

# Karlsruhe Brainstormers - A Reinforcement Learning approach to robotic soccer

A. Merke and M. Riedmiller

Institut für Logik, Komplexität und Deduktionssysteme  
University of Karlsruhe, 76131 Karlsruhe, Germany

**Abstract.** Our long-term goal is to build teams of agents where the decision making is based completely on Reinforcement Learning (RL) methods. It requires an appropriate modelling of the learning task and the paper describes how robotic soccer can be seen as a multi-agent Markov Decision Process (MMDP). It discusses how optimality of behaviours of agents can be defined and what difficulties one encounters in developing concrete algorithms which are supposed to reach such optimal agent/team policies. We also give an overview of already incorporated algorithms in our 'Karlsruhe Brainstormers' simulator league team and report some results on learning of offensive team behaviour.

## 1 Introduction

The robotic soccer domain became very popular during the last few years. Since 1997 there are annual world championships to measure the progress of playing quality between the different approaches pursued around the world. Our group takes the approach of viewing the robotic soccer as MMDP while using techniques from Reinforcement Learning. Section 2 introduces basic definitions of optimality. In section 3 we show what difficulties have to be mastered to use the theoretical optimality criteria defined in section 2. Finally in section 4 we discuss some practical work in the development of our Robocup competition team 'Karlsruhe Brainstormers'.

## 2 Robotic soccer as a Reinforcement Learning problem

A MMDP is defined as a tuple (cf. [3])

$$M_n := [S, A, r, p],$$

where  $S$  is the space of all states,  $A$  is a cartesian product of action sets  $A = A_1 \times \dots \times A_n$  and  $p$  denotes the state transfer probabilities, i.e.  $p(\cdot | s, a)$  is a probability measure on  $S$  depending on the current state  $s$  and the joint action  $a = (a_1, \dots, a_n)$ . In this paper we will concentrate on two special MMDPs cases: cooperative and zero sum MMDPs. We will combine them to characterise the

robotic soccer environment<sup>1</sup>. In the cooperative MMDP case all agents get the same reinforcement i.e.  $r_1 = \dots = r_n$ . In the zero sum MMDP we have two agents with reversed reinforcements,  $r_1 = -r_2$ .

To define optimality we need the notion of a (total) policy. A total deterministic policy  $\pi$  is a mapping

$$\pi : S \rightarrow A = A_1 \times \dots \times A_n$$

It is important to see that this definition concerns the behaviour of all agents simultaneously and that the policy of a single agent  $i$  can be seen as a projection  $\pi_i : S \rightarrow A_i$  in  $\pi = (\pi_1, \dots, \pi_n)$ . Every agent concerns the value  $v_i[\pi]$  of a total policy, which is a expectation of the sum of its future reinforcement signals

$$v_i[\pi](s) = v_i[\pi_1, \dots, \pi_n](s) = E \left[ \sum_k r_i(s_k, \pi(s_k)) \mid s_0 = s \right].$$

In the cooperative MMDP case we have  $v_1[\pi] = \dots = v_n[\pi]$  for all  $\pi$ . All we need is an optimal total policy  $\pi^*$  which has the maximal value

$$\pi^* = \arg \max_{\pi} v_i(\pi).$$

The existence of  $\pi^*$  is guaranteed ([6]), but this existence doesn't say anything how such a  $\pi^*$  can be computed. We will discuss this peculiarity further in section 3. In the two agent zero sum MMDP we have a total policy  $\pi = (\pi_1, \pi_2)$ . Borrowing from game theory we can define a minimax policy for each agent. We look for a total policy  $(\pi_1^*, \pi_2^*)$  which fulfils

$$v_1[\pi_1^*, \pi_2^*] = \max_{\pi_1} \min_{\pi_2} v_1[\pi_1, \pi_2] = \min_{\pi_1} \max_{\pi_2} v_1[\pi_1, \pi_2] = -v_2[\pi_1^*, \pi_2^*]$$

Existence of such a pair  $(\pi_1^*, \pi_2^*)$  can be always assured (see [3]). Such policies are in general no longer deterministic, i.e. they map from  $S$  to a probability distribution over  $A$ .

Let us now turn our attention to robotic soccer. Here we have two teams which perform a zero sum game. Each agent in the same team has the same objective: to score more goals then the agents from the opposite team. This can be expressed as zero sum MMDP of 2 cooperative teams

$$M = (S, A = B_1 \times \dots \times B_m \times C_1 \times \dots \times C_n, r, p)$$

where  $r = (r_b, \dots, r_b, r_c, \dots, r_c)$  and  $r_b = -r_c$  express the team respectively zero sum character. Using our previous optimality definitions it is now a straight forward task to define a total optimal policy in the case of two competitive teams (with cooperation within every team). We speak of an optimal total policy  $(\pi_1^*, \dots, \pi_m^*, \phi_1^*, \dots, \phi_n^*)$  if

$$v_b[\pi_1^*, \dots, \pi_m^*, \phi_1^*, \dots, \phi_n^*] = \max_{(\pi_1, \dots, \pi_m)} \min_{(\phi_1, \dots, \phi_n)} v_b[\pi_1, \dots, \pi_m, \phi_1, \dots, \phi_n]$$

---

<sup>1</sup> A more extensive presentation can be found in [5].

