Brainstormers 2D
Public Source Code Release 2005

M. Riedmiller, T. Gabel, and H. Schulz

Neuroinformatics Group
Institute of Mathematics and Computer Science
Institute of Cognitive Science
Universität Osnabrück, 49069 Osnabrück, Germany
{martin.riedmiller|thomas.gabel|hanschul}@uos.de

Abstract. The Brainstormers have been participating in RoboCup's soccer simulation tournaments since 1998. Ever since a number of successes could be achieved, including multiple World Vice Champion titles and the World Champion title at RoboCup 2005 in Osaka. By now, the source code of our team will be made publicly available. This way, we hope to make a contribution to the entire soccer simulation community, in particular to new teams for which, as is known, it is difficult to overcome basic problems, such as developing a reliable world model or basic skills. The document at hand aims at giving a coarse overview of our source code release directly relating to the structure of the code.

1 Overview

While the Brainstormers's implementation has been under continuous development and witnessed various changes over the years, the underlying and encouraging research goal has always been to exploit AI and machine learning techniques wherever possible. A more detailed description of our research efforts, particularly the employment of Reinforcement Learning (RL) methods, is beyond the scope of this paper. Instead, we refer to the team description paper provided for this year's RoboCup tournament in Osaka [1] as well as the team description papers of previous years [2–7]. There are also a number of articles focusing on the employment of machine learning and reinforcement learning techniques in the context of the Brainstormers and simulated robotic soccer [8–12].

The present source code release is made publicly available under the terms of the GNU General Public License (GPL)[1] and comes without any warranty. Its code has been developed in C++ under Linux.

1.1 Contents of the Source Code Release

The agent can be divided into two main modules, the world model module and the decision making module. Input to the decision making module is the

[1] http://www.gnu.org/copyleft/gpl.html

approximate, complete world state. The soccer environment is modelled as a Markovian Decision Process (MDP), where the respective action is taken with respect to the current state the agent finds itself in. Decision making is organized in complex and less complex behaviors, the structure of which will be described more thorougly below. For a big part of the agent's behavior there are also learned behaviors, which are part of this source code release, too.

The release contains the following components:

- the entire world model module, including
  - self-localization via particle filters,
  - world state memory and statistics,
  - communication with the soccer server (sensoring),
  - soccer server parameter handling
- the **entire set** of agent skills:
  - three variants of dribbling,
  - ball facing and searching,
  - going to specified positions forwards as well as backwards,
  - four variants of ball interception,
  - three variants of ball kicking,
  - ball holding,
  - goal scoring,
  - passing and self passing,
  - and several others
- numerous learned behaviours using reinforcement learning (value function representation has been done with neural networks typically, that are included as well)
- a complete behavior controller for neck control
- a complete behavior controller for view control
- a complete behavior controller for attention-to control
- useful tools, like
  - classes for handling geometric data structures,
  - a library (n++) for working with neural networks,
  - classes for communication via UDP,
  - tools for reading/writing configuration files etc.
- appropriate configuration files
- the Brainstormers online coach, called SputCoach, including its source code (version of February 2005)

Thus, this Brainstormers agent source code release is complete with respect to the current state of development of our team with one exception: On the strategic decision level the corresponding behavior has been replaced by a simple demo behavior (see below).

1.2 Utilization of Reinforcement Learning

As indicated at the beginning of this document, a main research goal of our efforts in the development of the Brainstormers is to employ reinforcement learning methods wherever applicable. In the current public source code release you will find a lot of our results in applying RL—some learning algorithms as well as behaviors that exploit the results of learning (e.g. those behaving greedily with respect to a value function represented by a neural network). The following learned modules are part of this release:

- learned skills, such as ball kicking (`NeuroKick` and `NeuroKick05`), ball intercepting (`NeuroIntercept`), going to a position (`NeuroGo2Pos`)
- learned medium-level capabilities, like passing (`NeuroWball`, `LearnWball`)
- learned team-level capabilities, such as cooperatively scoring a goal (`Score04` or positioning (`NeuroPositioning`)
- function approximators to represent state and state-action value functions (see directory `data/`)

1.3 Outlining the Brainstormers Agent

Looking at the Brainstormers agent from a top-level view, it can be subdivided into four substantial parts or controllers, respectively, which correspond to the four "types" of commands a player can send to the Soccer Server concurrently: The neck controller handles the player's head and decides where to look next, the view controller handles the player's eyes and decides on next view angle and width requested, the attention-to controller handles the player's ears and decides to which teammate should be listened next, and, finally, the main/base behavior controller handles the player's main actions. All of those are instantiated in the main file `client.c`.

