In *Proceedings of the Twenty-First National Conference on Artificial Intelligence* (AAAI-06), Boston, MA, July 2006.

Know Thine Enemy: A Champion RoboCup Coach Agent

Gregory Kuhlmann, William B. Knox, and Peter Stone
Department of Computer Sciences, The University of Texas at Austin
1 University Station C0500, Austin, Texas 78712-1188
{kuhlmann,bradknox,pstone}@cs.utexas.edu

Abstract

In a team-based multiagent system, the ability to construct a model of an opponent team's joint behavior can be useful for determining an agent's expected distribution over future world states, and thus can inform its planning of future actions. This paper presents an approach to team opponent modeling in the context of the RoboCup simulation coach competition. Specifically, it introduces an autonomous coach agent capable of analyzing past games of the current opponent, advising its own team how to play against this opponent, and identifying patterns or weaknesses on the part of the opponent. Our approach is fully implemented and tested within the RoboCup soccer server, and was the champion of the RoboCup 2005 simulation coach competition.

Introduction

In an integrated multiagent system, the ability to predict the behavior of the other agents in the environment can be crucial to one's own performance. Specifically, knowing the likely actions of other agents can influence an agent's expected distribution over future world states, and thus inform its planning of future actions.

In an adversarial environment, this predicted behavior of other agents is referred to as an *opponent model*. Opponent models are particularly useful if they include some identification of potential patterns or weaknesses on the part of the opponent. For example, a chess grandmaster may study past games of a future opponent so as to determine how best to play away from that opponent's strengths.

In multiagent adversarial settings, in which the adversary consists of a *team* of opponents, it can be useful to explicitly model the opponent as engaging in team activities. For example, Tambe (1996) presents a simulated air-combat scenario in which an individual's behavior can indicate the commencement of a team "pincer" maneuver that requires multiple participants, thus enabling the prediction of other opponents' future actions as well.

This paper presents an approach to team opponent modeling in the context of the RoboCup simulation coach competition. RoboCup is an international research initiative that uses the game of soccer as a testbed to advance the state of the art in AI and robotics (Kitano *et al.* 1997). In most RoboCup soccer leagues the goal is to create complete teams of agents that can succeed at the task of winning soccer games. Though opponent modeling can play a part in this task, it is often the case that opponents cannot be observed prior to playing against them (at least not by the agents themselves). Even when they can be observed, opponent modeling challenges are easily overshadowed by challenges such as vision, localization, locomotion, individual ball manipulation, teamwork, etc.

In contrast, the goal of the simulation coach competition is to focus entirely on opponent modeling. This focus is accomplished by i) providing entrants with recordings of the opponents' past play that is understandable by the coach agent; ii) providing each entrant with an identical team of fully competent player agents; and iii) restricting the actions available to *advice* regarding how the team should alter its playing style to fit a particular opponent.

This paper introduces UT Austin Villa, the champion of the RoboCup 2005 coach competition. We begin with a description of the details of the RoboCup coach competition. We then present our approach to modeling team behavior within this context, followed by UT Austin Villa's strategy for identifying opponent patterns or weaknesses. We then summarize and analyze the results of the RoboCup 2005 competition and present the results of subsequent experiments that further emphasize the agent's abilities.

The RoboCup Coach Competition

RoboCup simulated soccer has been used as the basis for successful international competitions and research challenges. As presented in detail by Stone (Stone 2000), the RoboCup soccer server (Noda *et al.* 1998; Chen *et al.* 2003) provides a fully *distributed*, *multiagent* domain with both *teammates* and *adversaries*. There is *hidden state*, meaning that each agent has only a partial world view at any given moment. The agents also have *noisy sensors and actuators*, meaning that they do not perceive the world exactly as it is, nor can they affect the world exactly as intended. In addition, the perception and action cycles are *asynchronous*, prohibiting the traditional AI paradigm of using perceptual input to trigger actions. *Communication* opportunities are limited, and the agents must make their decisions in *real-time*. These italicized domain characteristics combine to make simulated robotic soccer a realistic and challenging domain.

Since 1997, there have been annual international competitions in the soccer server in which each entrant creates a complete team of 11 autonomous agents to play against the other teams. The RoboCup simulation coach competition, introduced in 2001, is situated within the same soccer server. However, rather than creating full teams of agents, entrants create a single *coach* agent that has an omniscient view of the field, but whose only possible action is verbal advice to its team communicated via a formal language.