### 3 Learning in independent distributed systems

In section 2 we discussed how an optimal policy could be defined. What we did not discuss were the difficulties to find such a policy in a distributed way. We will demonstrate this in the cooperative MMDP case considering a possible solution in the zero sum MMDP with two teams as even harder to reach.

To illustrate the problem of action choice coordination imagine a small 1 state (deterministic) system with 2 cooperative agents, each having two actions:

$$M_2 = [\{s\}, \{a_1, a_2\} \times \{b_1, b_2\}, r = (r_1, r_2), p]$$

The reward function is for both agents equal  $r_1 = r_2$  and given by  $r_1(s, (a_1, b_1)) = r_1(s, (a_2, b_2)) = 2$  and equal 0 in connection with the remaining actions. It is easy to see that we have two optimal policies  $\pi^*(s) = (a_1, b_1)$  and  $\hat{\pi}^*(s) = (a_2, b_2)$ . But if agent 1 decides to take the projection  $\pi_1^*(s) = a_1$  with respect to  $\pi^*$  and agent 2 takes the projection  $\hat{\pi}_2^*(s) = b_2$  with respect to  $\hat{\pi}^*$  we get a total policy  $\pi = (\pi_1^*, \hat{\pi}_2^*)$  which isn't optimal anymore. The problem lies in the lack of coordination. In the rest of this section we will use three different agent types to demonstrate problems of multi-agent learning and coordination

- White Box Agents (WBA) also called Joint Action Learners (see [2]) are agents which have knowledge of all the joint actions  $a = (a_1, \dots, a_n)$  performed in every step.
- Black Box Agents (BBA) also called Independent Learners are agents which just know about their own actions. They aren't really aware of the other agents, all the influence through actions of other agents could also be interpreted as environmental noise.
- Gray Box Agents (GBA) are Black Box Agents which can communicate with other agents. It is no further specified how much information can be exchanged. If no communication takes place we remain having BBAs, but if every agent communicates his action we get the special case of WBAs. But we can do even more with communicating agents, we can exchange information about their future intentions.

In figure 1 we can see the connections between the agents models. The WBA case can be identified with the single agent MDP situation. We will demonstrate this using the terminology of Q-learning ([11, 1]). To this end we define

$$q^*(s, a) := r(s, a) + \sum_{j \in S} p(j | s, a) v[\pi^*](j)$$

in which  $\pi^*$  is an optimal total policy. The value of  $q^*(s, a)$  represents the expected reward<sup>2</sup> for using action  $a$  in state  $s$  and pursuing an optimal total policy afterwards. The knowledge of  $q^*$  amounts to knowing all optimal policies as we have

$$\pi^*(s) \in \arg \max_{a \in A} q^*(s, a)$$

---

<sup>2</sup> We don't distinguish different  $r_i$  here because all agents get the same rewards.

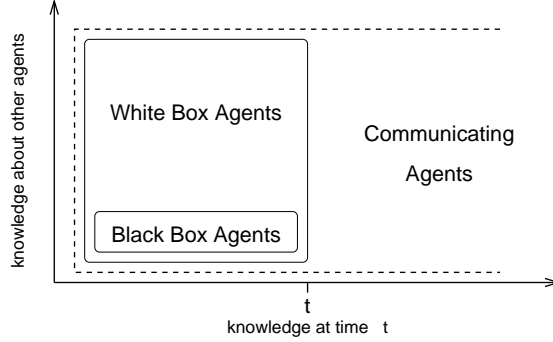


Fig. 1. Illustration of potential knowledge during learning in the different models.

for an arbitrary optimal policy  $\pi^*$ . Using Q-learning we are able to compute the  $q^*$  function provided we get access to enough tuples

$$\langle s^t, a^t, r^t, s^{t+1} \rangle$$

where  $s^{t+1}$  is the successor state after using the joint action  $a^t$  in state  $s^t$  while getting reward  $r^t = r(s^t, a^t)$ . Using Q-learning every WBA can build up his own  $q^*$  function. The problem of choosing an unique optimal total policy  $\pi^*$  remains, so that each agent can use the projection  $\pi_i^*$  of the same optimal total policy. We will not discuss the possibilities of attaining this goal but we just mention that theoretically sorting all optimal total policies and taking the first will do the job.

The major drawback using WBA and joint actions  $a = (a_1, \dots, a_n)$  comes from the fact that the number of such actions grows exponentially with the number of agents. Furthermore such learning is inflexible with respect to changing the number of participating agents. These reasons are the main motivation for using the considerably weaker model of BBAs. But in this model all that we can hope for is to be able to compute

$$q_i^*(s, b) = \max_{a \in A, a_i = b} q^*(s, a)$$

for each agent  $i$ . This is the maximal possible expected reward if agent  $i$  is using action  $b$  in state  $s$ . Computing the  $q_i^*(s, b)$  constitutes the first problem which has to be solved. The second problem has to do with a coordinated choice of actions from  $\arg \max q_i^*$ . If for example  $\{b, b'\} = \arg \max q_i^*$ , then we have somehow to decide which of these two actions we'll actually take. Both problems can be solved if the underlying MMDP is deterministic (see [4]). In the case of probabilistic state transitions and the pure BBA scenario, it can be shown, that the computation of the  $q_i^*(s, b)$  values is in general impossible (cf. [5]).

