In Daniel Polani, Brett Browning, Andrea Bonarini and Kazuo Yoshida, editors, *RoboCup-2003: Robot Soccer World Cup VII*, Springer Verlag, Berlin, 2004.

The Champion UT Austin Villa 2003 Simulator Online Coach Team

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 learning problem. The coach learns to predict agent behavior from past observations and automatically generates advice to improve its team's performance. Using this approach, the UT Austin Villa coach earned first place in this year's competition.

1 Introduction

In RoboCup simulated soccer, an online coach is able to omnisciently observe the simulation environment and occasionally give advice to one of the teams in the hope of improving that team's performance [1]. Also, before facing an opponent, the coach is able to "scout" them by viewing log files for a few of their previous matches. This information can be used to construct a model of team behavior which can then be converted into advice.

The basic operation of our UT Austin Villa coach [1] is as follows. The coach examines log files of previous games against the opponent to identify significant events such as passes, shots, etc. The events and features describing the state of the environment when they occurred are output to a database. This database is then used to train a group of classifiers. The classifiers are then converted into the standard coaching language, CLang [1]. At the beginning of the match, this learned advice is combined with a few handcoded rules and sent to the players.

In this paper, Section 2 describes how the coach identifies events from position data; Section 3 explains our learning algorithm; Section 4 describes additional handcoded advice; Section 5 describes how we handle role switching; Section 6 summarizes the competition results; finally, Section 7 concludes.

2 Game Analysis

When the coach is run in online mode, it receives state information from the soccer server each cycle in the form of a *global see* message, which contains

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

the global positions and velocities of the ball and every player on the field. Additionally, the server sends another type of message to notify the coach of play mode changes. In offline mode, the coach extracts these messages from a log file instead of receiving them from the server.

These messages are used to update a data structure that maintains a history of game state for the 300 most recent time cycles. This history structure, implemented as a circular queue, contains instantaneous data such as positions and velocities, as well as cumulative data such as average positions and total travel distances.

The stored data are used mainly to identify the high-level events in the game play. The identification procedure is described in the next section.

2.1 Play by Play

From the coach's perspective, a game proceeds as a sequence of possessions. A possession change occurs whenever a new player gets the ball, there is a goal, or the play mode changes to one of several dead ball modes. When a possession change occurs, the coach analyzes the previous possession and attempts to characterize it.

Each possession consists of two parts. The first part, the *hold interval*, starts when the ball owner gains possession and persists until the last time the ball is within kickable range. The final cycle in this interval is called the *last kickable* time. The second part, the *kick interval* begins in the cycle following the last kickable time and ends at the next possession change.

If the ball moves a significant distance during the hold interval then the ball owner's action sequence is classified as a **dribble**. If the ball remains stationary, but the interval lasts several cycles, then the sequence is declared a **hold**. Otherwise, the player is said to have not performed any action at all.

If at the end of the kick interval, the ball is in the goal, then the ball owner obviously shot and scored a **goal**. If a teammate, gains possession, then the ball owner is said to have made a successful **pass**. Although, the next possessor may not have been the ball owner's intended receiver, we found that it was a safe assumption to make. If the play mode at the end of the interval is a dead ball mode, then the ball owner is said to have caused a **foul**. The coach does not distinguish between different types of fouls.

The most difficult case to interpret is a turnover. The coach considers four possibilities. If the ball is still a short distance away from the ball owner at the time of the turnover, then the the sequence is classified as a **steal** by the opponent. If the ball was headed for the goal and originated from a position reasonably close to the goal, then the kick is declared a **missed shot**. If the ball was headed for a teammate within a reasonable distance of the ball owner, the kick is assumed to be an **intercepted pass**. If the kick cannot be classified as any the above categories, then it is called a **clear**.

3 Learning

As stated previously, the coach learns to model team behavior from log files of previous games. We assume that this set of log files includes some games in which the opponent wins and some games in which the opponent loses; in competition, we were given two of each. In the log files where the fixed opponent performs well, their offensive strategy is able to take advantage of a weakness in the other team's defense. We used these games to model the fixed opponent's offense and learn defensive advice to counter it. The games in which the fixed opponent loses provide examples of offensive strategies that are able to overcome the fixed opponent's defense. We used these log files to model the winning team and learn offensive advice. Notice that in both cases we have chosen to model only offensive strategy. In our future work we plan to incorporate defensive modeling as well.

