An Empirical Study of Coaching

Patrick Riley, Manuela Veloso, and Gal Kaminka*

Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213-3891

Abstract. In simple terms, one can say that team coaching in adversarial domains consists of providing advice to distributed players to help the team to respond effectively to an adversary. We have been researching this problem to find that creating an autonomous coach is indeed a very challenging and fascinating endeavor. This paper reports on our extensive empirical study of coaching in simulated robotic soccer. We can view our coach as a special agent in our team. However, our coach is also capable of coaching other teams other than our own, as we use a recently developed universal coach language for simulated robotic soccer with a set of predefined primitives. We present three methods that extract models from past games and respond to an ongoing game: (i) formation learning, in which the coach captures a team's formation by analyzing logs of past play; (ii) set-play planning, in which the coach uses a model of the adversary to direct the players to execute a specific plan; (iii) passing rule learning, in which the coach learns clusters in space and conditions that define passing behaviors. We discuss these techniques within the context of experimental results with different teams. We show that the techniques can impact the performance of teams and our results further illustrate the complexity of the coaching problem.

1 Introduction

As multi-agent systems continue to grow more important, the types of relationships between agents continue to be studied. One important relationship that humans often exhibit is still largely lacking among our agents. This relationship is one of a coach or advisor who provides advice to others. We consider this to be the central feature of a coach relationship, and autonomous agents could from benefit from the development of this sort of relationship. One of the primary ways that advice can be generated is through an agent's observations of and experience with the world. Processing past and current observations into a form usable as advice is indeed a challenging problem.

We have implemented a coach for the Soccer Server System [10], a simulated robotic soccer environment. Notably, because of the creation of a standard language CLang [16], coaches and teams from researchers around the world are able to work together. We have worked towards this research goal of our coach working with a team for which it was not specifically designed. This was the basis for a small coach competition at RoboCup2001 [5] in which four

* This research was sponsored by United States Air Force Grants Nos. F30602-00-2-0549 and F30602-98-2-0135 and by an NSF Fellowship. The content of this publication reflects only the position of the authors.

teams competed. By exploring a few possible techniques for processing observations and providing advice and then evaluating their effects, we hope to further understand the challenges this problem poses. This paper reports on our coaching strategies implemented in simulated robotic soccer and presents the results of our focused experimentation. We believe these results provide a basis for future experimental work, as well as a grounding for more general explication of the coaching problem.

2 Environment

The Soccer Server System is a server-client system that simulates soccer between distributed agents. Clients communicate using a standard network protocol with well-defined actions. The server keeps track of the current state of the world, executes the actions which the clients request, and periodically sends each agent noisy, incomplete information about the world. Agents receive noisy information about the direction and distance of objects on the field (the ball, players, goals, etc.); information is provided only for objects in the field of vision of the agent.

There are 11 independent players on each side as well as a coach agent. The coach agent sees the position and velocity of all players and the ball, but does not directly observe the actions or the perceptions of the agents.

Actions must be selected in real-time, with each of the agents having an opportunity to act 10 times a second. Each of these action opportunities is known as a "cycle." Visual information is sent 6 or 7 times per second. Over a standard 10 minute game, this gives 6000 action opportunities and 4000 receipts of visual information. All units of distance discussed here are simulated meters, with the whole field measuring 105m x 68m.

The communication model between the coach and players was designed to require significant autonomy for the players, especially during the active parts of the games. Basically, the model permits the coach to say one message every 30 seconds (every 300 cycles). Messages are delayed 5 seconds (50 cycles) before being sent to the players.

The coach messages are in a standard coach language called CLang, which was developed by members of the simulated soccer community. Each message basically consists of a set of condition-action rules for the players. The conditions can include relative and absolute positions of the players and the ball as well as the play mode and the player currently controlling the ball. The actions include directions to pass or dribble, move to an area of the field, and "mark" (take a defensive position) against a player or region.

The exact communication model as well as further technical details can be found in [16].