There remains the question of how to deploy communication or some other coordination scheme to solve both problems using BBA agents (i.e. to use some

incarnation of GBAs). As far as we know no such algorithm has been published yet. In the next section we will present some empirical work, which uses a sort of implicit coordination while working with Black Box Agents.

## 4 Our RL approaches to robotic soccer

In [8] we already published some of our work related to single agent reinforcement learning. This includes successful deployment of moves to learn several basic agent behaviors. All these moves were learned using reinforcement learning with neural nets as function approximators.

On the tactical or team behaviour level we didn't use reinforcement learning until very recently. In particular our last year team used planning for players with the ball and a priority list of moves for players without the ball (see [8] for more details). As we already mentioned we work with higher level moves on the tactical level. As the first step of developing a whole team policy with RL we started with the attack behaviour.

We use 7 attackers against 7 defenders and one goalie. At the moment we put all positions of the players as input to an neural network (34 dimensions). Simultaneously we also work on a feature extraction scheme, which will enable us to ignore the exact number of defenders and to lower the dimension of the encoding input vector.

Each of our attackers without ball has 10 actions to choose from. He can just go to one of 8 direction, go to his home position or try to intercept the ball. To go on we must say a little bit more about the home positions concept, as this is the main coordinating scheme for our agents. Our team can use different formations (for example 4-3-3) which are stretched over the soccer field with respect to the ball position and the offside lines. As each of our attackers has the choice to go towards his home position we have implicitly a mechanism which makes it easier to avoid two attackers going to the same position. If one attacker gains the ball he uses our planning algorithms which tries to find the shortest pass chain to score a goal.

Our algorithm learns along trajectories which lead to a goal or to the loss of the ball. In the first case we get a positive reward in the second a negative cost. To update the value for every state along a trajectory we use  $TD[1]$  (see [10]) to propagate the reward/cost along the whole trajectory. The update of the neural networks are performed with a variant of the backpropagation algorithm called RPROP (see [7]). The results of this learning scheme are very promising. We used our last year team attack and the attack of the FCPortugal team<sup>3</sup> for comparison and our last year defence as a benchmark. The results are presented in the following table

	Brainstormers2000	$TD[1]$ attack
success rate	13%	20%

<sup>3</sup> Our team was runner up and the FCPortugal team was winner of the Simulation League Robocup World Championship in Melbourne 2000.

## 5 Summary

The soccer domain can be modelled as a multi-agent Markov Decision Process (MMDP). To deal with the multi-agent aspects, we pursue both the investigation of theoretically founded distributed RL algorithms plus the empirical/heuristically motivated way of modified single-agent Q-learning. We report very promising results in using learning of a coordinated offensive behaviour. Still a lot of research questions are open, for example the dealing with partial observability of state information, the definition of theoretically founded and efficient distributed learning algorithms (including opponent modelling) and the search for appropriate features.

### 5.1 Acknowledgements

We would like to thank our students (D. Eisenhardt, A. Hoffman, M. Nickschas, A. Sinner, O. Thate, D. Withopf) for their active involvement in the development of our team 'Karlsruhe Brainstormers'. Also many thanks go to former members of our development team (S. Buck, S. Dilger, R. Ehrmann, D. Meier). Finally we thank the CMU-Team 99 for providing parts of their world model.

## References

1. Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-dynamic programming*. Optimization and neural computation series ; 3. Athena Scientific, 1996.
2. Caroline Claus and Craig Boutilier. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *IJCAI*, 1999.
3. Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer, 1997.
4. M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of International Conference on Machine Learning, ICML '00*, pages 535–542, Stanford, CA, 2000.
5. A. Merke. Reinforcement Lernen in Multiagentensystemen. Master's thesis, Universität Karlsruhe, 1999.
6. Martin L. Puterman. *Markov decision processes : discrete stochastic dynamic programming*. Wiley series in probability and mathematical statistics : Applied probability and statistics. Wiley, 1994.
7. M. Riedmiller and H. Braun. RPROP: A fast and robust backpropagation learning strategy. In Marwan Jabri, editor, *Fourth Australian Conference on Neural Networks*, pages 169 – 172, Melbourne, 1993.
8. M. Riedmiller, A. Merke, D. Meier, A. Hoffmann, A. Sinner, O. Thate, C. Kill, and R. Ehrmann. Karlsruhe brainstormers - a reinforcement learning way to robotic soccer. In A. Jennings and P. Stone, editors, *RoboCup-2000: Robot Soccer World Cup IV, LNCS*. Springer, 2000.
9. P. Stone, R. S. Sutton, and S. Singh. Reinforcement learning for 3 vs. 2 keepaway. In *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, 2001.
10. R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
11. Christopher J.C.H Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, 8:279–292, 1992.