Without any doubt, decision making with respect to body movements (i.e. sending kick, turn, or dash commands to the sever) as done by the behavior controller seems to be of utmost importance for the player's overall performance. 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 1.3. 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.

2 Source Code Review

The source code's documentation and commenting is of varying comprehensiveness. You will find numerous classes being documented very thoroughly, while

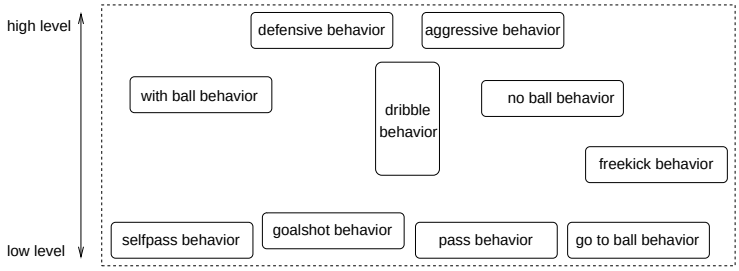

Fig. 1. The Behavior Architecture

others feature barely a line of comment. However, for the scope of this public source code release we could not accomplish/finish the comprehensive documentation of the entire package. To give the interested reader a quick start into the lots of source code, we provide an overview here.

2.1 Directory Structure

On the top level you will find 5 directories:

- `bs2k/` contains the complete sources (including makefile) of the Brainstormers agent
- `doc/` contains this short documentation of the source code release
- `lib/` contains useful tools to be used by the agent and coach
- `scripts/` contains a script to start an entire team
- `sputcoach/` contains the sources (including makefile) of the Brainstormers' online coach

Most of the details you will find in the following refer to the `bs2k/` directory since it contains the main load of the Brainstormers agent.

2.2 `bs2k/` — The Brainstormers Agent

In this directory you may find a makefile which, by invoking `make`, generates an executable binary that is stored as `artagent/BS2kAgent`. Furthermore, there are six relevant sub-directories.

- `artagent/` contains the core of the agent. Here, you will find the world model, sensor passing/buffering classes (communication with Soccer Server), parameter handling, as well as the agent's main file `client.c`.
- `basics/` contains basics. Here, you will find interfaces that facilitate the retrieval of information from the world model (e.g. classes `WSinfo` or `WSMemory`), classes that supoort logging, basic handling of formations, global definitions, simple (analytical) intercept routines, and the definition of commands (to be sent to the soccer server). Moreover, class `Tools` provides a model of the Soccer Server environment.

- `behaviors/` contains the largest part of this public code release. In `behaviors/`, higher level behaviors are to be found. Those include among others, behaviors for standard situations, for the goalie (which is seperated from the player's behaviors), no ball and with ball behaviors (demo only), passing/scoring behaviors as well as the main behavior, which is for historical reasons called `Bs03`. Of crucial importance are the two sub-directories you will find here:
  - `skills/` contains the entire set of basic behaviors, usually termed skills, the Brainstormers employ.
  - `view/` contains classes with functionality related to the player's neck, attention-to and view controlling.
- `conf/` contains configuration files that are read by the agent at start-up.
- The `data/` directory contains function approximators learned in a reinforcement learning context. Mainly, these are neural networks (multilayer perceptron feed-forward networks) that are emplyoed by some of the agent's behaviors.
- `policy/` contains helpful tools that may be mainly employed for taking strategic decisions (e.g. referring to passing and positioning).

2.3 Decision Making

Each implemented behavior—no matter if used for the attention-to, neck, view, or main functionality of the agent—must derive from its corresponding base class (`AttentionToBehavior`, `NeckBehavior`, `ViewBehavior`, `BaseBehavior`). As a consequence, each behavior is required to implement the method `bool get_cmd(Cmd &cmd)`, which takes a `Cmd` object as parameter. The implementation of `get_cmd` for each behavior is requested to fill the command parameter correspondingly, to return `true` in case a suitable command could be found, and `false` otherwise. As any behavior provides this `get_cmd`-based interface it is possible for each behavior to utilize any other available behavior to determine an appropriate command by invoking those behavior's `get_cmd` method (for example, a dribble behavior may employ the functionality provided by a go2pos or kick behavior).

