Brainstormers 2D – Team Description 2005

M. Riedmiller, T. Gabel, J. Knabe, and H. Strasdat

AG Neuroinformatik
Universität Osnabrück, 49069 Osnabrück, 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. In particular, we are interested in applying 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 behavior. 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 the architecture of the Brainstormers team, focuses on the use of Reinforcement Learning to learn various elements of out agents' behavior, and highlights other advanced artificial intelligence methods we are employing.

1 Introduction

The Brainstormers have been participating in RoboCup's soccer simulation tournaments since 1998. While the team's implementation has been under continuous development and witnessed various changes, the underlying and encouraging research goal has always been to exploit AI and machine learning techniques wherever possible. Particularly, the successful employment of Reinforcement Learning (RL) methods for diverse elements of the Brainstormers' decision making modules has been and is our main focus as shall be detailed in the subsequent sections.

1.1 Design Principles

The Brainstormers 2D rely on the following basic principles:

- There are two main modules: the world module and the decision making module.
- Input to the decision module is the approximate, complete world state.
- The soccer environment is modelled as a Markovian Decision Process (MDP).
- Decision making is organized in complex and less complex behaviors.
- A continuously growing part of the behaviors is learned by Reinforcement Learning methods.
- Modern AI methods are applied wherever possible and useful (e.g. particle filters are used for improved self localization).

1.2 The Brainstormers Agent

The decision making process the Brainstormers Agent is based upon is inspired by behavior-based robot architectures. A set of more or less complex behaviors realize the agents decision making as sketched in Figure 1.2. To a certain degree this architecture can be characterized as hierarchical, differing from more complex behaviors, such as "no ball behavior", to very basic, skill-like ones, e.g. "pass behavior". Nevertheless, there is no strict hierarchical sub-divisioning. Consequently, it is also possible for a low-level behavior to call a more abstract one. For instance, the behavior responsible for intercepting the ball may, under certain circumstances, decide that it is better to not intercept the ball, but to focus on more defensive tasks and, in so doing, call the "defensive behavior" delegating responsibility for action choice to it.

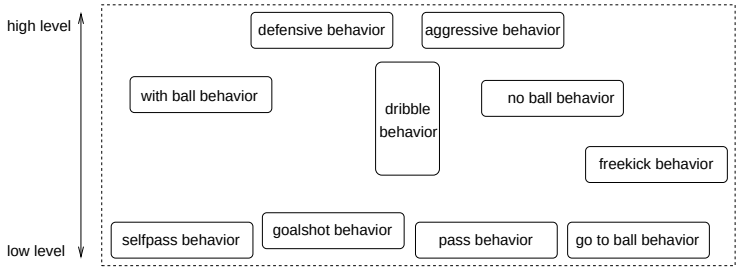

Fig. 1. The Behavior Architecture

2 Current Research Efforts

After having introduced the basics on the Brainstormers' competition team, we now want to give some particulars on the use of machine learning, and particularly Reinforcement Learning, approaches to enhance our team's capabilities.

2.1 Improved Skill Learning

As mentioned before and detailed in previous team description papers [1], many of our team's basic skills have been implemented using Reinforcement Learning approaches. These skills can be described as single Markov decision processes (MDP). Among others we currently have learnt solutions for the following low-level behaviors: intercepting the ball, going to a specific target position, kicking the ball with desired velocity and kick angle, 1-versus-1 situations, running free (in attack play mode), dribbling, passing, and scoring.

From a scientific point of view it is appealing to develop techniques to realize big parts of the agents' behavior by using machine learning methods. From a

competition point of view, however, it is also important to not lose track of searching for nearly optimal, and thus highly competitive solutions. So, in a recent work we developed an analytic, hand-coded routine for ball interception which outperformed our own RL solution by about 8.5% regarding the average number of steps to intercept the ball and which is near to the theoretical optimum (see [2] for more details on the problem of ball interception).

Motivated by this incitations, we recently started re-implementing the corresponding RL learning approach (the problem comprises a six-dimensional continuous state space), figuring out a high potential for improvements to the learning approach, and finally superseded our old neural net-based learnt solution by about 6.0%. This, admittedly, does not completely reach (ongoing work) the quality of the analytical solution, but supports the following two arguments: First, it is usually not too complicated too yield a behavior/policy of good or even high quality using RL. But, second, in many cases (particularly, when tackling real world problems of higher complexity, as in this case) significant effort is necessary in order to obtain learnt solutions that come near to or even reach the optimal solution and, hence, are competitive.