The goal of our learning algorithm is to create a classifier that can predict the next significant event in the game given the current state of the simulator environment. To encode the state of the environment, we used a large set of features:

- X-coordinate of the ball
- Y-coordinate of the ball
- X-coordinate of each player (22)
- Y-coordinate of each player (22)
- Distance from the ball to our goal
- Distance from the ball to opponent goal
- Distance from each player to our goal (22)
- Distance from each player to opponent goal (22)
- Distance from each player to the ball (22)
- Our score
- Opponent score
- Goal Difference
- Cycle number

The output of the game analysis module is a set of instances of these variables labeled with the actual action taken by the modeled team. These examples are used to train a series of decision trees, one for each modeled player. We used the J48 decision tree algorithm implemented in the weka machine learning software package [2]. 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 sections.

3.1 Offensive Advice

When learning offensive advice, the coach attempts to model the behavior of the player with the ball. Although we are able to identify several behaviors (see Section 2.1), currently we focus only on classifying passing and shooting. 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 log file analysis, when a shot or pass is identified, the state of the environment at the last kick 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. Consider the example decision tree for player 5 shown in Figure 1.

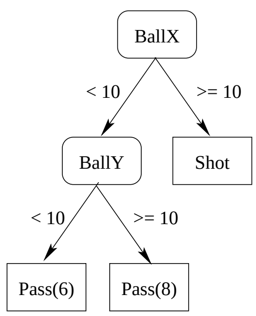

Fig. 1. Example decision tree learned for offensive advice.

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})))
  )
)
```

3.2 Defensive Advice

To generate defensive advice, we model the behavior of the opponent and attempt to foil its predicted strategy. Although we initially approached defensive advice similarly to offensive advice in that we planned to model the behavior of the player with the ball, we found more success when we changed to modeling how a player acquires the ball instead.

For defensive advice, we predict only passes. The set of classes is simply $Pass(k)$ where k is the number of the playerby whom the pass was made. Because we are interested in predicting a pass before it is made, we don't simply record the state at the last kick time as we did in the offensive case. Instead, we record the 10 cycles prior to the last kick time and label each with the true class label and the player number of the pass receiver.

While we would expect that most instances will be labeled as a pass from the player with the ball at the time (or the player most likely to get the ball), if the ball is acquired from a sequence of kicks lasting less than 10 cycles, the state will be labeled as a pass from the intermediate passer. The hope is that the classifier will be able to use this information to identify pass chains.

An example tree learned for player 5 is shown in Figure 2.

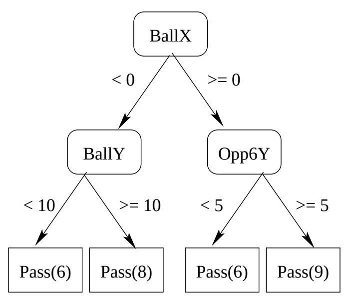

Fig. 2. Example decision tree learned for defensive advice.

In the case of offensive learning, the players could simply be advised to execute the actions that were observed to be successful in the past. However, for defensive learning, the generation of advice is not as straightforward. Here, we use a heuristic model to convert the learned predictions regarding opponent behaviors to defensive actions that can *prevent* that action.

For example, 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)))))
  )
)
```

3.3 Learning Formations

Our approach to learning a team formation was 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 terms of 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 simply 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.

In addition to the in-play formation, the set-play formation used for kickoffs contained the same home positions, but did not include any ball attraction.

This technique has a great deal of room for improvement. In the future, we plan to limit the space of possible formations to only those that are symmetric so as to make more efficient use of the training data. This optimization will be beneficial under the (reasonable) assumption that the opponents use a symmetric formation. Additionally, we plan to extend the learning procedure to learn the ball attraction values.

4 Handcoded Advice

In order to magnify the impact of the coach on team performance in this year's competition, the coachable players were given no default strategy. As a result, it was necessary to provide the players with advice about general soccer strategy. After brief experimentation with the coachable players, we identified the basic skills that they were missing and added rules to help them overcome these weaknesses. The following is a list of the handcoded rule names and descriptions:

- `utfwd`: This rule advises our forwards to shoot more often. We noticed that the coachable team wasn't taking enough shots. By adding this rule, we hoped to increase our chances of scoring.