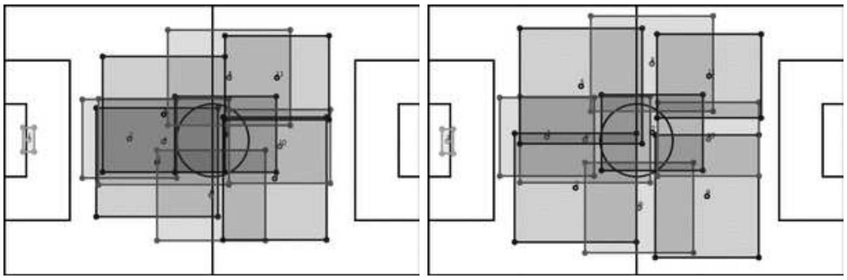
3 Coaching Techniques

This section covers the techniques we use to coach simulated robotic soccer. All of these techniques are designed to learn information about the opponents and how to play effectively against them. Learning about the team to be coached the next research step, as discussed in the empirical results (Section 4).

3.1 Formations by Learning

One important concept in robotic soccer is that of the formation of the team [19]. The concept of formation used by CLang is embodied in the "home area" action. The home area specifies a region of the field in which the agent should generally be. It does *not* require that the agent never leave that area; it is just a general directive.

Our coach represents a formation as an axis aligned rectangle for each player on the team. From the home areas, agents can also a infer a role in the team, with the common soccer distinctions of defenders, midfielders, and forwards.



(a) After Phase 1 (b) After Phase 2

Fig. 1. The learning of the CMUnited99 formation from RoboCup2000 games.

All coaching based on formation uses an algorithm for learning the formation of a team based on observation of that team. The algorithm's input is the set of locations for each player on a team over one or more games. The learning then takes place in two phases.

1. The goal of the first phase is, for each agent, to find a rectangle which is not too big, yet encompasses the majority of the points of where the agent was during the observed games. The learning is done separately for each agent with no interaction between the data for each agent. First the mean position of the agent (c_x, c_y) is calculated, as well as the standard deviation (s_x, s_y). We then do a random search over possible rectangles

(σ is used a parameter for the search). The rectangles to evaluate are generated from the following distribution (for the left, right, top, and bottom of the rectangles), where $N(m, \sigma)$ represents a Gaussian with mean m and standard deviation σ (note that we use a coordinate frame where (0,0) is in the upper left):

$$(N(c_x - s_x, \sigma), N(c_x + s_x, \sigma), N(c_y - s_y, \sigma), N(c_y + s_y, \sigma)) \tag{1}$$

The evaluation function is a weighted sum (with parameter γ which we set to 0.95) of two quantities, both with maximum values of 1. The first involves f, the fraction of points where the agent was which are inside R. We simply use f^β to (where β is a parameter which we set to 1/3). The second quantity uses A (the area of R) and a scaling parameter M (which we set to 900). The evaluation function is then:

$$E(R) = \gamma f^\beta + (1 - \gamma)\left(1 - \frac{A}{M}\right) \tag{2}$$

2. The first phase of learning ignores correlation among the agents. In fact it quite common for all agents to shift one direction or another as the ball moves around the field. This tends to cause the average positions (and therefore the rectangles from phase 1 of the learning) to converge towards the middle of the field, as shown in Figure 1(a). The second phase is designed to capture some pairwise correlations among the agents. The rectangles are moved around, but their shape is not changed.
   For this phase, conceptually think of a spring being attached between the centers of the rectangles of every pair of agents. The resting length for that spring is the observed average distance between the agents. Also, attach a spring with a resting length of 0 between the center of a rectangle and its position at the end of phase 1. A hill-climbing search is then done to find a stable position of the system. Figure 1(b) shows an example result after the second phase of learning.
   Now we describe the details of the algorithm. First, the observed average distance t_{ij} between every two agents is calculated. Next, for each pair of agents, a value α_{ij} roughly corresponding the the tension of the spring in the above description is calculated as follows (w and m are parameters):

$$\alpha_{ij} = e^{m t_{ij}} \quad (i \neq j) \tag{3}$$
$$\alpha_{ii} = w \tag{4}$$

   Eq. (3) provides a higher value (i.e. higher spring tension) between agents which are closer, reflecting the assumption that the correlated movement of nearby agents is more important than those of far away agents. The parameter m (which we set to -0.01) controls that exact weighting. Eq. (4) is used for the connection of an agent to its original position, since Eq. (3) would provide an extremely high weight since $t_{ii} = 0$. The constant w

is set to 0.5, which is the weight calculated by Eq. (3) for a distance of approximately 69 meters.

