FRA-UNIted — Team Description 2023

Thomas Gabel, Berkan Eren, Yalcin Eren, Fabian Sommer,
Eicke Godehardt

Faculty of Computer Science and Engineering
Frankfurt University of Applied Sciences
60318 Frankfurt am Main, Germany
{tgabel|fabian.sommer|godehardt}@fb2.fra-uas.de,
{berkan.eren|eren}@stud.fra-uas.de

Abstract. The main focus of FRA-UNIted's effort in the RoboCup soccer simulation 2D 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. In this paper, we review some of our recent efforts taken during the past year, putting a special focus on reinforcement learning experiments in the context of 2D soccer simulation.

1 Introduction

The soccer simulation 2D team FRA-UNIted is the continuation of the former Brainstormers project which has ceased to be active in 2010. The ancestor Brainstormers project was established in 1998 by Martin Riedmiller, starting off with a 2D team which had been led by the first author of this team description paper since 2005. Our efforts in the RoboCup domain have been accompanied by the achievement of several successes such as multiple world champion and world vice champion titles as well as victories at numerous local tournaments. Our team was re-established in 2015 at the first author's new affiliation, Frankfurt University of Applied Sciences, reflecting this relocation with the team's new name FRA-UNIted. Thus, the FRA-UNIted team is proud to look back on a 25-year history in robotic soccer simulation this year.

As a continuation of our efforts in the ancestor project, the underlying and encouraging research goal of FRA-UNIted is to exploit artificial intelligence and machine learning techniques wherever possible. Particularly, the successful employment of reinforcement learning (RL, [13]) methods for various elements of FRA-UNIted's decision making modules – and their integration into the competition team – has been and is our main focus. Moreover, the extended use of the FRA-UNIted framework in the context of university teaching has moved into our special focus.

In this team description paper, we refrain from presenting approaches and ideas we already explained in team description papers of the previous years [4]. Instead, we focus on recent changes and extensions to the team as well as on reporting partial results of work currently in progress. We start this team

description paper, however, with a short general overview of the FRA-UNIted framework. Note that, to this end, there is some overlap with our older team description papers including those written in the context of our ancestor project (Brainstormers 2D, 2005–2010) which is why the interested reader is also referred to those publications, e.g. to [6, 11].

1.1 Design Principles

FRA-UNIted relies on the following basic principles:

- There are two main modules: the world module and decision making
- Input to the decision module is the approximate, complete world state as provided by the soccer simulation environment.
- The soccer environment is modeled as a Markovian Decision Process (MDP).
- Decision making is organized in complex and less complex behaviors where the more complex ones can easily utilize the less complex ones.
- A large 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).

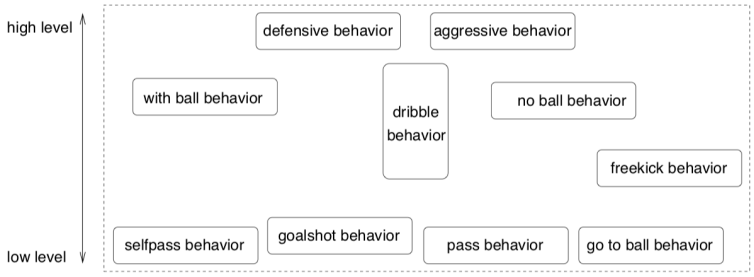

Fig. 1. The Behavior Architecture

1.2 The FRA-UNIted Agent

The decision-making process of the FRA-UNIted agent 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. 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 doing so, call the "defensive behavior" and delegating responsibility for action choice to it.

2 Case Studies on Data-Efficient Model-Based Batch-Mode Deep Reinforcement Learning

One of our recent streams of activity aims at a deeper analysis and understanding of fitted batch-mode reinforcement learning approaches where soccer simulation provides an excellent testbed for controlled experiments.

