1000
- *Failure:* The ball is outside the field, reward: -400
- *Time-out:* Abortion after 300sec, no additional reward

In addition to the ten situations defined for the scoring test, we define a set of training situations. This is to ensure that the agent regularly encounters the different regions of the state space. The agent alternately starts in a situation from the scoring test and in a training situation. A pair of training episode and scoring test episode is called a trial. The performance is evaluated separately for the scoring test and for the training situations, since we are interested in the performance on the given task, i.e. the scoring test. The presented experimental results refer to the performance in the scoring test.

Our main performance criterion is the success rate, i.e. the number of goals divided by the number of episodes. The success rate is averaged over the last 50 episodes. Since we initialized the success history with 50 failed episodes, the success rate is 0 in the beginning and only after the 50 episodes, the average is based on actual episodes. This is indicated by a vertical line in the plots. With the real system, fewer episodes can be run due to time constraints. In this case, the performance is averaged over 20 episodes.

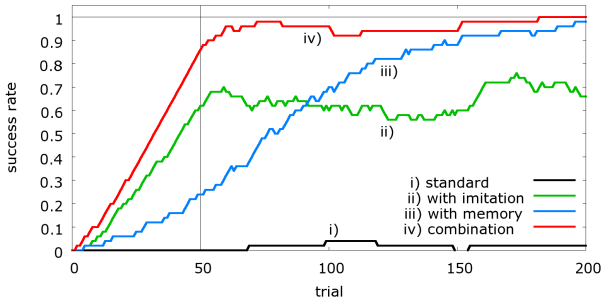

Fig. 2. The effects of imitation and memory. As can be seen, by applying our proposed approach using imitation and memory (Setup iv), the robot needs only few trials in order to come up with a good policy.

Our second criterion is the duration of successful episodes. It allows to distinguish the performance of policies that have a high success rate. This value is averaged over 50 successful episodes.

The influence of the different parameters can be evaluated faster and more systematical in a simulated environment, compared to using the real system. We first present the results obtained in a simulator and show the performance on the real robot afterwards.

7.1 Accelerating Learning by Using Imitation and Memory

First, we show that imitation and the reevaluation of own experiences seriously accelerate learning. Both approaches are evaluated separately and in combination using the following setups:

i) Standard Q-learning with $\eta = 0.2$ (constant); $\gamma = 0.98$; $\epsilon = 0$;[1]
ii) Same as i), additionally after each episode: evaluation of 36 successful episodes of an experienced agent
iii) Same as i), additionally after each episode: evaluation of the last 36 own episodes
iv) Same as iii), additionally: memory initialized with 36 successful episodes of an experienced agent (that will be replaced by own experiences after each trial)

The experienced agent is a human controlled RoboSapien with a success rate of 100%. As Fig. 2 shows, classical Q-learning does not lead to noticeable success within a reasonable time. Isolated imitation as employed in Setup ii) results in a behavior with a success rate of about 70% already after few trials. However, the robot does not improve further. It seems that the extensive use of stored experiences leads to a biased transition model that prohibits further progress in learning. When the robot uses a memory of own experiences only, learning starts more slowly but leads after 200 trials to a success rate near 100%. The combination of the two concepts imitation and memory (Setup iv) yields an almost immediate success rate of about 90%. As can be seen, after 70 trials the agent has learned a good policy with a stable success rate near 100%.

[1] Experiments with $\epsilon > 0$ ended up with similar results.

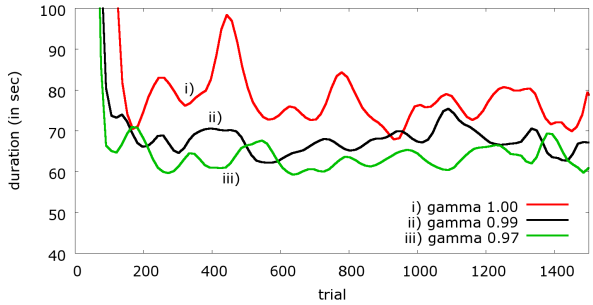

Fig. 3. Duration of successful episodes. A lower value of γ leads to a faster accomplishment of the task.

7.2 Influence of the Discount Factor

The parameter γ is used to discount rewards that will be gained in the future. One interesting effect of this future discount is that the robot prefers solutions with shorter sequences of actions, as can be seen in Fig. 3. This figure shows the duration of successful episodes for different values of γ. The success rate (not shown in the plot) does not differ significantly for the different values. However, the goal is reached much faster with $\gamma = 0.97$ than without discounting the future ($\gamma = 1$). Thus, the discount factor has an important influence on the policy that is going to be learned. A lower value may be advantageous for time-sensitive tasks. On the other hand, quick solutions might involve a higher risk of failure, so that γ cannot be chosen arbitrarily small.