At each step of the hill-climbing search, a particular agent p is chosen at random to have its rectangle shifted. All other rectangles are held fixed. For all i, let o_i be the original position of rectangle i and let c_i be the vector of the center of current position of rectangle i. The evaluation function used is (where smaller is better):

$$\alpha_{pp} \left(\mathrm{dist}(c_p, o_p)\right)^2 + \sum_{j \neq p} \alpha_{pj} \left(\mathrm{dist}(c_p, c_j) - t_{pj}\right)^2 \tag{5}$$

This simply uses the α values computed in Eqs. (3) and (4) to compute the additive penalty for the imaginary springs not being at their resting length.

The gradient of the evaluation function as a function c_p is easily calculated and a small step is taken in the direction of the gradient (with learning rate 0.001).

Formation learning is used in two ways. The first is an instance of imitation where we imitate the formation of another team. This is especially important for the rule learning described in Section 3.3. The other technique we call "formation based marking." Here the coach observes the previous games of the opponent we will play and learns their formation. Each of the defenders is then assigned one of the forwards of the opponent to mark for the whole game. Ordinarily a team may change its assignment of which defenders mark which forwards. Sending static marking assignments may reduce flexibility, but it also reduces coordination problems and gives the player knowledge of the opponent they did not previously have.

3.2 Set plays

Set plays refer to times of the game when the ball is stopped (due to an out of bounds call, free kick, or kick off) and one team has time to prepare before kicking the ball. Our coach takes advantage of this time to make a plan for the movement of the ball and the agents. This plan is based on refinement of plan templates with a model of the opponent used in evaluating plan changes. Details about this process are described elsewhere [14].

An important difference to be noted is that the plans used in this work were described as a set of rules in CLang rather than as a Simple Temporal Network [7]. Compiling coordination constraints into rule-based systems can be difficult.

3.3 Rule Learning

The passing patterns of a team are an important component in a team's performance. Our coach observes the passes of teams in previous games in order to learn rules which capture some of these passing patterns. These rules can then be used either to imitate a team, or to predict the passes that an opponent will do.

The rule learning uses a combination of clustering (using Autoclass C [3]) to create regions on the field and C4.5 [12] to generate rules describing the passing behavior of a team. The attributes for the rules are the locations of the passer and receiver (using the regions learned from clustering) and the relative position of all teammates and opponents. The rules from C4.5 are then transformed into rules in CLang.

To illustrate, we now provide an example of an learned rule. The format here is almost the format of the CLang language. A few things have been renamed or left out for clarity.

```
1   ((and (play_mode play_on)
2         (bowner our)
3         (bpos "PLINCL0")
4         (ppos our {6} (arc (ball) 23 1000 -180 360))
5         (ppos opp {10} (arc (ball) 0 1000 151 29)))
6     (do our {2 - 11} (bto "PLOUTCL1" {p}))
7     (do our {11} (pos "PLOUTCL1")))
```

Lines 1–5 are the conditions for the rule and lines 6–7 are the actions. Line 2 says that some player on our team is controlling the ball. Line 3 says that that the ball in a particular cluster ("PLINCL0" is the name of the cluster). Lines 4 and 5 are on the position of particular players. Line 4 says that teammate number 6 is at least 23m away, while line 5 says that the angle of opponent number 10 is between 151 and 180 degrees. Note that we do use absolute player numbers here. This is one of the reasons we developed the formation learning techniques described in Section 3.1. As long as the opponent has not changed its formation, the absolute player numbers should be valid. Line 6 instructs all players on our team (except the goalie who is number 1) to pass the ball to the a particular cluster. Line 7 instructs a teammate number 11 (whose home formation position is closest to cluster "PLOUTCL1") to position itself in that region.

4 Experimental Setup and Results

The language CLang was adopted as a standard language for a coach competition at RoboCup2001. Four teams competed providing a unique opportunity to see the effects of a coach designed by one group on the team of another.

We participated in the coach competition, which consisted a single game in each test case. This section reports on our later thorough empirical evaluation of our coach and the techniques used. Each experimental condition was run for 30 games and the average score difference (as our score minus their score) is reported. Therefore a negative score difference represents losing the game and a positive score difference is winning. All significance values reported are for a two tailed t-test.