2.2 Reinforcement Learning of Team Strategies

One of our main research interest currently lies in the field of multi-agent Markov decision processes (MMDP). Learning a team strategy can be described as such a MMDP with individual learners. As already stressed in [3] there is no guarantee that an optimal strategy is learned in such a case. In this section we describe the learning algorithm that was used to learn the positioning of players without ball and also the actions of the player with ball.

The key idea behind this approach is to learn a central value function $V(s)$ for all players that describes how desirable a situation is. In other words, $V(s)$ is a mapping from a state s to a value in $[-1, 1]$. A value close to 1.0 indicates that this situation is close to success (goal), a value near -1.0 means that it is very probable to loose the ball in that situation.

In our approach a situation consists of ball position and velocity plus the position of all attacking teammates and all defending opponents. The number of teammates and opponents that are used in the state representation for the value function has to be fixed beforehand.

The learning is done in epochs. In the beginning the value function is initialized randomly so the players pursue a random strategy. Now the players play according to that strategy until five successful trajectories have been collected. If a trajectory was unsuccessful (e.g. loss of the ball) it is also stored. An example set E consisting of situations and rewards is generated from these five successful plus maximal five unsuccessful trajectories.

The terminal state s_n of a trajectory gets a reward of 1.0 ($V(s_n) = 1.0$) if it is a successful terminal state and a reward of -1.0 ($V(s_n) = -1.0$) otherwise. The reward of the other states in the trajectory depends on the distance from the terminal state. Let $s_1, ..., s_n$ be the states encountered on a trajectory. The

value of these states is then computed by:

$$V(s_i) = decay^{(n-i)} * V(s_n) \qquad i = 1...n-1 \tag{1}$$

The parameter $decay \in (0, 1)$ determines to what extent early states along the trajectories are responsible for the outcome of the trajectory. This idea is similar to the eligibility traces used by $TD(1)$.

The states together with the values are then used to train a function approximator. In our implementation we used a neural net that was trained with a variant of the backpropagation algorithm called RPROP ([4]).

The resulting net is then used in the next epoch to evaluate the different actions of a player. The action selection for a single player is done by generating an appropriate set of possible actions that the player could execute. The state after applying each of these actions is predicted using a model of the environment. The successor states are then evaluated using the value function and the best action is selected by choosing the corresponding action that leads to the successor state with the highest value.

The action set of a player that owns the ball can be arbitrarily put together by selecting different types of actions like passes, dribblings, kick-and-rushes and so on. In our current implementation we use the following action types:

- passes directly to a teammate
- passes that a teammate can intercept before the opponent
- dribblings in one of three directions

This approach is very flexible and it is very easy to use new types of actions. The only thing that has to be done is to implement a model of the environment that predicts the successor state for these actions. The model for all of these actions assumes that the opponents don't move, only the teammates that are involved in the action are modelled. One could also think of a model that predicts the opponents behavior, but this would restrict the learned strategy to opponents that follow that modelled behavior. Therefore it is better to assume nothing about the opponents, to be able to generalize to unknown opponents.

The action set for a player without the ball is very simple and consists only of moving to different positions relative to the current player position. To prevent the players from moving arbitrarily on the field, we use the concept of home positions and home areas.

A player without ball can choose from the following actions (Actions that would move the player out of his home area are not allowed):

- go in one of eight directions from current position
- go to one of eight positions around the players home position
- go to home position

One limitation of that concept is that each acting players only considers its own action set to search for an optimal action. Theoretically a player would have to consider the joint action set $a = (a_1, ..., a_n)$ of all attacking players. As already mentioned in [3] this problem can be solved if the underlying MMDP is

deterministic (see also [5]). As the soccer server environment is non-deterministic, one could change the soccer server to provide a deterministic environment and then hope that the optimal solution in this environment also works in the non-deterministic case. But as the action set size increases exponentially with the number of agents, this solution is intractable in practice.

2.3 Learning to Score

Scoring a goal or, to term it more generally, to get ahead the final few meters towards the opponent's goal is a difficult task, especially when playing against a well-organized, compact opponent defense. Hence, a highly specialized behavior is necessary for that problem. Based upon the ideas presented in Section 2.2 we also learnt a cooperative team-play behavior to be used when attacking (up to four attackers and four defenders plus goalie were considered) and having the ball in a distance of less than $20m$ to the opponent goal. Players are allowed to choose from the same sets of actions as above. Accordingly, as this behavior takes control only when the ball is nearer than $20m$ to the target goal, the number of (meta) actions, e.g. passes or dribblings, until an episode ends – either by scoring a goal or by losing the ball – is rather limited. Therefore, we here can also afford to at least partially model the behavior of involved teammates and opponents, up to maximal five action steps ahead, using conservative, i.e. pessimistic opponent models. When training is done for more complex situations, i.e. with many opponents, it can be hard to obtain positive episodes with a randomly initialized value functions. Prior experience can be carried over from learning in simpler situations to provide better chances for scoring with exploration. For lots of starting situations the learnt behavior yields better results, i.e. higher scoring success rates, than our hand-coded attack behavior used at the German Open 04.

