In Daniele Nardi, Martin Riedmiller and Claude Sammut, editors, *RoboCup-2004: Robot Soccer World Cup VIII*, pp. 636--644, Springer Verlag, Berlin, 2005.

The UT Austin Villa 2003 Champion Simulator Coach: A Machine Learning Approach

Gregory Kuhlmann, Peter Stone and Justin Lallinger

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188
{kuhlmann,pstone,hosty}@cs.utexas.edu
http://www.cs.utexas.edu/~{kuhlmann,pstone,hosty}

Abstract. The UT Austin Villa 2003 simulated online soccer coach was a first time entry in the RoboCup Coach Competition. In developing the coach, the main research focus was placed on treating advice-giving as a machine learning problem. Competing against a field of mostly hand-coded coaches, the UT Austin Villa coach earned first place in the competition. In this paper, we present the multi-faceted learning strategy that our coach used and examine which aspects contributed most to the coach's success.

1 Introduction

The Coach Competition is a fairly recent addition to the RoboCup Simulated Soccer League. The competition aims to encourage research in multiagent modeling and advice-giving. The challenge is to create a coaching agent that significantly improves a team's performance by providing strategic advice.

In the RoboCup simulator [1], an online coach agent has three main advantages over a standard player. First, a coach is given a noise-free omniscient view of the field at all times. Second, the coach is not required to execute actions in every simulator cycle and can, therefore, allocate more resources to high-level considerations. Third, in competition, the coach has access to logfiles of past games played by the opponent, giving it access to important strategic insights.

On the other hand, the coaching problem is quite difficult due to two main constraints. First, to avoid reducing the domain to a centralized control task, a coach agent is limited in how often it can communicate with its team members. In addition, the coach must give advice to players that have been developed independently, often by other researchers. For this to be possible, coaches communicate with coachable players via a standardized coach language called CLANG [1].

Our UT Austin Villa coach was a first time entry in the coach competition. Similarly to some previous approaches to coaching ([5],[6],[7]), we treat advice-giving as a machine learning problem. Competing against a field of mostly hand-coded coaches, the UT Austin Villa coach earned first place in the competition. In this paper, we present the multi-faceted learning strategy that our coach used and examine which aspects contributed most to the coach's success.

2 Coach Framework

The basic operation of our UT Austin Villa[1] coach is as follows. Prior to a match, the coach examines the provided logfiles of games played by the *fixed opponent*. We call this team the fixed opponent, because in competitions it is determined by the league organizers and is the common opponent for all coach entries. The logfiles contain data from the fixed opponent's previous games. In particular, the coach does *not* see log files of the coachable team itself playing against the fixed opponent. All coaches advise the same coachable team consisting of players that understand and react to CLANG messages.

The coach collects data about players on the fixed opponent team as well as the players on the team the fixed opponent is playing against. For each player, the coach collects aggregate data such as the player's average location, as well as data about high-level events, such as passes and dribbles. After every change in possession, the coach's game analysis module attempts to classify the prior possession as a sequence of high-level events. The details of the identification procedure are described in our team description [2].

The data collected during logfile analysis are fed into a group of learning algorithms that generate player models for both teams. The models are then used to produce three different kinds of advice: formational, offensive, and defensive. The learned advice is combined with a few hand-coded rules and sent to the coachable team at the beginning of the match.

In past years' competitions, the team to be coached consisted of players developed at a single institution. Even in the absence of a coach, the players constituted a coherent team. In order to magnify the impact of a coach, in the 2003 competition, coachable teams were assembled from players developed at three different institutions: UT Austin Villa (our own), Wyverns from Carnegie Mellon, and WrightEagle from USTC in China. Furthermore, the coachable players were designed with only limited default strategy.

As a result, it was necessary to provide the players with advice about general game play. After brief experimentation with the coachable players, we identified the basic skills that they were missing and added hand-coded rules to help them overcome these weaknesses.

While the coach is best able to reason about players in terms of their roles, CLANG requires players to be specified by their uniform numbers. For this reason, the coach maintains a mapping between roles and uniform numbers for each player on both teams. Learned rules and hand-coded advice are created with role variables in the place of uniform numbers. When the rules are sent, the coach uses the current role map to insert the uniform numbers corresponding to each role variable. If during the course of the game players change roles, the affected rules are sent again with the updated player numbers. The details of how role mapping was used in the 2003 competition are described in our team description [2].

[1] `http:/www.cs.utexas.edu/~AustinVilla`

3 Learning

The core of the UT Austin Villa coach is its ability to learn player models from logfiles of past games. Similarly to Riley et al. [7], we break this problem down into learning three basic types of strategies: offensive (how the player should try to score), defensive (how they should act near their own goal), and formational (where the players should position themselves by default). This paper describes an independent formulation and implementation of these three basic strategies which differs in many of the particulars from previous work.

