Team Description
Mainz Rolling Brains 2002

Felix Flentge, Christoph Schneider, Martin Wache

Department of Computer Science
Johannes Gutenberg-University Mainz
D-55099 Mainz, Germany

1 Introduction

In the simulation league strategic and tactical matters are gaining increasing importance as turned out at last year's tournaments. In order to handle these issues we will make use of the online coach and the coach language. Therefore, one big point will be to enhance our team's abilities to understand the coach language and to adjust our formation to the opponent tactics in the course of the game. Additionally, we will deal with other subjects, too. Besides the adaptation to the new soccer server's features, we are experimenting with learning techniques, improving our world model, and extending our visualization and debugging tool FUNSSEL for automatic execution of games (see [1] for a description of FUNSSEL).

The overall agent design remains the same as last year's and is described in more detail in [3]. The agent consists of three layers: the technical layer (e.g. server communication), the transformation layer (player skills and world model) and the decision layer. The latter uses different modules for various tasks (goal shot, pass, ballhandling, positioning and standard situation) which rate the adequateness of their respective actions and are responsible for the execution of these actions.

2 Learning Simple Behaviors

To improve our technical skills and to acquire new skills, we are working on integrating machine learning techniques. As yet, we tried a reinforcement approach as described in [2] on three different learning tasks: going to a position, keeping the ball in kick range with a minimum velocity, and placing the ball away from an opponent.

The basic idea of *reinforcement learning* (RL) is to reward or punish certain states (e.g. reaching or not reaching a position) and to find a sequence of actions maximizing the rewards over time, called *optimal policy.* A way to achieve this is to approximate a *value function* which specifies the 'value' of a state, i.e. the sum rewards (discounted by a certain factor) an agent gets when choosing this state and following the optimal policy (see [4] on RL in general). Thereafter, the agent can simulate all possible actions (provided he has a model how his actions affect the state transitions, which is true in our case) and choose the one which results in the state with the highest value.

First, we used feed forward neural networks to approximate the value function, but the convergence to a good policy was not optimal. The problem is that small local changes affect the whole network and since the network influences the choice of actions we did not get enough positive feedback to learn the task satisfyingly. As a consequence we now use an n-dimensional grid of sampling points to approximate the value function. We store the function values and all directional derivatives of each grid point. Function values between grid points are approximated using cubic spline functions. We use a gradient descent method to update the function values and the values of the directional derivatives about every hundred steps. This kind of function approximation led to a good and reliable convergence in the first two scenarios. In the third scenario (placing the ball away from an opponent) we currently work on some problems stemming from the high dimensional input space (ball position, ball velocity and direction of the opponent).

3 Positioning

3.1 Goalie Positioning

An optimal goalie position is a position from which the goalie can reach and catch a ball shot from any position where an opponent might reach the ball (either directly or as a pass). Such an optimal position may not exist, and even if it exists, it may not be feasible to calculate it. Therefore, certain simplifications are needed. First we restrict the number of opponents to at most three (assumed to be the 'most dangerous' players) the goalie should take into account when choosing his position. A good measure for a potential goalie position is the minimum distance from the goal line to the position where the goalie could catch the ball if it is shot from certain positions. To determine this distance the goalie assumes that the considered opponents shoot to either corner of the goal. He simulates his respective own movement to catch the ball and assigns the corresponding distance to the position he started his movement. This starting position is varied among a number of possible positions, and the optimal position the goalie should place himself is the position maximizing the minimum distances calculated. But he will only go to this position if enough time is left or if his old position is significantly worse than the optimal one.

3.2 Field Player Positioning

The positioning behavior of a field player is determined by his role in a so called *Tactical Team Formation* (TTF). So far, we only used two TTFs (a defensive and an offensive one) in a single game, but this year we want to adapt the TTFs in the course of a game (see next section). An agent's role in a TTF consists of his *home* position (relative to both offside lines), of a value of how much he is attracted by the ball (i.e. how much the player alters his home position towards the ball position), of the area the player should cover, and of up to three other players the player watches and whose roles he might take (at least to some extent) if these players are not able to fulfill their roles (e.g. if they are attacking an opponent). An open question is how our concept of TTFs can be expressed in the current coach language.