2.1 Efficient State Transition Modelling

In soccer simulation, the dynamics of the environment are given by the Soccer Server [9] software which models the effect of each agent's individual action as well as the impact of different sources of noise. Since the exact implementation of the dynamics is publicly available thanks to the open-source implementation of the Soccer Server, it is possible to employ it as a fundament for model-based RL where we frequently need to determine the successor state s' for a given pair of state s and action a executed by the agent. This is, e.g., necessary in the context of any kind of value iteration-like algorithm or, more generally, whenever some action needs to be selected greedily given some value function [13].

We developed a light-weight reimplementation of the Soccer Server's dynamics that supports the modeling of all field player action types (kick, dash, turn, tackle) which also correctly considers potential collisions of the agent with the ball and allows basic support of noise modeling. So far, our investigations focus on a single agent, i.e. the agent at hand intends to model the result of its own action. Nevertheless, it is also possible to apply this kind of state transition modeling for other agents (assuming that one agent would know the action executed by a teammate or opponent), but so far our implementation does not yet support the modeling of superpositions of interactions (e.g. when two players kick for the ball at the same time).

Since this implementation avoids any kind of (inter-process) communication with the Soccer Server and leverages the general speed advantage of C++, it is extremely fast and allows for pure agent-internal learning and analyses. For a balanced mix of randomly generated single-agent states and random actions that this agent executes, we found that we can simulate, model, and, hence, determine the successor state on average in $0.137\mu s$ which gives rise to 7.3 million state transition modelings per second on a single core of a contemporary 3-GHz CPU.

So far, we have employed the described state transition modeling primarily in a deterministic matter, i.e. disregarding the noise that is added by the Soccer Server. Bringing our modeling of noise in full alignment with the way the Soccer Server adds randomness to the simulation represents a direction for future work.

2.2 Batch-Mode RL for the Dribble Task

The task in model-free batch reinforcement learning (BRL, [8]) is to find a policy that maximizes the sum of expected rewards where, in contrast to the general, online learning case, the agent itself is not allowed to interact with the system

during learning. Instead of observing a state s, executing an action a and changing the policy according to the successor state s' and reward r, the agent receives a set $\mathbb{B} = \{(s_t, a_t, r_{t+1}, s_{t+1}) | t = 1, \ldots, b\}$ of transitions (s, a, r, s') sampled from the environment. If the model of the environment is known, things get even simpler: Then, only a batch of states $\mathbb{B} = \{s_t | t = 1, \ldots, b\}$ is required before the agent can start batch-processing the data in a model-based manner.

In a deterministic environment, the basic algorithm of fitted value iteration (FVI, historically also called smooth value iteration [1]) takes $\mathbb{B}$, a reward function $r : S \times A \times S \to \mathbb{R}$ and a transition function $f : S \times A \to S$ that provides the successor state given a current state and action, as well as a regression algorithm as input [2]. After having initialized the value function V and a counter i to zero, FVI repeatedly processes the following three steps until some stop criterion becomes true:

1. increment i
2. build up a training set $\mathbb{F}$ for the regression algorithm according to:

$$\mathbb{F} := \{(in^t, out^t) | t = 1, \ldots, b\}$$

   where $in^t = s^t$ and $out^t = r(s^t, a^*, f(s^t, a^*)) + \gamma V^{i-1}(f(s^t, a^*))$ for $a^* = \arg\max_{a \in A} V^{i-1}(f(s^t, a))$
3. use the regression algorithm and the training set $\mathbb{F}$ to induce a new approximation $V^i : S \to \mathbb{R}$ (for example, fit a neural network)

BRL approaches are frequently praised for their data efficiency. In the most general case, the agent does not make any assumptions on the sampling procedure of the states/transitions. They may be sampled by an arbitrary – even purely random – policy, they need not be sampled uniformly from the state or state-action space, nor along connected trajectories. However, using this limited information, the agent shall come up with a policy that is to be used for interacting with the environment. In practice, so-called "growing batch" approaches [8], which steadily increase the data set size by state samples from the current greedy policy and which somehow lie in the middle between pure batch and pure online learning approaches – are frequently used to improve learning as they tend to incentivize the agent to focus more on the relevant regions of the state space.