We assume that the set of available logfiles of the fixed opponent includes some games in which the opponent wins and some games in which it loses; in competition, we were given two of each. In the logfiles in which the fixed opponent performs well, we model the fixed opponent's offense and attempt to learn defensive advice to counter it. For the games in which the fixed opponent loses, we model the winning team and learn formational and offensive action selection advice.

For both offensive and defensive advice, the product of our learning algorithm is a classifier that is able predict the next high-level event to occur, given the current state of the game. To encode the simulator's state, we used a large set of features including the positions of all 22 players, the position of the ball, and the distances between them.

We used the J48 decision tree algorithm, implemented in the Weka machine learning software package [8], to train a series of decision trees, one for each modeled player. Because the structure of a decision tree is easily understandable, it is fairly straightforward to convert a tree into CLANG advice. The details of the example creation and advice generation procedures for the offensive and defensive advice are described in the following two sections. We then present the methods behind our formational advice learning.

3.1 Offensive Advice

When learning offensive advice, the coach attempts to model the behavior of the player with the ball. During the learning process, the coach builds a classifier for each player that tries to predict what that player will do with the ball in any given situation. For player i, we define the possible classes to be:

- $Pass(k)$: Pass to teammate with uniform number $k \in \{1..11\} - \{i\}$.
- *Shot*: Take a shot on goal.

During logfile analysis, when a shot or pass is identified, the state of the environment at the last kickable time is stored in the database along with the true class label and the player number: i. A classifier is then built for each player using only the examples corresponding to its own player number.

Once we have trained a decision tree for player i, we can convert it into advice to be given to our own corresponding player. To understand the advice generation process, consider the example decision tree for player 5 shown in Figure 1. Each leaf node of the decision tree is an action. The path from the root to the leaf defines a conjunction of conditions under which that action should be executed. Therefore we can construct a rule, {condition}→{action}, for each leaf node in the decision tree. For example, the rule for the leftmost leaf of the example decision tree is:

```
(BallX < 10) ∧ (BallY < 10)->Pass(6)
```

Or in CLANG:

```
(define
  (definerule OffRule1 direc
    ((and (bpos (rec (pt -52.5 -34) (pt 10 34)))
          (bpos (rec (pt -52.5 -34) (pt 52.5 10))))
     (do our {5} (pass {6})))
  )
)
```

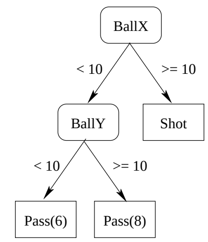

Fig. 1: Example decision tree learned for offensive advice.

3.2 Defensive Advice

To generate defensive advice, we model the behavior of the opponent and attempt to foil its predicted strategy. Here, we aim to predict how a given player will acquire the ball. The set of classes is $Pass(k)$ where k is the uniform number of the player *by* whom the pass was made. Because we are interested in predicting a pass before it is made, we don't just record the state at the last kick time as we did in the offensive case. Instead, we record the 10 cycles (1 second) prior to the last kickable time and label each instance with the true class label and the player number of the pass receiver. An example tree learned for player 5 is shown in Figure 2.

We use a heuristic model to convert the learned predictions regarding opponent behaviors to defensive actions that can *prevent* that action. To prevent a pass, it is a good idea to position a defender along a passing lane closer to the intended receiver than to the passer. We found that positioning the defender at about 70% of the pass length away from the ball was a reasonable choice. Assuming that our player 7 is guarding opponent 5, then the CLANG rule corresponding to the leftmost branch of the decision tree in Figure 2 is:

```
(define
  (definerule DefRule1 direc
    ((and (bpos (rec (pt -52.5 -34) (pt 0 34)))
          (bpos (rec (pt -52.5 -34) (pt 52.5 10))))
     (do our {7} (pos (((pt opp 6) * (pt .7 .7)) +
                       (pt opp 5) * (pt .3 .3)))))
  )
)
```

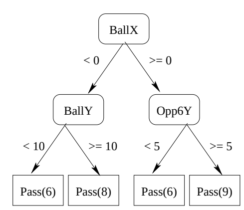

Fig. 2: Example decision tree learned for defensive advice.