4 Detecting and Analyzing Team Formations for Online Adaptation

In conjunction with our DFG (German Research Foundation) project *Adaptive Situation Recognition and Machine Learning for Acquiring individual and cooperative Abilities in Multiagent-Systems (DyMAS)*, we analyze simulation league game data with *Kohonen Feature Maps* (KFM) to distinguish between typical situations and to detect certain team characteristics [6]. First experiments with KFMs to detect team formations failed because of the too high dimensional input space (all player positions of one team at one time step). For this purpose we developed an algorithm able to distinguish between different (idealized) *types* of team formations (e.g. 3-2-2-3 or 4-3-3)and to classify the actual formation (indicated by the positions of all field players at one time step) according to the formation types. The algorithm matches the actual formation onto the formation types and calculates the respective 'error' of this mapping. The formation with the smallest error is assumed to be played by the team. Preliminary results show that for most teams the algorithm can identify some typical formations.

To examine which formations are more successful than others we need criteria for successful formations. Our criterion for offensive formations is that the ball is moving towards the opponent's goal. For defensive formations the criterion is that the team regains ball possession. Obviously, these criteria are also dependent on both teams' overall skills and the formation of the opponent team. Therefore, a formation must be judged in respect to a certain team and to a certain opponent. Thus an opponent model is needed. At first, our model should only include the opponent's formation, but other characteristics will be added later (e.g. the opponent's abilities to dribble or pass). In playing with different formations against different opponents we get a probabilistic model for the success of our formations (dependent on the opponent's formation). This model can be used by the online coach to choose an adequate team formation. He updates the model by watching games (reading logfiles) and may decide to change the formation in the course of a game based on the model and his observations (e.g. if events occur that do not match his expectations).

5 Improving the Precision of the World Model

To improve the precision of our world model, we introduced a more intelligent control of the view angle and changed the timing of our agents. Up to now, we calculated our command as soon as a sense body message arrived, so we only had visual information of the cycle before. Now the agent waits until he receives visual information or a fixed amount of time has passed before calculating his command.

When we introduced our new agent in 2000 we were using only the normal view angle to get visual information in order to calculate the agent's position precisely (for details, see [1]). By not changing the view angle we encountered the problem that the agent controlling the ball did not have sufficiently good information of the ball speed. Therefore we extended our modular concept (cf. [5]) such that now each module also evaluates where it wants to look at (by specifying the view angle) and how urgent it needs the information. This has the advantage that e.g. the ball handling module can place the ball around the player while the pass module is independently looking towards a possible pass partner to improve the agent's knowledge of the partner's position.

Additional Information: The Mainz Rolling Brains belong to the Department of Computer Science of the Johannes Gutenberg-University, Mainz. Members of the Mainz Rolling Brains are: Axel Arnold, Jochen Ditsche, Manuel Gauer, Marc Hoeber, Christian Knittel, Claudia Lautensack, Christian Meyer, Tobias Reithmann, Christoph Schneider, Goetz Schwandtner, Martin Wache. Current team leaders are Felix Flentge (flentge@informatik.uni-mainz.de) and Thomas Uthmann (uthmann@informatik.uni-mainz.de). We would like to thank our former team leader Daniel Polani.
Please visit our website: `http://www.rollingbrains.de`

References

1. Arnold, A., Flentge, F., Schneider, Ch., Schwandtner, G., Uthmann, Th., Wache, M.: Team Description. Mainz Rolling Brains 2001. In: RoboCup-01. Robot Soccer World Cup V, Springer (to appear)
2. Riedmiller, M.: Concepts and Facilities of a neural reinforcement learning control architecture for technical process control. Neural Computation and Application Journal (1999) 8:323-338
3. Schappel, B., Schulz, F.: Mainz Rolling Brains 2000. In: Stone, P., Balch, T., Kraetzschmar, G. (ed.): RoboCup 2000: Robot Soccer. World Cup IV. Lecture Notes in Computer Science, Vol. 2019. Springer-Verlag, Berlin Heidelberg New York (2001)
4. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA (1998)
5. Uthmann, Th., Meyer, C., Schappel, B., Schulz, F.: Description of the Team Mainz Rolling Brains for the Simulation League of RoboCup 2000 `http://www.rollingbrains.de/mrb2000/MainzRollingBrains2000.ps`
6. Wuenstel, M., Polani, D., Uthmann, Th., Perl, J.: Behavior Classification with Self-Organizing Maps. Scientific Challenge Award, Proceedings of the Fourth International Workshop on RoboCup, Melbourne (2000)