BRL for Dribbling We modelled the dribble task using a seven-dimensional state space (ball position and velocity relative to the agent (4d), player velocity (2d) and angular deviation from target direction (1d)) and a discretization of the action space using 576 (kick/dash/turn) actions in total. Immediate rewards are equated with the agent's velocity component into the target dribble direction. Since we have access to a model of state transitions (cf. Section 2.1), we employed a variant of fitted value iteration using neural network-based value function approximation (simple four-layered multi-layer perceptron neural nets). Interestingly, as far as the fitting part is considered, we trained the neural network using Rprop [10] for only five epochs in each FVI iteration, but we refrained from re-initializing the net in between FVI iterations such that the value function approximation is going to evolve slowly over the iterations. We employed

a discount factor $\gamma = 0.9$. We were interested in the question of data-efficiency and, to start with, selected varying amounts of randomly chosen states from the mentioned seven-dimensional state space. As mentioned above, we sticked to a noise-free environment, leaving learning under stochasticity for future work. We compared the learned policy to FRA-UNIted's established dribble behaviors (a hand-coded one as well as ND17 [3] which has been learned using online RL).

Evaluation Learning using a fixed batch of data, i.e. following a pure batch approach, and varying the data set size from 2k to 200k training examples, policies of varying quality can be obtained (cf. Figure 2). In particular, we have to acknowledge that (a) for small batch sizes no convergent learning behavior can be observed over the FVI iterations which means that the actual outcome of learning is subject to high variability, and (b) that the resulting policies do neither match the quality of our hand-coded dribble policy (which dribbles on average 81.67 metres per 100 time steps in a noise-free environment; $67.17m$ in a noisy one), nor the ND17 policy ($79.81m$ noise-free, $74.24m$ noisy). Using a growing batch approach that starts out with $b = 20k$ states and where we stopped increasing the amount of data after having collected 100k states, both reference dribble policies are surpassed in a noise-free soccer environment ($82.41m$).

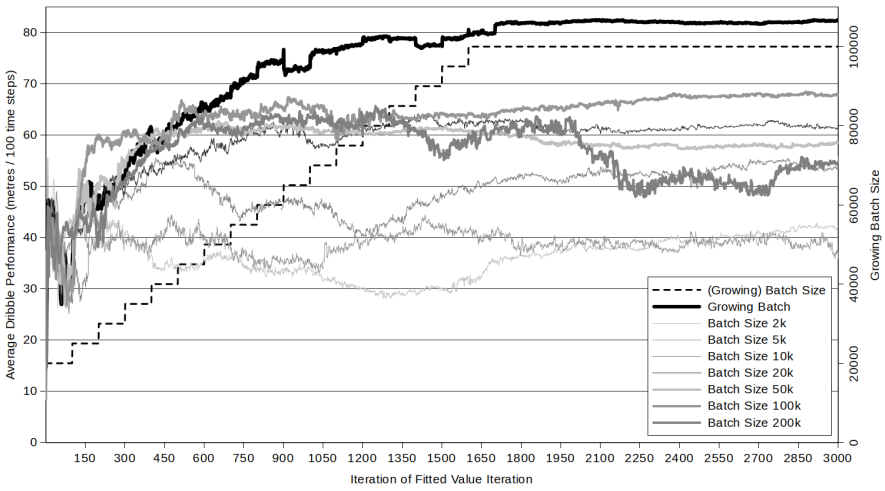

Fig. 2. Opposing Pure Batch and Growing Batch RL for the Dribble Task

Discussion It must be critically noted that quite some complexity of the explained approach is hidden in its "fitting part", i.e. in the way the target values provided by value iteration's Bellman update, are implanted into the neural network that is utilized as value function approximator. This process, its potential pitfalls as well as its subtle connection to the main loop of FVI will become the topic of a separate paper that is currently in preparation.