3.3 Learning Formations

Our approach to learning a team formation is similar to our approach to learning offensive advice. The coach observes a team that can beat the opponent and then attempts to mimic that team's behavior. We model the formation as a home position (X, Y) and ball attraction vector (BX, BY) for each player. In CLANG, a formation is a positioning rule of the following form for each player, P:

```
(do our {$P} (pos ((pt $X $Y) + ((pt ball) * (pt $BX $BY)))))
```

The X and Y values are calculated as the average x and y coordinates of the observed player during the course of the game. Values for BX and BY were handpicked for each position and were found through brief experimentation. In some cases, we found that the ball attraction would cause the forwards to play too far towards the opponent goal, so to compensate, we manually moved the home positions back a bit.

4 Experimental Results

In this section we present the results of several experiments involving our learned coach, both in competition and in more controlled settings.

4.1 The Competition

The UT Austin Villa coach came in first place out of 12 entries in the 2003 RoboCup Coach competition. The competition consisted of three rounds. In each round, the coached team played three ten-minute games against a fixed opponent. Coaches were evaluated based on goal difference: the number of goals scored by the coachable team minus the number of goals scored by the opponent. The fixed opponents were all teams that competed in the main simulator competition: Boldhearts in round 1, Sirim in round 2, and EKA-PWr[2] in round 3.

The score differences and rankings for the top four finishing teams are shown in Table 1.[3] Our coach was ranked 7th after the first round. After making improvements to the hand-coded advice (but still retaining the learned offensive and formation advice as described above), we moved into first place after the second round. Four coaches (our own UT Austin Villa along with FC Portugal[4], Iraniansand Helli-Amistres[5]) progressed to the final round with UT Austin Villa coming out on top.

Because the number of games in the coach competition is too small to provide statistically significant results, we reran the final round for 50 games on our own. The advice sent by the other coaches was extracted from the logfiles of the competition and duplicated verbatim. We used the same team of coachable players as the one used in the competition so as to reproduce the exact conditions. The results of the comparision are summarized in Table 2.

In our tests, our coach had the highest average, but based on a two-tailed student's t-test the results are not statistically better than those of the second place team, FC Portugal. Even after 50 games, the results were not significant ($p > 0.9$). However, the scores for FC Portugal and our UT Austin Villa coach

[2] `http://autonom.ict.pwr.wroc.pl/RoboCup/english/english.html`
[3] Complete results are available from `www.uni-koblenz.de/~fruit/orga/rc03/`
[4] `http://www.ieeta.pt/robocup/`
[5] `http://www.allamehelli.net/pages/robo.html`

Coach	1st Round (Boldhearts)		2nd Round (Sirim)		3rd Round (EKA-PWr)	
UT Austin Villa	0:19	7th	0:2	1st	8:2	1st
FC Portugal	1:21	8th	0:8	4th	7:3	2nd
Iranians	0:14	4th	0:5	3rd	3:2	3rd
Helli-Amistres	1:12	2nd	0:3	2nd	7:7	4th

Table 1: Total scores and rankings for the top four finishing teams in the 2003 RoboCup coach competition. The score consists of the number of goals scored by the coached team followed by the number scored by the fixed opponent.

were significantly[6] better than the next best team: Helli-Amistres. Therefore, under controlled conditions our coach tied with FC Portugal for first place.

Coach	Score	StdDev	Rank
UT Austin Villa	2.38	2.61	1st
FC Portual	2.24	1.53	1st
Iranians	-0.4	1.74	4th
Helli-Amistres	0.85	1.81	3rd

Table 2: Summary of 50 runs of the final round of the 2003 RoboCup Coach Competition. Average goal differences are shown along with their standard deviations and overall ranking.

4.2 Additional Experiments

After the competition, we conducted additional controlled experiments to isolate the key components of our learned coach agent. For all tests, we used the same fixed opponents as in the competition (BoldHearts, Sirim, and EKA-PWr), and the same scoring metric. All reported scores have been averaged over 25 games.

Our first experiments were aimed at isolating the impact of each variety of advice given by our coach. We tested several different configurations of advice using a coachable team consisting of only our (UT Austin Villa) players.

The results of these first experiments are presented in Table 3. In the table, the column labled "w/ HC" indicates whether or not the hand-coded advice was included (**Y**es/**N**o). The column labeled "Formation" contains the results for learned formation advice only. The "Offensive" and "Defensive" columns show the results of adding offensive and defensive advice, respectively. "Full" includes all three types of learned advice. These advice configurations were compared with the default behavior of the coachable players without any advice, labeled "None".

From the table, it is clear that both with and without hand-coded rules, across all opponents, the learned advice did significantly better than no advice at all ($p < 0.05$).

Another observation that is true across the board is that the learned formation advice had the most significant impact of all advice types. On the other

[6] For all of the results presented in this paper, significance was determined by using a two-tailed student's t-test with $p < 0.05$.