7.3 Results with the Real Robot

As a first approach to generate successful behavior for the real robot, we transfered the learned policy from the simulator onto the real robot. The result was that all ten out of ten episodes in the scoring test were completed successfully. The average duration was 153s. Thus, the policy learned in the simulator already outperforms the hard coded behavior by 20s [14].

In another experiment, we evaluated the learning process on the real robot. We used the proposed concept of imitation and memory as it was described by Setup iv) above. As can be seen in Fig. 4, the success rate increases quickly up to 70% after just 30 trials. The final performance is a success rate of 85%. Although the curve is not yet stable at this point, the results from the simulations suggest that this performance level can be maintained and maybe further improved.

8 Related Work

Machine learning has been widely investigated in recent works on robotics, intelligent agents or control systems. Classical approaches are increasingly combined with psychological mechanisms like imitation, curiosity, selective attention, and memory.

A well-known example for the successful application of RL techniques is the backgammon computer developed by Tesauro [2]. It consist of a feed-forward neural network, which is trained by playing against itself. TD-backgammon outperformed all

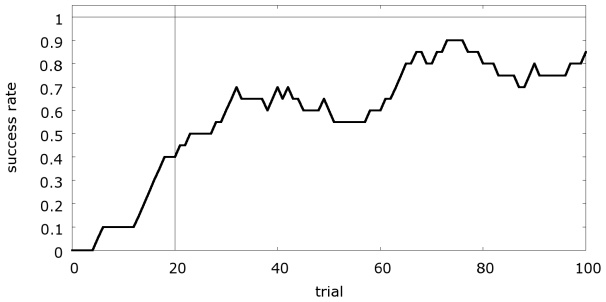

Fig. 4. Learning with the real robot. The proposed concepts of imitation and memory as well as function approximation lead to a quick acquisition of successful behavior.

commercial backgammon programs available at that time. The extensive training was essential part of this great success. The program was trained in up to 200,000 matches against itself.

Learning with memory can be used to overcome the hidden-state problem of non-Markovian environments. Algorithms like U-tree [17] or HQ-learning [18] integrate past information into the current state. As memory is only used within an episode, this can be seen as a form of short-term memory. Our approach does not aim at the solution of the hidden state problem within an episode but rather at preserving experience for later access. This corresponds to episodic memory. Our work is closely related to the fitted Q iteration algorithm [19], where RL is applied in batch mode to a large set of single observations. An observation is a four-tuple (s, a, s', r).

Different concepts of guided exploration have been proposed to accelerate RL methods. Reinforcement-driven information acquisition (RDIA) [20] combines knowledge from information theory with RL to model curiosity. Experiments with table-based Q-learning in a simulated environment show that transition probabilities can be learned much faster than with random exploration. Another form of guided exploration is coaching. The RATLE algorithm [21] uses Q-learning with a feed-forward neural network and allows to process external advice in form of rules. This is done by translating the rules into neural units and inserting these new units directly into the network. In [22] imitation is used to solve a maze problem.[2] The learning agent has an "innate" imitation behavior, which consists in following a teacher. Experiments in simulations show that increasingly complex mazes can be solved. The results are not compared to other approaches.

The successful application of RL methods to robotic soccer has been recently demonstrated by the team Brainstormers Osnabrück, the 2005 World Champion in the RoboCup 2D-Simulation-League. Their research focuses on the use of RL techniques for multi-agent systems [3]. Classical RL with feed-forward networks is used to learn basic skills that are combined to more complex behaviors [23]. The application of RL to real-world soccer robots is investigated in [24]. First, basic skills are learned via RL. Then, a policy is learned as well that chooses among the basic skills according to the cur-

[2] The problem is called a maze problem in the original publication. However, it is rather a corridor. So the problem is not finding the exit, but to follow a given path without collisions.

rent situation. Thus, a two-level hierarchy is used. The agent successfully learns good behavior in a simulated environment. Applied to the real robot, the behavior does not reach the performance of an explicitly programmed solution. The results are compared to a robot, whose top speed has been reduced to $\frac{1}{3}$ of the usual top speed.

9 Conclusions

In this paper, we presented several techniques to reduce the amount of training data for RL. The main idea was to build extensive knowledge from few experiences. This is crucial for the application of RL methods to real-world scenarios.

We use imitation to replace the random exploration of the huge state and action space with a guided exploration. In our approach, the agent has full access to experiences of a teacher, which has the same state and action space and gets identical rewards. Perceptions, actions, and rewards of the experienced agent are stored and can be accessed and reused later. Similarly, own experiences are stored and reevaluated later. This dramatically reduces the training expenses. Classical RL methods process the current observation and discard it immediately. This way, valuable information might be lost, since it cannot be correctly assessed at the moment of the experience. We let the agent repeatedly reprocess past experiences to avoid this problem.

In addition, the quick generalization of similar situations while preserving the possibility to distinguish between different situations, essentially contributes to the acceleration of the learning process. Coarse coding with binary features allows locally constant function approximation. In this case, inputs that activate identical features are treated identically. Our function approximation with continuous features is locally linear. Inputs that activate identical features remain discriminable by the different intensities of the activation. This way, generalization and discrimination can be better combined.