In the soccer server, 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. During game play, the coach may only send advice every 30 seconds and the advice does not reach the players until 5 seconds after it is sent. 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 (Chen *et al.* 2003).

In its original conception, the coach competition provided 24 hours from the time the opponent logfiles (recordings of their past games) were released until the competition was run. As a result, coach developers were able to manually modify their coaches to help the coachable players defeat the opponent (though without any opportunities to actually practice against the opponent). Though the coach competition was developed for the purpose of encouraging automated opponent modeling, with just two exceptions that we are aware of (Riley, Veloso, & Kaminka 2002; Kuhlmann, Stone, & Lallinger 2005), most entrants created their coaches entirely by hand.

For the first four years of the competition, beginning in 2001, coaches were evaluated on the performance of the coached team against a fixed opponent. While the intention was for the coach to model the opponent to find a team-specific strategy, many coaches performed well by using highly tuned general purpose strategies.

In order to place all the emphasis on opponent modeling, in 2005, the competition underwent drastic rule changes. In the 2005 competition, coach agents were directly evaluated based on their ability to model a teams with specific weaknesses. Such weaknesses include pulling the goalie out too far, leaving part of the field unguarded, kicking the ball out of bounds frequently, etc. After modeling the team, the coach is rated on how well it recognizes these weaknesses when they show up in a different context. To be successful, a coach must isolate these weaknesses, called *patterns*, from the rest of the team's behavior, called its *base strategy.* The 2005 Coach Competition Rules define these terms as:

pattern : "a simple behavior that a team performs which is predictable and exploitable"

base strategy : "the general strategy of the test team", independent of any present pattern

Each round of the competition consists of two phases. In the *offline* phase, the coach reviews game log files to construct its pattern models. A log file for a complete 10-minute game contains the positions and velocities of the ball and every player on the field for each of the game's 6000 cycles. High-level events such as passes, dribbles and shots on goal do not explicitly appear in the log file, and therefore, can only be inferred from the positional data.

For each of the 15-20 patterns it must model, the coach is given two log files: one from a team that displays the pattern, and another with the same base strategy, but without the pattern present. Each file is labeled with the pattern's name. The coach is given 5 minutes per log file to construct its models and save their representations to be used in the next phase.

In the *online* phase, the coach observes a live game of a team that exhibits an unknown combination of five patterns from those encountered during the offline phase. For example, in one of the iterations of the second round of the competition, the opponent team exhibited the following patterns:

- Goal keeper moves up and down penalty area
- Sweeper is poorly positioned
- Defenders always clear the ball
- Wing midfielders play too aggressively
- Forwards immediately kick towards goal

When the coach recognizes one of these *active* patterns, it announces it by sending the pattern's name to the simulator. At the end of the game, the coach's responses, along with their announcement times, are entered into a scoring function to rate the coach's performance. In this metric, more points are awarded for announcements made earlier in the game. A correct pattern declared at time, t_d, earns the coach the following number of points:

$$(9000 - t_d) \times \frac{N_T - N_A}{N_T} \tag{1}$$

where N_T is the total number of patterns (15–20), and N_A is the number of currently activated patterns (5). An incorrect pattern declaration, at time t_d, earns the coach the following penalty:

$$9000 - t_d \tag{2}$$

This scoring function has the property that the expected value of random guessing is 0.

During the online phase, the coach is able to control a team of coachable players to play against the team to be modeled, by sending advice in a language called CLANG. The behavior of the coachable team can be "programmed" by issuing CLANG instructions that the players are designed to understand. While the coach is not evaluated on its team's playing ability, the coach may use the team in an attempt to elicit the patterns in the opponent team.

For our coach, we chose a CLANG strategy that we thought would be most "normal", in that the team's behavior would be close to the types of strategies exhibited by the unmodeled teams in the pattern log files. Our coach's CLANG rules specify player formations for before kickoff and during game play. Also, it advises the defenders to pass to the midfielders and the midfielders to pass to the forwards. When in the opponent's half of the field, our forwards are told to dribble towards the near end of the opponent's goal. If any player possesses the ball within shooting range of the goal, the player is advised to take a shot.

Modeling Team Behavior

During the course of the competition, the coach must model three different types of teams. In the offline phase, for each pattern, the coach must model the team with the pattern activated, which we call the *pattern team*, and the team with the pattern deactivated, which we call the *base team*. Lastly, the team modeled during the online phase is called the *online team*.