Opponent	w/ HC	None	Formation	Offensive	Defensive	Full
BoldHearts	N	-8.8	-3.3	-2.9	-2.9	-2.7
	Y	-6.8	-0.5	-1.4	-5.7	-6.5
Sirim	N	-4.1	2.6	1.2	0.9	1.7
	Y	-5.4	-1.6	-0.3	0.8	-0.4
EKA-PWr	N	-0.6	2.8	2.9	3.4	2.7
	Y	1.0	3.62	2	2.12	2.43

Table 3: Average goal differences with varying levels of advice.

hand, with the exception of EKA-PWr, it appears that the offensive and defensive advice conflicted with the hand-coded advice. During the competition, we noticed that this was occuring after the first round. As a result, we decided to turn off the learned defensive advice for the remaining rounds. In retrospect, this was a very prudent decision.

Except in the case of Boldhearts (with defensive advice removed), we would have probably achieved higher scores in the competition, had we not added the hand-coded rules. This is a surprise considering the preliminary tests we performed with the coachable players, which suggested that hand-coded advice was necessary.

5 Related Work

Some previous work has been done on learning to give advice to RoboCup simulated soccer players. Similarly to our own work, Riley et al. [7] approached advice-giving as an action-prediction problem. Both offensive and defensive models were generated using the C4.5 [3] decision tree learning algorithm. Their work also stressed the importance of learned formation advice. While our decomposition of the problem is similar to theirs, our model representations and advice-generation procedures are quite different. For example, whereas our approach learns the player's average position and then considers the positioning of the players with respect to the ball when giving advice, theirs ignores the ball and instead focuses on correlations between players. In addition, the semantics of our learned defensive rules, which aim to learn not what the player with the ball will do, but how it will get the ball in the first place differs from what was done previously.

In other work, Riley and Veloso [6] used Bayesian modeling to predict opponent movement during set plays. The model was used to generate adaptive plans to counter the opponent's plays. In addition, Riley and Veloso [5] have tried to model high-level adversarial behavior by classifying opponent actions as belonging to one of a set of predefined behavioral classes. Their system was able to classify fixed duration *windows* of behavior using a set of sequence-invariant action features.

ISAAC [4] is a game analysis system created as tool for simulated soccer team designers. Similar to a coach, this system analyzes logfiles of a game in order to suggest advice for how a team's play can be improved. However, this advice is meant to be understood by the team's developers instead of the agents themselves.

6 Conclusion and Future Work

We have presented our multi-facted learning approach to giving advice in RoboCup simulated soccer. Using this approach, our UT Austin Villa coach won first place in the 2003 RoboCup Coach Competition. Through controlled experiments, we found that our coach was significantly better than the third and fourth place finishing teams and at least as good as the second place finisher. In addition, we have identified the learned formation rules as the most effective type of advice.

In our research we are continuing to enhance and carefully test the learned defensive and offensive advice, and we plan to test the degree to which each type of learned advice is opponent-specific. Meanwhile, we plan to continue working on finding ways to learn better, more adaptive formations. In addition, we intend to explore various methods for generating set play advice. Finally, we will be adapting our learning strategy to include online learning.

Acknowledgments

We thank Patrick Riley for helpful comments and suggestions. This research is supported in part by NSF CAREER award IIS-0237699.

References

1. Mao Chen, Ehsan Foroughi, Fredrik Heintz, Spiros Kapetanakis, Kostas Kostiadis, Johan Kummeneje, Itsuki Noda, Oliver Obst, Patrick Riley, Timo Steffens, Yi Wang, and Xiang Yin. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later, 2003. Available at `http://sourceforge.net/projects/sserver/`.
2. Gregory Kuhlmann, Peter Stone, and Justin Lallinger. The champion UT Austin Villa 2003 simulator online coach team. In Daniel Polani, Brett Browning, Andrea Bonarini, and Kazuo Yoshida, editors, *RoboCup-2003: Robot Soccer World Cup VII*. Springer Verlag, Berlin, 2004.
3. J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
4. Taylor Raines, Milind Tambe, and Stacy Marsella. Automated assistants to aid humans in understanding team behaviors. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 85–102. Springer Verlag, Berlin, 2000.
5. Patrick Riley and Manuela Veloso. On behavior classification in adversarial environments. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, 2000.
6. Patrick Riley and Manuela Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002.
7. Patrick Riley, Manuela Veloso, and Gal Kaminka. An empirical study of coaching. In H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, editors, *Distributed Autonomous Robotic Systems 5*, pages 215–224. Springer-Verlag, 2002.
8. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999. `http://www.cs.waikato.ac.nz/ml/weka/`.