Brainstormers 2002 - Team Description

M. Riedmiller, A. Merke, A. Hoffmann,
M. Nickschas, D. Withopf, and F. Zacharias

Lehrstuhl Informatik I, Universität Dortmund, 44221 Dortmund, Germany

Abstract. The main interest behind the Brainstormers' effort in the robocup soccer domain is to develop and to apply machine learning techniques in complex domains. Especially, we are interested in reinforcement learning methods, where the training signal is only given in terms of success or failure. Our final goal is a learning system, where we only plug in 'win the match' - and our agents learn to generate the appropriate behaviour. Unfortunately, even from very optimistic complexity estimations it becomes obvious, that in the soccer domain, both conventional solution techniques and also advanced today's reinforcement learning techniques come to their limit - there are more than $(108 \times 50)^{23}$ different states and more than $(1000)^{300}$ different policies per agent per half time. This paper describes a modular approach of the Brainstormers team to tackle this complex decision problem at different levels.

1 The Architecture

The environment of the Soccerserver testbed does confront us with 3 major levels of difficulty. These are: 1. maintenance of an up to date world model, 2. development of individual abilities for the participating agents 3. efficient combinations of agent abilities to a team winning strategy. The architecture of our agent does reflect this subdivision into different

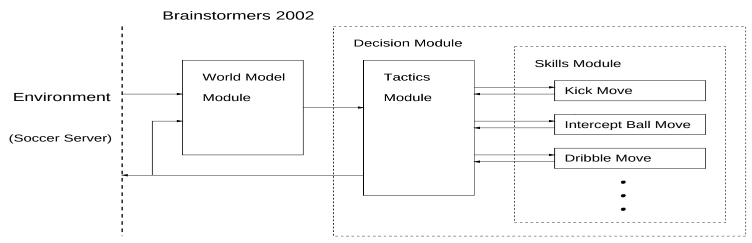

Fig. 1. Architecture of Brainstormers 2002 Agent

levels, cf. figure 1. To this end the agent is divided into independent modules, which are loosely coupled. Information flow between the modules takes place along the sketched arrows.

The main purpose of the world model module is the processing of data from the environment and creation of an approximated view of the current Soccerserver state[1]. The decision module is now confronted with a time discrete (markov) decision process. In the ideal case it gets a full description of the current state $s(t)$ and generates a command $a(t)$.
The procedure of generating a command $a(t)$ is subdivided into two steps. The tactics module abstracts from basic Soccerserver commands (i.e. *kick, turn, dash*) and decides which move within the skills module should be applyed. A move generates then the basic command $a(t)$, which can be sent back to the server.

2 Reinforcement Learning of Moves

A move is a sequence of basic actions, that transforms a current situation $s(0)$ into a new situation $s(t)$ some time steps later. The resulting situation is one of a set of terminal states $\mathcal{S}^f$, which might be either positive/desired outcomes ($\mathcal{S}^+$) or negative/undesired situations ($\mathcal{S}^-$). The move ends, if either a terminal state is reached ($s(t) \in \mathcal{S}^f$), or the time exceeds a certain limit ($t > t_{max}$).
For example, the move *intercept-ball* terminates if either the ball is within the player's kick range ($\mathcal{S}^+$) or if it encounters a situation, where it is no more possible for the player to reach the ball ($\mathcal{S}^-$).
Since each move has a clearly defined goal, it is now possible to find sequences of basic commands, that finally reach the defined goal. This can be done either by conventional programming, or, as it is the case in our approach, by reinforcement learning methods. In both cases, it is important that the goal of a move is reasonably chosen, that means that the solution policy is not too complex (e.g. a move 'win that game' would be desirable but its implementation will be as complex as the original problem).
The above move definition directly allows to formulate the problem of 'programming' a move as a (sequential) Reinforcement Learning (RL) problem. The general idea of reinforcement learning is that the agent is only told, what the eventual goal of its acting is. The agent is only provided with a number of actions, that it can apply arbitrarily. In the course of learning, it should incrementally learn a (closed-loop) policy, that reaches the final goal increasingly better in terms of a defined optimisation criterion. Here we apply Real-Time Dynamic Programming methods [1], that solve the problem by incrementally approximating the optimal value function in repeated control trials. A feedforward neural network is used to approximate the value function [4].
In the current version of our agent five main moves use reinforcement learning techniques: 1. a kick move which can accelerate the ball to play it with high velocity in a desired direction, 2. an intercept ball move that enables the agent to intersect efficiently the trajectory of a rolling ball, 3. a dribble move that makes it possible to run without losing control over the ball, 4. a goto position move which reaches a particular position

[1] This procedure is somewhat technical and is not discussed further in this paper.

while avoiding collisions with other players and finally 5. a stop ball move which is a supplementary move specialised in stopping high velocity balls. See [6] for more information about move learning in our team.