We model all three of these team types by characterizing their behavior with a set of features calculated from statistics gathered while observing a game. These features can be characterized as i) the team *formation* indicating the general positioning of the agents (e.g. how many players are defenders or attackers) and ii) play-by-play statistics indicating the frequency of game events such as passes, shots, dribbles, etc.

Formation Modeling

As has been noticed by past coach league participants (Riley, Veloso, & Kaminka 2002; Kuhlmann, Stone, & Lallinger 2005), one of the most distinguishing features of a team is its formation. For this reason, we focused much of our effort on building accurate formation models.

Our particular formation model was chosen to take advantage of background knowledge about the teams to be modeled. In the competition, the behaviors of the base team, the pattern team, and the online team are all specified in CLANG. In CLANG, a player's position is described by a rule such as the following:

```
(say (define (definerule Rule_Home_2 direc
       ((playm play_onb) (do our {2} (pos ((
  (pt ball) * (pt 0.1 0.45)) + (pt -10 10))))))))
```

This rule instructs player 2 to move to a position calculated from a fixed home position, $(-10, 10)$, and a ball attraction vector, $(0.1, 0.45)$. The player tends to stay around its home position, but be *attracted* towards the ball in proportion to the magnitude of the x and y components of the ball attraction vector. If these components are close to 0, the player always stays at its home position. If they are closer to 1, the player tries to move up and down the field along with the ball.

To learn a team formation model, we estimate the home positions and ball attraction vectors for each of the players. The relationship between the ball's position and the player's position is linear, so we use linear regression to estimate values. The x and y components of the position are independent of each other, so they are learned independently.

In each game cycle, the players' and ball's positions are recorded. For each player, we learn its x model and its y model. To learn player p's x model, we use linear regression to find the parameters a_x and h_x that minimize the mean-squared error of the following equation:

$$p_x = a_x \times b_x + h_x$$

on the recorded data:

$$< b_x^{(1)}, p_x^{(1)} >, < b_x^{(2)}, p_x^{(2)} >, \cdots, < b_x^{(n)}, p_x^{(n)} >$$

where $b_x^{(i)}$ is the x-coordinate of the ball and $p_x^{(i)}$ is the x-coordinate of player p, both at time i.

The player's y model is learned in the same way. The total of four learned parameters constitute the player's formation model. The player's estimated home position is (h_x, h_y) and its estimated ball attraction vector is (a_x, a_y). We can evaluate the accuracy of the model on new data by measuring the Euclidean distance between $(p_x^{(i)}, p_y^{(i)})$ and $(a_x \times b_x^{(i)} + h_x, a_y \times b_y^{(i)} + h_y)$. In other words, the error e_{dm} of a ball attraction model, bam_m, with learned parameters h_x, h_y, a_x and a_y on a set of test data d of size n is:

$$\sum_{i=1}^{n} (a_x b_x^{(i)} + h_x - p_x^{(i)})^2 + (a_y b_y^{(i)} + h_y - p_y^{(i)})^2 \qquad (3)$$

While the ball attraction model works well most of the time, it does not accurately model the behavior of the player who is *fastest to the ball.* Given the velocities of the players and the ball, the player that is fastest to the ball is the one that could intercept the ball sooner than any of its teammates. Because this player typically leaves its formation to go after the ball, using its data during this period would add noise to the model. For this reason, in both training and testing, data for the player who at that time is fastest to the ball is ignored.

Although this method is motivated by the assumption that CLANG is used to describe team behavior, many past teams have used ball attraction vectors to determine player positioning (Veloso, Stone, & Bowling 1999; Kok *et al.* 2003). Thus the approach is likely to apply more generally to other teams.

Play-by-Play Statistics

In addition to the formation model, we gather additional statistics to characterize team behavior. Our coach identifies high-level events from the cycle-by-cycle positions and velocities of both the players and the ball. These high-level events include passes, shots on goal, directed kicks, and directed dribbling. Although many such features are collected, the 11 player ball attraction models are the only features that are used in UT Austin Villa's model, in part because they work well and in part due to time constraints leading up to the competition. An important direction for future work is to identify which of the remaining collected features could benefit our model.

After the offline logfile analysis completes, all of the formation model parameters for each pair of pattern / base log files are stored to disk in a *model file*. For both the *base* and *pattern* log files of each of the 15 to 20 patterns, the coach stores 44 formation parameters, 4 for each of the 11 players. After a post-processing step designed to normalize the comparisons of online teams against the various patterns (see the section on pattern normalization below), the model is ready to be used in the online phase.