We use eight teams for our evaluation. We will use initials (denoted in parentheses here) for the teams. The teams that understand CLang are: the DirtyDozen (DD) from University of Osnabrück; and ChaMeleons (CM) from Carnegie Mellon University. Also from RoboCup2001, we use Gemini (GEM) from the Tokyo Institute of Technology and Brainstormers (B) from the University of Karlsruhe. Team descriptions for these teams are available in [5]. From the RoboCup2000 competition, we use the following teams: VirtualWerder (VW) from the University of Bremen; ATHumboldt (ATH) from Humboldt University; and FCPortugal (FCP) from the Universities of Aveiro/Porto (team descriptions can be found in [2]). We also use CMUnited99 (CMU99) from Carnegie Mellon [18], which competed at RoboCup99 and RoboCup2000. In order to run these experiments, we slowed the server down to 3-6 times normal speed so that all agents could run on one machine.

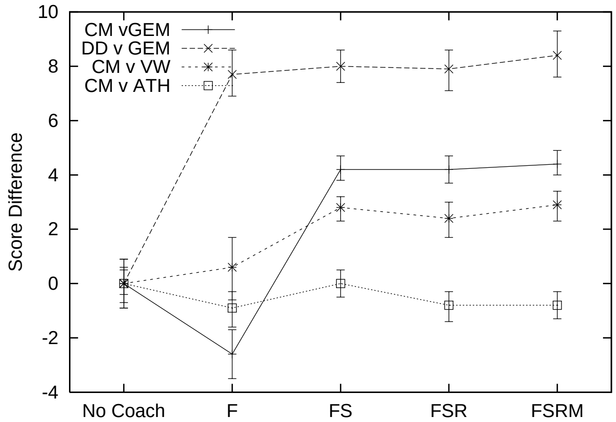
Our experiments aim to separate out the effects of the techniques of our coach. To do this, we ran a sequence of games with different combinations of the five techniques: formation (F) (Section 3.1), set plays (S) (Section 3.2), offensive and defensive rules (R) (Section 3.3), and formation based marking (M) (Section 3.1).

For playing against GEM, our coach observed one game of B playing against GEM. Advice was sent to imitate B's formation and formation based marking was used against GEM's formation. Rule learning was also done for those games. Similarly, our coach learned from 5 games of CMU99 playing against VW and from 10 games of FCP playing against ATH.

The results are shown in Figure 2. The CMvATH set is different from the others in several respects. No combination of the techniques resulted in an improvement for CM, and several combinations (F, FSR, FSRM) resulted in significantly worse performance ($p < .05$) compared to no coach.

For the other teams, the combination of all techniques (FSRM) is always significantly better than no coach ($p < .000002$). Looking at the individual techniques is also illustrative. Sending a formation sometimes helps the team (DD v GEM) and sometimes hurts the performance (CM v GEM), even though exactly the same formation is sent in each case. Even though the advice is the same, the effect on the team being coached is vastly different. From this, we conclude that the coach needs to learn something about the team being coached.

Except for the CM v ATH line, neither the rules nor the formation based marking make a significant impact on the score difference of the games. The formation based marking was a minor part of the coach and it is no great surprise that it's impact is small. The rule learning, however, was the most ambitious of the coaching techniques used. There are several reasons why the rules may have failed to have a large impact. The number of examples from which to learn varied considerably, from 51 to 1638) and so did the accuracy of



CMvGEM	DDvGEM	CMvVW	CMvATH
-6.5 [-7.2,-5.9]	-17.2 [-18.1,-16.3]	-2.8 [-3.7,-1.9]	1.2 [0.8,1.7]

Fig. 2. The score difference of teams coached by a random coach and various techniques of C-CM. The score differences have been additively normalized to the no coach values shown in the lower table. All error bars are 95% confidence intervals. Note that we do not have random coach results for all cases.

the rules on the reserved test set (35%–75%). Some preliminary experiments indicate that changing the input attributes could improve the performance. The attributes are currently based on the absolute player numbers, where sorting the player's by distance to the passer may be useful. This was done primarily because of the expressibility of the current version of CLang, and a new version is in progress (see [16] for details).

