Brainstormers 3D – Team Description 2005

Marc Halbrügge and Arne Voigtländer

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. 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. As the server for the 2005 competition is not yet finished, this paper mostly describes changes since 2004 in the architecture of the Brainstormers3d team and the self-localization.

1 Design Principles

The 3D team is based on the following principles:

- Two main modules: world module and decision making module
- Linear models are used to approximate future world states
- 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
- Modern AI methods are applied wherever possible and useful

The decision making module of the 2004 agent was based on the `agenttest` reference implementation. Different methods of a behavior class were called depending on the current playmode. Our new architecture differs significantly from this first approach.

The different skills are layered corresponding to their complexity as shown in Figure 1. The idea behind this architecture is to divide the team tactic into subtasks that can be solved more easily (divide and conquer).

To avoid complicated nested `switch` or `if` statements, we created different classes for each type of player (the top box in Figure 1). The different players use the same medium level skills like GoalShot or Intersect. These depend on the low level skills like GoToPos.

Whenever we can come up with a simple analytical algorithm, we use it. Reinforcement Learning methods are used in the remaining cases.

An example for the former case is the GoToPos behavior: The agent drives at maximum speed towards the target position until the minimal break distance is reached. Then the drive vector is inverted.

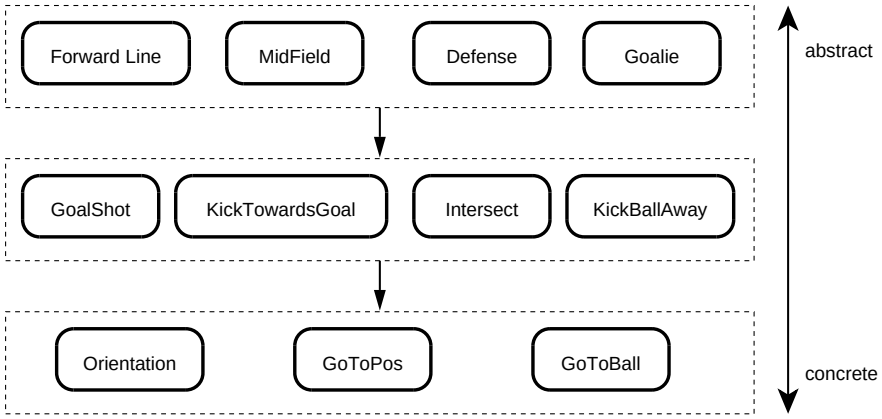

Fig. 1. The Behavior Architecture

2 Reinforcement Learning of Team Strategies

The more complex decisions, for example when to stop the positioning to perform the goalshot, are hard to program 'by hand'. Machine Learning provides algorithms that find good solutions to these problems.

The idea behind our approach is to find a value function $V(s, u)$ that describes how desirable a pair of situation and action is. V is a mapping from a state s and an action u to a value in [0,1]. A value close to 1 indicates success, a value close to 0 failure.

The value function is estimated using the Q(λ) algorithm [1, chapter 7] with λ close to 1. The state s consists of the position of the agent, the ball and the opponents. The action u consists of five different relative positions, the decision whether to kick or not and the kick force.

As we restrict the value of Q to the range [0,1], we can reinterpret it as the probability of success and use logistic regression [2] as function approximator. This is mathematically equivalent to a neural net without hidden layers.

3 Self-Localization

In order to be able to model the environment of the agent as a Markovian Decision Process [3], the actual state of the agent is a 8-dimensional vector containing means and variances of actual coordinates (disregarding the height) and velocities.

Monte-Carlo-Methods of estimation have been used to approximate the transition function of the agent's state given an action vector. We use multivariate regression as function approximator. We prefer linear and polynomial models for several reasons:

- The resulting function is easy to comprehend and verbalize, therefore testable for plausability.
- Both the model and every single regression weight can be tested for statistical significance. This way the optimal number of weights can be determined easily. The well known "bias-variance-dilemma" trading good fit for good generalization, can be solved this way.
- Repeated tests allow a stepwise procedure.

The position of the agent is determined by integrating the vision and odometric information using Kalman filter techniques [4]. When the current vision is very unlikely ($p < 0.001$) given its prediction, an orientation reaction is forced, just like a human raises his brows after being stumbled by something very unusual.

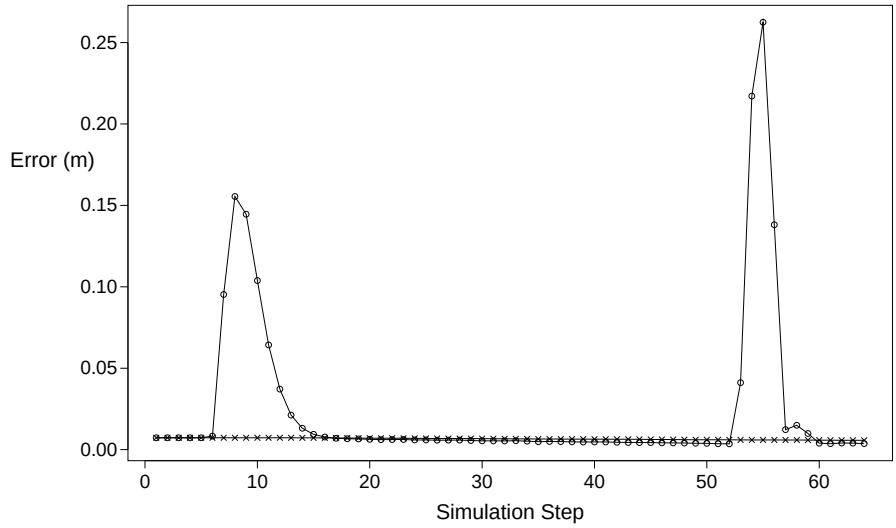

Fig. 2. Self-Localization Error

The 2004 agent only used visual information for the self-localization. The Kalman filter estimated the current velocity indirectly from the change in position between two steps. This led to big errors when the force vector changed. Figure 2 shows the localization error of an agent first standing still, then moving 15m and finally stopping again. Dots are Kalman filter errors in meters, crosses

show the error of the new self localization algorithm. There are two big peaks in the dot-line caused by changes in the acceleration. As acceleration was not modelled in the Kalman filter, it is not able to predict the next state.

Our new approach leads to much better results. Starting at a moderate error level, the integration of odometric data fits nicely into the perceived position changes. Although the old Kalman filter sometimes shows slightly lower errors (after long periods of constant movement), the new model is much better in situations that really count: when the agent stops to kick the ball.

As the agent's movement and vision will change completely in the 2005 competition, these successes wont last long. But we are confident that our approach has enough degrees of freedom to fit well into the new environment.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
2. Hosmer, D.W., Lemeshow, S.: Applied logistic regression. Wiley, New York (1989)
3. Puterman, M.L.: Markov decision processes : discrete stochastic dynamic programming. Wiley series in probability and mathematical statistics : Applied probability and statistics. Wiley (1994)
4. Kalman, R.E.: A new approach to linear filtering and prediction problems. Trans. ASME, Series D, Journal of Basic Engineeing **82** (1960) 35–45