3 Continuous Integration System with Diversified Opponents and Dynamic Team Configurations

Since each match of 2D soccer is tainted to various degrees by randomness, the need to play a large number of matches arises, in order to limit the impact of chance on the overall results. For this reason, our team has developed and established the use of a continuous integration (CI) system which we already presented in an earlier team description paper [5]. Recently, our focus has been on a further development of this system with the goal of assessing our team's performance more reliably. In particular, we have targeted the following issues:

- analysis and reduction of the risk of overfitting, by having CI-based evaluations no longer by playing only against the same team
- no interaction and influence on the CI system besides new commits
- extensions towards using the potential for configuration testing

In [7], we present the details and recent enhancements of the CI system, allowing the management and usage of arbitrary team binaries, along with the ability to freely specify configuration values for arbitrary configuration files.

An outlier is defined as a match whose final result and/or course of the match strongly deviates from expected values. Evaluating outlier detection algorithms in the context of analyzing soccer simulation games is not trivial. In [7], we argue that some gold standard must be defined according to subjective beliefs that assess the data. On this basis, we defined patterns to identify outliers and trained two different machine learning models, viz local outlier factors and isolation forests, for classifying matches as outliers. Generally, these patterns are complicated and involve a lot of soccer background knowledge, which is why they may be improved by incorporating more features in order to score a higher true negative rate. Therefore, the results of the evaluation presented in [7] can be considered satisfactory, knowing that almost half of the outliers specified by the golden standard are successfully detected.

4 On the Impact of Soccer Server Changes: The Case of the Goalie

Due to two recent changes of the Soccer Server in its version 17 (SS_{V17}, for short) a lot of effort had to be invested in our goalkeeper in the last year. Firstly, in version 17.0 it is no longer possible to perform dashes with a power of -100. Secondly, the goalkeeper is now restricted to perform a catch command into a direction that must be within [-90,90] degrees relative to its body orientation, previously it had been possible for the keeper to also catch balls that were located behind it. Both of these capabilities were previously used intensively by our goalkeeper in order to move quickly and efficiently on the goal line and catch the ball without having to perform turns. As a result, numerous changes had to be introduced to the goalkeeper's movement strategy in order make its behavior competitive again. Using the described CI system (cf. Section 3) and

team Helios 2022, which is adapted to the mentioned changes of the simulation physics, as reference opponent, we could easily verify and quantify the severeness of the changes introduced by SS_{V17}. The FRA-UNIted version, which had been prepared for SS_{V16} and disregards the recent changes in the goalie action model, achieves an average score of 0.27 : 6.14 against Helios 2022. By contrast, the extended FRA-UNIted version of 2022, which has been altered to accomodate the necessary changes, achieves a score of 0.66 : 2.05 in the same setting. Thus, the goal received-to-scored quotient has been reduced from 22.85 to 3.11.

In addition, we found that, due to the changes, we only get a goal against in every 5th shot on our goal, instead of every 2nd shot before. As a side effect, while working on the goalkeeper, other weaknesses could be discovered. For example, some teams manage to play the ball behind our defensive lines and create a 1vs1 situation against our goalkeeper. At the moment, we are working on a solution to identify such situations more accurately and safely in order to enable our goalkeeper for taking better counter measures.

5 Code Maintenance and Redesign of Standard Situations

The FRA-UNIted team has been worked on by numerous students on a voluntary basis over the last decades. At places, the project was left disorganized in terms of structure and design. This issue is particularly noticeable in those parts that are responsible for handling standard situations. Therefore, the `StandardSituation` behavior serves as the ideal environment for the incorporation of adaptive solutions in a competitive setting or for the swift implementation of additional features. Our team's `StandardSituation` behavior has been published in [12]. While that version's implementation for standard situations (URL in footnote[1]) comprised 950 lines of C++ code, its contemporary 2022 counterpart has grown to 2900 lines. In particular, several of the conditional statements used currently contain rather intricate cases which are difficult to comprehend without a firm grasp of the underlying code. In an effort to stop the growth of the class and make it more comprehensive, we are going to restructure and redesign the standard situation behaviors and classes. This ongoing endeavor is intended to yield a source code that is easier to maintain due to being more readable and better structured. The new design of the Standard Situation behavior has thus two simple goals in mind:

- Improving readability: Write complex conditional statements in separate boolean functions with self-describing names
- Improving maintainability: Expand the base structure with additional classes in order to modularize Standard Situation by its core features

Due to the mentioned size of the current `StandardSituation` behavior it is nonsensical to recode the entire soccer-playing logic from scratch. Instead we intend to extract methods into separate classes and modularize complex conditional statements into separate boolean methods.