Online Pattern Matching

The goal of the offline phase is to build up models of all the possible opponent patterns that can be observed during an online game. Whereas the offline training data can be extensive and the offline computation time is relatively unconstrained, the online phase requires identification of the opponents' weaknesses (patterns) in real time. The underlying premise is that if the the coach can quickly identify the opponent team's decision-making style, then it will be able to advise its team regarding what strategy to adopt in response.

During each game of the online phase, the coach identifies and records positional data and high-level events exactly as it did in the offline phase to construct an online model. As in the offline phase, we currently only use the 11 players' ball attraction models for pattern matching. Each of these models is a *feature* to which the coach assigns a score from 0 to 1.

At predefined times during the game, the coach calculates certainty scores for each of the 15 to 20 patterns in its model file based on how well the current opponent's behavior matches the pattern log versus the base log. For each offline pattern, the coach assigns a score for each player's current ball attraction model, such that if the online formation model is close to that of the pattern, then the score is close to 1. If the online formation model is closer to that of base strategy, then the score is closer to 0. The overall (unweighted) score for a pattern is calculated as the sum of the feature scores over all 11 players for that pattern.

We calculate a single player's formation model score for a pattern by determining whether its position data better matches the model of the pattern team or that of the associated base team as follows. Given the ball attraction model, bam_b, for the player constructed from the data in the base logfile, we can calculate the mean-squared error, e_{bb}, on the training data according to Equation 3. We do the same for the pattern model, bam_p, to calculate e_{pp}.

We then calculate the error, e_{pb}, of the base model on the data from the pattern logfile. Likewise, we find the error of the base log data on bam_p to get e_{bp}. In practice, since these error values are all based on offline data, they are actually calculated during the offline phase and saved in the model file to conserve time online. From these values, we compute the following ratio, which is a measure of the difference between the base and pattern models, relative to their own training errors:

$$r_{b,p} = \frac{e_{bp} + e_{pb}}{e_{bb} + e_{pp}} \tag{4}$$

Now, from the data that the coach has collected during the online phase, we construct a ball attraction model bam_o and find the mean-squared error, e_{oo}. We then use the samples from the base strategy file on bam_o to find e_{bo} and vice versa to find e_{ob}. Note that this requires us to keep around all of the data from the offline phase. While it may be possible to approximate the error well using only part of the data, we found that there was sufficient time to use the complete data set.

We can now compute how close the current online game is to the base strategy with the following ratio:

$$r_{b,o} = \frac{e_{bo} + e_{ob}}{e_{oo} + e_{bb}} \tag{5}$$

Likewise, we test bam_o on the samples from the pattern log file and vice versa to calculate e_{po} and e_{op}. We compute $r_{p,o}$ as above. Each of these ratios is greater than or equal to 1. We can calculate *distance* values from these ratios, by simply subtracting one:

$$\begin{aligned} dist(b,p) &= r_{b,p} - 1 \\ dist(b,o) &= r_{b,o} - 1 \\ dist(p,o) &= r_{p,o} - 1 \end{aligned} \tag{6}$$

This definition of distance has the property that the distance of a model to itself is always 0.

Finally, we can calculate the score for this feature as the percentage distance of the online model to the base model:

$$score = \frac{dist(b,o)}{dist(b,o) + dist(p,o)} \tag{7}$$

As intended, if the online model is close to the pattern, then $score =\ 1$. If it's closer to the base strategy, then $score =\ 0$.

Once we compute the scores for all of the features, we can sum them to find the overall score for the pattern. After the coach scores all of the patterns, it ranks them and declares some of them to be active, based on the time of the game.

The predefined cycle times for announcement are as follows. Recall that a 10-minute game lasts for 6000 cycles. At cycle 800, the coach executes the first round of the scoring procedure. After the roughly 8 or 9 cycles used for scoring, the 2 most certain patterns are declared. Then, at cycle 2500, the patterns are scored again and the 3 most certain patterns that have not yet been declared are announced.

These announcement times were determined empirically. Since the scoring metric favors early announcements, we pushed up the announcement time as early as safely possible, without sacrificing accuracy. In informal preliminary experiments, we determined that at cycle 2500, the top 5 patterns rarely differed significantly from the top 5 at cycle 6000 (the end of the game). Similarly, we found that the top 2 patterns at cycle 800 were almost always in the top 5 at cycle 2500.

