Brainstormers 2004 - Team Description

M. Riedmiller, A. Merke, and D. Withopf

AG Neuroinformatik, Universität Osnabrck, 49069 Osnabrck, 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 the new architecture of the Brainstormers team, the improved self-localization using particle filters and the extensions of the learning algorithm to simultaniously learn with and without ball behaviors.

1 Design Principles

Both our 2D and our 3D teams are based on the following principles:

- two main modules: world module and decision making module
- input to the decision module is the approximate, complete world state
- the soccer environment is modelled as an Markovian Decision Process (MDP)
- decision making is organized in complex and less complex behaviours
- a steadily growing part of the behaviours is learned by Reinforcement Learning methods (2004: intercept, go2pos, shoot, 1-versus-1, attack play: running free, dribbling, passing, scoring)
- modern AI methods are applied wherever possible and useful (e.g particle filters are used for improved self localisation)

2 Reinforcement Learning of Team Strategies

Many of the basic skills of our team like *intercept-ball*,*goto-position* and *kick* have been implemented using Reinforcement Learning approaches. These skills can be described as single-agent markov decision processes (MDP). For more information on learning single-agent skills see [4].

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

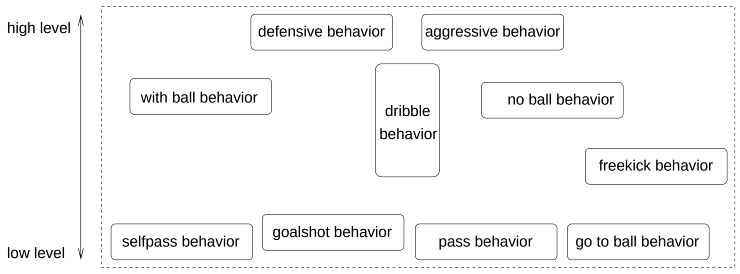

Fig. 1. The new behavior architecture

MMDP with individual learners. As already stressed in [5] 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 epoches. 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 5 succesful 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 5 succesful plus maximal 5 unsuccesful trajectories.

The terminal state s_n of a trajectory gets a reward of 1.0 ($V(s_n) = 1.0$) if it is a succesful 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 ([6]).

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 succesor states are then evaluated using the value function and the best action is selected by choosing the corresponding action that leads to the succesor 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 [5] this problem can be solved if the underlying MMDP is deterministic (see also [7]). 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.

References

1. Betke, M., Gurvits, L.: Mobile robot localization using landmarks. IEEE Transactions on Robotics and Automation **13** (1997) 251–263
2. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: Efficient position estimation for mobile robots. In: Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99). (1999)
3. Fox, D., Thrun, S., Burgard, W., Dellaert, F.: Particle filters for mobile robot localization. In Doucet, A., de Freitas, N., Gordon, N., eds.: Sequential Monte Carlo Methods in Practice, New York, Springer (2001)
4. 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)
5. 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
6. 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
7. 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