2.4 Balancing Deliberativity and Reactivity

There are two fundamental approaches to realize autonomous agents: the reactive and the deliberative approach. A purely reactive agent receives some input via its external sensors, processes it and produces an output. On the contrary, an entirely deliberative agent has its own internal view of its environment and is thus able to build and follow its own plans, aiming to reach some specific goal state. In practice, it is, however, desirable to have a hybrid agent that represents a mixture of the former ones [6], being able to follow its own plans, but sometimes directly reacts to external events without deliberation.

The Brainstormers agent can be characterized as some hybrid agent with an accentuation on its reactive side. Although following that paradigm is suitable for the soccer simulation environment, it also has its drawbacks: Under certain circumstances, it can be advantageous to reduce reactivity, thus abandoning a short-term improvement of the current situation (e.g. getting into ball possession during the next simulation cycle), and to aim at a long-term profit instead.

A typical example in robotic soccer simulation is the problem of overcoming the opponents' offside line: For the attackers, who are assumed to be in ball possession, it is difficult to get ahead, if the defenders position themselves in a line and in so doing employ the offside rule in their interest. A reactive approach would most probably tend to repeated safe passes among teammates until, eventually, a "gap" in the opponents' defending line appears. This strategy, however, is deemed to last for a rather long time, if the opponent defense is competitive. Consequently, a goal-oriented behavior, which accepts losing the ball in case of failure but most likely improves our team's current situation significantly in case of success, seems promising.

Having this in mind, one of our current efforts aims at enhancing the Brainstormers with a set of more deliberative behavior strategies. So, for example, we target to overcome the opponent offside line by using an explicit, cooperative two-player behavior, which is based on communication between the players involved and on accurate timing. One of the inherent difficulties we have to face here, is the specification of start conditions under which the behavior will most likely produce good results. For this task we developed a probability model that reflects the probabilities that the strategy to overcome the offside line is successful. Here, we discretized the space of viable starting situations for the behavior, performed sample runs for varying parameter settings over that grid and that way generated training examples representing exemplary success probabilities. Using these, we then trained a neural network in a supervised manner in order to apply them for the determination of critic behavior-specific parameter values (e.g. target velocity and angle of the final pass when overcoming the offside line) which most likely yield highest success rates. During testing, the success rates predicted by the net for the use of behavior-specific parameters have shown to feature an average error of about 7% only. Furthermore, it became obvious that in half of all standard scenarios, in which two of our attackers tried to cooperatively overcome the opponent's offside line, a parameter setting could be found for which the success probability is higher than 50%. Using these numbers, it is not difficult to infer and implement a suitable strategy for pushing ahead.

3 Summary

In this team description paper we have outlined the characteristics of the Brainstormers team participating in RoboCup's 2D Soccer Simulation League. We have stressed that our main research focus lies on the development of Reinforcement Learning techniques and their integration into our team. Currently, we are particularly interested in extending RL approaches to multi-agent scenarios (team play) as well as in advancing the quality and accuracy of learnt behaviors to be equal or superior to hand-coded ones.

References

1. Riedmiller, M., Merke, A., Meier, D., Hoffmann, A., Sinner, A., Thate, O., Kill, C., Ehrmann, R.: Karlsruhe brainstormers - a reinforcement learning way to robotic

soccer. In Jennings, A., Stone, P., eds.: RoboCup-2000: Robot Soccer World Cup IV, LNCS. Springer (2000)
2. Stolzenburg, F., Obst, O., Murray, J.: Qualitative Velocity and Ball Interception. In: KI 2002: Advances in Artificial Intelligence, Twentyfifth Annual German Conference on AI, Aachen, Germany (2002) 283–298
3. Merke, A., Riedmiller, M.: Karlsruhe brainstormers - a reinforcement learning way to robotic soccer ii. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup-2001: Robot Soccer World Cup V, LNCS. Springer (2001) 322–327
4. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In Ruspini, H., ed.: Proceedings of the IEEE International Conference on Neural Networks (ICNN), San Francisco (1993) 586 – 591
5. Lauer, M., Riedmiller, M.: An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: Proceedings of International Conference on Machine Learning, ICML '00, Stanford, CA (2000) 535–542
6. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. Massachusetts Institute of Technology (1999)