As the experimental results show, fundamental soccer skills can be learned using RL in simulation. The approach also works with a real humanoid robot on the soccer field. The given task is accomplished quickly and reliably. Although the training with the real robot requires more time than the training in simulation, it stays within a reasonable limit. We also showed that the learned behavior in the simulator can be directly used by the real robot and yields good results.

Acknowledgment

This project is supported by the German Research Foundation (DFG), grant BE2556/2-1.

References

1. Russell, S., Norvig, P. In: Artificial Intelligence: A Modern Approach. 2nd edn. Prentice-Hall, Englewood Cliffs, NJ (2003)
2. Tesauro, G.: Practical issues in temporal difference learning. In: Proc. of Conference on Advances in Neural Information Processing Systems. Volume 4., Morgan Kaufmann Publishers, Inc. (1992) 259–266

3. Riedmiller, M., Merke, A., Nowak, W., Nickschas, M., Withopf, D.: Brainstormers 2003 - team description. In: Proceedings of Robocup 2003, Padua, Italy (2003)
4. Asada, M., Ogino, M., Matsuyama, S., Ooga, J.: Imitation learning based on visuo-somatic mapping. In: Proc. of International Symposium on Experimental Robotics (ISER). (2004)
5. Bentivegna, D.C., Atkeson, C.G., Cheng, G.: Learning tasks from observation and practice. Journal of Robotics & Autonomous Systems **47**(2-3) (2004) 163–169
6. Dillmann, R.: Teaching and learning of robot tasks via observation of human performance. Journal of Robotics & Autonomous Systems **47**(2-3) (2004) 109–116
7. Ito, M., Tani, J.: Joint attention between a humanoid robot and users in imitation game. In: Proc. of the Int. Conf. on Development and Learning (ICDL). (2004)
8. Mataric, M.J.: Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In Dautenhahn, K., Nehaniv, C., eds.: Imitation in Animals and Artifacts. MIT Press (2002)
9. Schaal, S.: Learning from demonstration. In: Proc. of the Conf. on Neural Information Processing Systems (NIPS). (1997)
10. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine Learning **3** (1988) 9–44
11. Hinton, G.E.: Distributed representations. Technical Report CMU-CS-84-157, Carnegie-Mellon University, Computer Science Department, Pittsburgh, PA (1984)
12. Peng, J., Williams, R.J.: Incremental multi-step Q-learning. In: Proceedings of the 11th International Conference on Machine Learning. (1994) 226–232
13. Jaakkola, T., Jordan, M.I., Singh, S.P.: Convergence of stochastic iterative dynamic programming algorithms. In Cowan, J.D., Tesauro, G., Alspector, J., eds.: Proc. of 7th Conference on Advances in Neural Information Processing Systems, Morgan Kaufmann Publishers, Inc. (1994) 703–710
14. Behnke, S., Müller, J., Schreiber, M.: Playing soccer with RoboSapien. In: Proceedings of 9th RoboCup International Symposium. (2005)
15. Aloul, F.A., Markov, I.L., Sakallah, K.A.: Efficient symmetry breaking for Boolean satisfiability. In: International Joint Conference on Artificial Intelligence. Volume 3., AAAI (2003) 271–282
16. Withopf, D., Riedmiller, M.: Effective methods for reinforcement learning in large multi-agent domains. Information Technology Journal **47**(5) (2005)
17. McCallum, A.: Learning to use selective attention and short-term memory in sequential tasks. In Maes, P., Matari, M., Meyer, J.A., Pollack, J., Wilson, S., eds.: From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior, Berlin, The MIT Press (1996) 315–324
18. Wiering, M., Schmidhuber, J.: HQ-learning. Adaptive Behavior **6**(2) (1997) 219–246
19. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. Journal of Machine Learning Research **6** (2005) 503–556
20. Storck, J., Hochreiter, J., Schmidhuber, J.: Reinforcement driven information acquisition in non-deterministic environments. In: Proc. of ICANN'95. Volume 2., Paris (1995) 159–164
21. Maclin, R., Shavlik, J.W.: Incorporating advice into agents that learn from reinforcements. In: Proc. of 12th National Conference on Artificial Intelligence. (1994) 694–699
22. Demiris, J., Hayes, G.: A robot controller using learning by imitation. In: Proceedings of the 2nd International Symposium on Intelligent Robotic Systems, Grenoble, France (1994)
23. Riedmiller, M., Merke, A., Meier, D., Hoffmann, A., Sinner, A., Thate, O., Ehrmann, R.: Karlsruhe Brainstormers — A reinforcement learning approach to robotic soccer. Lecture Notes in Computer Science (2001)
24. Dietl, M.: Reinforcement-Lernen im Roboterfußball, *Diplomarbeit* (in German), Albert-Ludwigs-Universität Freiburg (2002)