The main behavior can be specified in `agent.conf` (e.g. Bs03), its `get_cmd` will be called every cycle (see main loop in `client.c`). Having a look at the corresponding file `bs03_bmc.c`, one quickly finds out that—depending on the current play mode—other behaviors are called. Most importantly, for play mode "play on" it is further distinguished whether the considered player is in ball possession or not. Depending on that, the responsibility for decision making is played on the respective strategic "no ball" or "with ball" behavior. We have provided an examplary demo no/with ball behavior (classes `NoballDemo` and `WballDemo`) that act as follows: Each player *not* in ball possession moves to a specific position as calculated by the current formation (utilizing several skill behaviors), except for the player which is nearest to the ball who always goes to the ball (also by utilizing several skill behaviors). A player *in* ball possesion will—with decreasing priority—try to shoot a goal, advance by means of dribbling, play a pass to a teammate, or (if everything fails) do nothing. Though consisting

of a few hundred lines of code only, the simple logic implemented by these policies reaches remarkable playing quality.

3 Summary

Driven by the idea to further push forward the development in the Soccer Simulation (2D) environment, we have decided to make the source code of our World Champion Team 2005, Brainstormers, publicly available. The source code is published and may be used under the terms of the GNU General Public License (GPL).

References

1. Riedmiller, M., Gabel, T., Knabe, J., Strasdat, H.: Brainstormers 2D — Team Description 2005. In: RoboCup-2005: Robot Soccer World Cup IX, LNCS. Springer (2005) To appear.
2. Riedmiller, M., Buck, S., Merke, A., Ehrmann, R., Thate, O., Dilger, S., Sinner, A., Hofmann, A., Frommberger, L.: Karlsruhe Brainstormers - Design Principles. In: RoboCup-1999: Robot Soccer World Cup III, LNCS. Springer (1999)
3. 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, Melbourne, Australia (2000)
4. 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, Seattle, USA (2001) 322–327
5. Riedmiller, M., Merke, A., Hoffmann, A., Withopf, D., Nickschas, M., Zacharias, F.: Brainstormers 2002 - Team Description. In Birk, A., Coradeschi, S., Tadokoro, S., eds.: RoboCup-2002: Robot Soccer World Cup VI, LNCS. Springer, Fukuoka, Japan (2002)
6. Riedmiller, M., Merke, A., Nowak, W., Nickschas, M., Withopf, D.: Brainstormers 2003 - Team Description. In: RoboCup 2003: Robot Soccer World Cup VII, LNCS, Padua, Italy, Springer (2003)
7. Riedmiller, M., Merke, A., Withopf, D.: Brainstormers 2004 - Team Description. In: Team Description Papers on CD-ROM for Proceedings of RoboCup 2004. Springer, Lisbon, Portugal (2004)
8. Riedmiller, M., Merke, A.: Using Machine Learning Techniques in Complex Multi-Agent Domains. In Stamatescu, I., Menzel, W., Richter, M., Ratsch, U., eds.: Adaptivity and Learning. Springer (2003)
9. Gabel, T., Riedmiller, M.: CBR for State Value Function Approximation in Reinforcement Learning. In: Proceedings of the 6th International Conference on Case-Based Reasoning (ICCBR 2005), Chicago, USA, Springer (2005) 206–221
10. Gabel, T., Riedmiller, M.: Learning a Partial Behavior for a Competitive Robotic Soccer Agent. KI Zeitschrift (2006. To appear)
11. Withopf, D., Riedmiller, M.: Comparing Different Methods to Speed-up Reinforcement Learning in a Complex Domain. In: Proceedings of the International Conference on Systems, Man, and Cybernetics, Big Island, USA (2003)
12. Withopf, D., Riedmiller, M.: Effective Methods for Reinforcement Learning in Large Multi-Agent Domains. it - Information Technology Journal **47** (2005)