– `utmid`: This rule tells our midfielders to pass forward or shoot. We found that our midfielders were passing backwards far too often. We added this rule to improve our team's ability to move the ball in the right direction.
– `utdefclr`: This rule tells each of our defenders to clear the ball to the nearest sideline, thus decreasing the likelihood of dangerous passes across the front of our goal. This rule had to be implemented in two different ways because of different *clear* implementations (see Section 5.2).
– `utgoalie`: This rule serves the same basic purpose as the previous rule. It simply tells the goalie to clear towards the nearest sideline.

5 Role Mapping

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 role-to-unum mapping for each player on both teams. Learned rules and handcoded 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 this mapping appears to have change, the affected rules are resent with the updated player numbers.

5.1 Opponents

In most cases, teams use a fixed formation that is consistent across games, which makes opponent role mapping trivial. However, we found that some teams switch formations frequently during the course of a game. For instance, YowAI [2] switches players' roles within its formation after each goal is scored. The look of the formation is the same (e.g. 4-3-3), but players at each of the positions change.

To handle such cases, we create a role mapping based on the position of each player before kickoff. Each player is given an initial role and the player's starting location is recorded. Before the next kickoff, each role is reassigned to the player whose position most closely matches the role's position during the previous kickoff.

5.2 Our team

One of the challenges of the coach competition is that the coach must coordinate a team made up of coachable players developed by different research groups. These players not only have different implementations of basic skills, but they also have different ways of interpreting advice. Therefore, we would like the coach to give a player advice that is tailored to its implementation. Also, we would like the coach to assign roles to players based on their perceived strength.

[2] `http://ne.cs.uec.ac.jp/~koji/`

During the American Open, a regional event that occurred about 2 months before RoboCup,[3] the identities of the players were not known to the coach ahead of time. However, the default formations of coachable players varied widely, so their positions at startup were sufficient for identification. The problem was simplified at the RoboCup competition by making the identities known a priori. Thus, we were able to use a fixed role mapping.

Shortly before the competition, the teams are given access to all of the coachable player binaries. By performing a few tests, we can guage the relative strengths and weaknesses of the various players and discover any nuances in their CLang interpretations.

In this year's coach competition, three coachable teams were submitted: UT Austin Villa (our own), Wyverns (from Carnegie Mellon), and WrightEagle (from USTC in China). Although we initially found that the UT Austin Villa players were best on defense and the WrightEagle players were best on offense, with some changes to the handcoded advice, we found that the best distribution of players was to have players of each type in the defender, midfielder, and forward positions.

Additionally, we discovered that the UT Austin Villa players interpreted the `clear` directive differently than the other players. Our players cleared the ball *to* a region, while the other players cleared the ball *from* a region. We found that for the other players, a better way to implement the clear action was to advise them to make a distant pass. The player-role mapping allowed our coach to send this kind of customized advice.

6 Competition Results

The UT Austin Villa coach came in first place out of 11 entries in this year's RoboCup Coach competition. The competition consisted of two elimination rounds and a final round, each with a different fixed opponent. In each round, the coached team played 3 games against the fixed opponent. Coaches were ranked by the total score difference across the games.

The score differences and rankings for the top four finishing teams are shown in Table 1.[4] Our coach was ranked 7th after the first round, which was barely good enough to move on. During this round, we notice that the defensive advice seemed to cause problems due to differing interpretation of the "pos" directive amongst players. Thus, we limited the automatic defensive advice for the second and third rounds, in particular by specifying some of the defensive positioning by hand. After making such improvements to the handcoded advice (but still retaining the learned offensive and formation advice as described above), we moved into first place after the second round, with the best goal difference out of the 8 remaining coaches. Four coaches progressed to the final round with UT Austin Villa coming out on top.

---

[3] `http://www-2.cs.cmu.edu/~AmericanOpen03/`

[4] Complete results are available from `http://www.uni-koblenz.de/~fruit/orga/rc03/`

Team	1st Round		2nd Round		3rd Round	
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 first number in the score is the number of goals scored by the coached team in the three games that made up that round. The second number is the total score of the fixed opponent.

7 Conclusion and Future Work

The main focus of the UT Austin Villa simulated coach team is on developing agent modeling techniques suited to the problem of advice-giving. As described above, our current coach uses only scouted data to construct its opponent model. Updating this model during game play is planned for future work. We also plan to try to improve the accuracy of predictive power of our learning algorithm while finding new ways to generate good advice from this knowledge. In addition, we will be exploring the new advice-giving possibilities of proposed CLang extensions, such as the ability to express directives with sequences of actions.

Acknowledgments

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. 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/`.