3 Reinforcement Learning of Team Strategies

Having high quality moves is an important requirement for a successful team strategy. But as the attention by moves is put on individual agent goals (like intercepting the ball) a higher level learning scheme is required to reach team oriented goals. As we already stressed in [3, 2] the team learning task cannot be described by a markov decision process anymore, one requires multi agent decision processes (MMDP). In this paper we present one possibility to learn with individual learners (IL)[2] in a MMDP. Using IL one looses theoretical guarantees about reaching optimality in learning, but as we show in this paper, the results are encouraging to further explore in this direction.
We conducted experiments with different numbers of attackers against some number of defenders (for example 7 attackers versus 8 defenders or 3 attackers versus 4 defenders). In the following we report about experiments with 3 attackers against 4 defenders, as this problem is sophisticated enough to show the general tendency. In Figure 2 we pictured

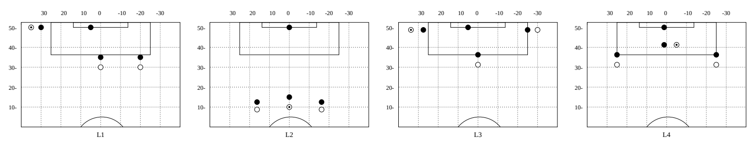

Fig. 2. Starting positions of learning, dotted player owns the ball.

four start position from which we start our learning trajectories. The number of visited states is actually much larger due to the exploration of the agents and the stochastic nature of all actions. The learning is done in epoches. We start with a random policy which chooses for each attacker one of 10 actions. The action set comprises going to one of eight directions, moving towards one's home position and intercepting the ball. The defenders pursue a fixed policy, that of the Brainstormers team 2000 (runner up in the Melbourne competition, in which they got just 4 goals in 11 tournament games). This defending strategy has a parameter of using an aggressive offside trap, which is switched off during learning.

The learning is now done using the following scheme

1. set example set to $E = \emptyset$. Each example in E comprises of a situation and the corresponding reward.
2. run current policy until approximately 10 successful trajectories could be stored. Discard not successful trajectories, but use situations

[2] Individual learners were called black box agents in [3].

where the ball was stuck as immediate negative examples (i.e. just the final situation is punished, not the whole trajectory).
3. add situations along a successful trajectory to E. The cost of each situation depends on the time from that situation to the goal state. Exploit symmetries during generation of examples from trajectories.
4. use the full set E for computing the gradient in RPROP learning of the neural network for 5000 iterations.
5. return to step 2

At the moment we use all positions of the players and the position and velocity of the ball as input to a neural network (which has 18 input dimensions, 10 hidden units). 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. Once the example set E is available learning is done in a supervised manner. The updates of the neural network are performed with a variant of the backpropagation algorithm called RPROP (cf. [5]). In step 3 of the above algorithm one could object that the set E should be emptied before each new RPROP learning epoch. But keeping the old examples prevents the agent from forgetting former successful (but maybe not time optimal) trajectories. We observed that it is better to accumulate more examples (with partially slightly wrong values, but the right tendency) to reach better generalisation results then to discard such valuable experience.

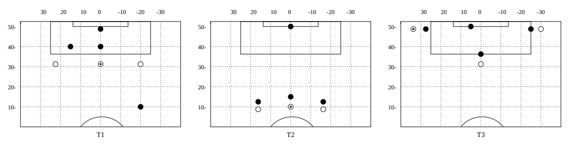

Fig. 3. Testing positions of learning, dotted player owns the ball.

The results of our learning scheme are very promising. The main questions of the above approach are
1. are we able to generalise to different situations on the field.
2. are we able to generalise if playing against another teams.

Answering the first question is simple: consider different representative situations and test them. In our presentation we concentrate on just 3 further situation depicted in figure 3, which are different enough to show the actual tendency. Answering the second question is much more difficult. This can just be indicated by using another set of strong teams[3], and we do this here by using the same sophisticated defending team, but with the offside trap turned on. This does not claim universality but just indicates that the learned policy is stable against decisive opponent strategy changes. The following table summerises our results by showing success rates in reaching a goal, and compares them to the programmed offense used in the Melbourne competition.

[3] Confer [3, 2] for more details on this problem

	Against known defense				Against unknown defense			
	learned offense		Melbourne offense		learned offense		Melbourne offense	
	goal	stuck	goal	stuck	goal	stuck	goal	stuck
L1	0.585	0.025	0.01	0.925	0.645	0.03	0.0	0.97
L2	0.43	0.005	0.4	0.08	0.225	0.145	0.01	0.505
L3	0.485	0.0	0.01	0.94	0.45	0.04	0.0	0.965
T1	0.605	0.0	0.4	0.04	0.655	0.01	0.31	0.205
T2	0.55	0.01	0.485	0.14	0.39	0.035	0.14	0.0
T3	0.52	0.005	0.515	0.08	0.445	0.05	0.145	0.415

4 Conclusion

The current version is an intermediate step within our Brainstormers concept of a learning agent. The final goal is to have an agent, which has learned all of its decision behaviour by (reinforcement) learning. However, until then a lot of work has to be done in the field of multi-agent RL, on Semi- Markov Decision Processes, partially observable domains (POMDPs) and on large-scale RL problems. Some of very recent RL ideas have already been successfully realised. For example, our moves-concept is closely related to Sutton's et.al 'options'-framework [7]. Also our experiments with learning of an attacking team strategy are very promising and can be extended to other game situations in the future.

References

1. A. G. Barto, S. J. Bradtke, and S. P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72:81–138, 1995.
2. A. Merke. Reinforcement Lernen in Multiagentensystemen. Master's thesis, Universität Karlsruhe, 1999.
3. A. Merke and M. Riedmiller. Karlsruhe brainstormers - a reinforcement learning way to robotic soccer. In P. Stone, editor, *RoboCup-2001: Robot Soccer World Cup V, LNCS*. Springer, 2001.
4. M. Riedmiller. Concepts and facilities of a neural reinforcement learning control architecture for technical process control. *Journal of Neural Computing and Application*, 8:323–338, 2000.
5. 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.
6. 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.
7. R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.