Pattern Normalization

During initial tests of our scoring algorithm, we found that some patterns would consistently produce much higher scores (by a factor of 100) than other patterns. Regardless of which pattern was activated in a test game, the coach generated very similar pattern rankings. In other words, it thought that all opponents exhibited the same pattern. Thus an absolute ranking of pattern scores was uninformative. However, the patterns' *relative* scores did respond in the correct way by increasing or decreasing when an online opponent did or did not exhibit the pattern. To make these relative responses impact the pattern rankings, we introduced a method for normalizing pattern scores.

Normalization is performed after log file analysis, in the time remaining in the offline phase. The coach simulates numerous online phases to score each pattern multiple times using the pattern log files in place of online data. In the competition, each round included 16 pattern log files, one for each pattern. Therefore, a pattern's score was calculated for 16 simulated games. The pattern's average score was stored as a weight in its corresponding pattern file. Then, during the real online phase, the raw pattern score is divided by its weight, yielding the score used in ranking.

Results

The UT Austin Villa coach won the World Championship at RoboCup 2005 in Osaka, Japan. However, technical errors almost kept us from making it to the final round of the competition. In the first round, the coach binary crashed, but because only 8 of the 10 registered teams submitted an entry, we were by default among the top 8 teams to advance. In the second round, evidence strongly suggested that our normalization script failed, which caused our coach to declare a very similar pattern set in each iteration. Luckily, its score of -13,199 was high enough to attain fourth place, barely sufficient to advance. In the final round, our coach had no major problems and was able to score the highest of the four remaining teams. Competition scores from 2nd and 3rd round are shown in Table 1.

Though it's a positive indicator to win the competition, in practice, the main purpose of such a competition is to serve as a deadline for completing the implementation of a complete autonomous agent capable of executing all phases together. Since every round is run only once, nothing can be concluded about the reproducibility of the results. In order to make any conclusions with confidence, additional controlled experimentation is needed.

Team	Round 2 Score	Round 3 Score
UT Austin Villa	-13,199	65,841
Aria	18,783	31,178
Kasra	1,750	12,180
Susa	34,700	1,800

Table 1: Total scores in the second and third rounds of the competition for the top 4 finishers. Each round score is the sum of 3 game scores.

With that goal in mind, after the competition, we ran controlled experiments to obtain more complete data regarding coach performance. We reconstructed the competition, using the same base and pattern log files and CLANG files. For each round, because the log files don't change, the offline phase was run only once. However, the online phase was run several times. In each instance, we ran the 3 games that comprised the 3 iterations of the round with the exact CLANG used in the competition.

The results of our post-competition testing are shown in Table 2. In our experiments, our coach produced a mean second round score of 48,240 over 5 full-round tests, with individual scores ranging from 44,400 to 63,600 and a standard deviation of 7,680. Between rounds 2 and 3, we adjusted our own team's CLANG advice and the online coach's pattern declaration times. In round 2, we found that our players were inactive during free kicks, wasting valuable time. To remedy this problem, prior to round 3, we added a CLANG command instructing our players to kick the ball inbounds during free kicks. This 3rd round coach was also tested post-competition on the 2nd round pattern set. It had very similar results over 10 full-round tests, achieving a mean score of 45,961 ± 23,855. These results support our suspicion that the failure of our normalization process caused our low 2nd round score in the competition.

In the final round of the competition, our coach performed very well, correctly identifying 7 out of the 14 patterns. Its score of 65,841 was more than double that earned by the second place team, Aria, which also identified the patterns correctly but made few announcements. In post-competition testing, UT Austin Villa averaged 78,680 points over 10 full-round tests, with individual scores ranging from 39,891 to 139,337 and a standard deviation of 28,342. We didn't run the Round 2 coach on the Round 3 data, as this coach is neither the official version from that round nor our current best.

Adding the lower mean of our round 2 testing and our round 3 competition results, we earned a total of 111,802 points. In comparison, the combined scores of the final ranked 2nd, 3rd and 4th teams were 49,961, 13,930, and 36,500, respectively. These results suggest

Coach Version	Round 2 Score	Round 3 Score
Round 2	48,240	-
Round 3	45,961	78,680

Table 2: Average scores for the current and Round 2 version of the coach on patterns from Round 2 and Round 3. The scores for the Round 3 coach are averaged over 10 games each. The score of the Round 2 coach on the Round 2 data was averaged over 5 games.

that UT Austin Villa was indeed the rightful champion of the competition. Our coach appears to be at least as strong as the competition results indicate. At the same time, because we do not have as many data points for the other coaches, it is possible that the competition results underrepresent their true performance.