[1] `https://sourceforge.net/projects/bsrelease/`, see file bs2k/behaviors/standard_situation.c

6 Conclusion

In this team description paper we have outlined the characteristics of the FRA-UNIted team participating in RoboCup's 2D Soccer Simulation League. We have stressed that this team is a continuation of the former Brainstormers project, pursuing similar and extended goals in research and development as well as for teaching purposes. Specifically, we have put emphasis on our most recent research activities and practical implementation of our results.

References

1. Boyan, J., Moore, A.: Generalization in Reinforcement Learning: Safely Approximating the Value Function. In: Advances in Neural Information Processing Systems 7 (NIPS 1994). pp. 369–376. MIT Press, Denver, USA (1994)
2. Ernst, D., Geurts, P., Wehenkel, L.: Tree-Based Batch Mode Reinforcement Learning. Journal of Machine Learning Research 6(1), 503–556 (2006)
3. Gabel, T., Breuer, S., Roser, C., Berneburg, R., Godehardt, E.: FRA-UNIted - Team Description 2017 (2017), Supplementary material to RoboCup 2017: Robot Soccer World Cup XXI
4. Gabel, T., Breuer, S., Sommer, F., Godehardt, E.: FRA-UNIted - Team Description 2019 (2019), Supplementary material to RoboCup 2019: Robot Soccer World Cup XXIII, LNCS, Sydney, Australia
5. Gabel, T., Eren, Y., Sommer, F., Vieth, A., Godehardt, E.: FRA-UNIted - Team Description 2022 (2022), Supplementary material to RoboCup 2022: Robot Soccer World Cup XXVI, LNCS, Bangkok, Thailand
6. Gabel, T., Riedmiller, M.: Brainstormers 2D - Team Description 2009. In: L. Iocchi, H. Matsubara, A. Weitzenfeld, C. Zhou, editors, RoboCup 2009: Robot Soccer World Cup XII, LNCS (CD Supplement). Springer, Graz, Austria (2009)
7. Godehardt, E., Allani, M., Vieth, A., Gabel, T.: 1001 Games a Night – Continuous Evaluation of an Intelligent Multi-Agent Based System. In: Proceedings of the 8th International Congress on Information and Communication Technology (ICICT 2023). Springer, London, United Kingdom (2023)
8. Lange, S., Gabel, T., Riedmiller, M.: Reinforcement Learning: State of the Art. In: Wiering, M., Otterlo, M. (eds.) Batch Reinforcement Learning. Springer (2012)
9. Noda, I.: Soccer Server: A Simulator of RoboCup. In: Proceedings of the AI Symposium 1995. pp. 29–34. Japanese Society for Artificial Intelligence (1995)
10. Riedmiller, M., Braun, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: Proceedings of the IEEE International Conference on Neural Networks (ICNN). pp. 586–591. San Francisco, USA (1993)
11. Riedmiller, M., Gabel, T.: Brainstormers 2D - Team Description 2007. In: U. Visser, F. Ribeiro, T. Ohashi and F. Dellaert, editors, RoboCup 2007: Robot Soccer World Cup XI, LNCS (CD Supplement). Springer, Atlanta, USA (2007)
12. Riedmiller, M., Gabel, T., Schulz, H.: Brainstormers 2D: Public Source Code Release 2005. Technical Report, University of Osnabrück (2005), https://sourceforge.net/projects/bsrelease/files/bs05publicrelease.documentation.pdf
13. Sutton, R., Barto, A.: Reinforcement Learning. An Introduction. MIT Press/A Bradford Book, Cambridge, USA (1998)