Related Work

Some previous work has been done on learning to give advice to RoboCup simulated soccer players. Riley et al. (2002) approached advice-giving as an action-prediction problem. Both offensive and defensive models were generated using the C4.5 (Quinlan 1993) decision tree learning algorithm. Their work also stressed the importance of learned formation advice. Subsequently, Kuhlmann et al. (2005) decomposed the problem similarly, but using different model representations and advice-generation procedures.

In other work, Riley and Veloso (2002) 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 (2000) 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 could classify fixed duration *windows* of behavior using a set of sequence-invariant action features.

Opponent team modeling has also been studied in military-like scenarios. In addition to Tambe's work mentioned in the introduction (Tambe 1996), Sukthankar and Sycara (2005) use HMMs to monitor and classify *human* team behavior in a MOUT (military operations in urban terrain) scenario, especially focussing on sequential team behaviors.

Conclusion and Future Work

The RoboCup simulation coach competition presents a detailed and challenging domain for autonomous agents that is specifically geared towards opponent modeling. UT Austin Villa is a complete and fully implemented agent that advises a team of soccer-playing agents and identifies patterns in the opponent teams based on statistical patterns in their positions over time. Despite winning the 2005 RoboCup simulation coach competition, UT Austin Villa leaves much room for improvement. For example, there are many additional potential features that could boost performance within the same framework, and the announcement strategy could be extended to explicitly model the likelihood of correctness relative to the cost in score from waiting longer to announce. Ultimately, it is important to build on this work towards a team that is able to exploit the opponent model on-line as well, both in the coach competition and in other team adversarial domains.

Acknowledgements

This research is supported in part by NSF CAREER award IIS-0237699.

References

Chen, M.; Foroughi, E.; Heintz, F.; Kapetanakis, S.; Kostiadis, K.; Kummeneje, J.; Noda, I.; Obst, O.; Riley, P.; Steffens, T.; Wang, Y.; and Yin, X. 2003. Users manual: RoboCup soccer server manual for soccer server version 7.07 and later. Available at `http://sourceforge.net/projects/sserver/`.

Kitano, H.; Kuniyoshi, Y.; Noda, I.; Asada, M.; Matsubara, H.; and Osawa, E. 1997. RoboCup: A challenge problem for AI. *AI Magazine* 18(1):73–85.

Kok, J. R.; de Boer, R.; Vlassis, N.; and Groen, F. 2003. Towards an optimal scoring policy for simulated soccer agents. In Kaminka, G. A.; Lima, P. U.; and Rojas, R., eds., *RoboCup-2002: Robot Soccer World Cup VI*. Berlin: Springer Verlag. 292–299.

Kuhlmann, G.; Stone, P.; and Lallinger, J. 2005. The UT Austin Villa 2003 champion simulator coach: A machine learning approach. In Nardi, D.; Riedmiller, M.; and Sammut, C., eds., *RoboCup-2004: Robot Soccer World Cup VIII*. Berlin: Springer Verlag. 636–644.

Noda, I.; Matsubara, H.; Hiraki, K.; and Frank, I. 1998. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence* 12:233–250.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Riley, P., and Veloso, M. 2000. On behavior classification in adversarial environments. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*.

Riley, P., and Veloso, M. 2002. Recognizing probabilistic opponent movement models. In Birk, A.; Coradeschi, S.; and Tadokoro, S., eds., *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Berlin: Springer Verlag.

Riley, P.; Veloso, M.; and Kaminka, G. 2002. An empirical study of coaching. In Asama, H.; Arai, T.; Fukuda, T.; and Hasegawa, T., eds., *Distributed Autonomous Robotic Systems 5*. Springer-Verlag. 215–224.

Stone, P. 2000. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press.

Sukthankar, G., and Sycara, K. 2005. Automatic recognition of human team behaviors. In *Proceedings of Modeling Others from Observations, Workshop at the International Joint Conference on Artificial Intelligence (IJCAI)*.

Tambe, M. 1996. Tracking dynamic team activity. In *National Conference on Artificial Intelligence(AAAI96)*.

Veloso, M.; Stone, P.; and Bowling, M. 1999. Anticipation as a key for collaboration in a team of agents: A case study in robotic soccer. In Schenker, P. S., and McKee, G. T., eds., *Proceedings of SPIE Sensor Fusion and Decentralized Control in Robotic Systems II*, volume 3839, 134–143. Bellingham, WA: SPIE.