5 Related Work

The area of imitation has been studied under many different names. There has been extensive research in the robotics literature on learning a task by imitating a human being, called variously "teaching by guiding," "learning by watching," "programming by demonstration," and "imitation learning." Bakker and Kuniyoshi have a recent survey [1] and Dautenhahn emphasizes the biological connection [6]. Similarly, an area commonly called behavioral cloning deals with learning a control strategy for a task [17, 20]. Imitation is only one possible aspect of successful coaching. In particular for this work, we are imitating one aspect of agent interaction (passing), not simply agent interaction with the environment.

Some work has also been done in creating agents capable of receiving advice. For example, the RATLE system by Maclin and Shavlik [8] can incorporate advice generally specified as if-then rules (similar to the language we use here) into a reinforcement learning agent. Their results in Pengo, a grid and blocks world, also suggest that the learning agents need to be able to refine advice to achieve high performance. Clouse [4] find a similar results in a discrete driving task. They created an automated trainer to improve the learning speed of the learning agent. If the trainer gives too much advice, the learner can fail to converge.

Previous research in Intelligent Tutoring Systems (ITS) has examined how to give advice to human beings. For example, Miller, *et. al.* [9] consider how to give advice to students who are constructing arguments based on scientific data. The system works by comparing the structure of the student's argument (explicitly given by the student) to known patterns. The CAST system [11] trains humans to act in a team. Here, a coach agent provides advice based on tracking belief state of the human being coached. The primary difference between the ITS literature and this research is that tutoring systems generally rely on a fairly rigid and predefined task structure. Deviations from that structure are the focus of the advice. Here, we have no such predefined plan or structure.

The ISAAC system [13] is an automated game analysis tool for simulated robotic soccer. It does off-line analysis of games at several levels. It employs a local adjustment approach to suggest small changes (such as "shoot when closer to the goal") to a team's designer in order to improve performance. The suggestions are backed up by examples from the games analyzed and provided in a format useful for the designers to examine. However, ISAAC's suggestion are provided to the *designer* of the team, not to the agents themselves; there is no automated effect on the team.

6 Conclusion

We have presented several implemented coaching techniques for a simulated robotic soccer domain. We further presented the results from an extensive set of experiments to understand the effects of the coaching techniques presented here, using agents created at a variety of institutions. The experiments represent 630 games and over 20 days of computer time. The experiments justify that coaching can help teams improve in this domain. However, all of our coaching techniques are based on learning about the adversary and not on understanding the functioning of the team to be coached. Consequently, the effect of the advice on different teams varies greatly. Our results support the need for a coach to understand its team in order to achieve robust performance.

These empirical study is a first but significant step in the project of understanding an advice-based relationship between automated agents. We intend to use this experimental basis to aid in the understanding of the general coaching problem (see [15] for one characterization). This research raises many interesting question which we will continue to pursue.

References

1. P. Bakker and Y. Kuniyoshi. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, pages 3–11, Brighton,UK, 1996.
2. T. Balch, P. Stone, and G. Kraetzschmar, editors. *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001.
3. P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *ICML-88*, pages 54–64, San Francisco, June 1988. Morgan Kaufmann.
4. J. Clouse. Learning from an automated training agent. In D. Gordon, editor, *Working Notes of the ICML '95 Workshop on Agents that Learn from Other Agents*, Tahoe City, CA, 1995.
5. A. B. S. Coradeschi and S. Tadokoro, editors. *RoboCup-2001: Robot Soccer World Cup V*. Springer Verlag, Berlin, 2002.
6. K. Dautenhahn. Getting to know each other—artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, 16:333–356, 1995.
7. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
8. R. Maclin and J. W. Shavlik. Creating advice-taking reinforcement learners. *Machine Learning*, 22:251–282, 1996.
9. M. S. Miller, J. Yin, R. A. Volz, T. R. Ioerger, and J. Yen. Training teams with collaborative agents. In *ITS-2000*, pages 63–72, 2000.
10. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
11. M. Paolucci, D. D. Suthers, and A. Weiner. Automated advice-giving strategies for scientific inquiry. In *ITS-96*, pages 372–381, 1996.
12. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.
13. T. Raines, M. Tambe, and S. Marsella. Automated assistant to aid humans in understanding team behaviors. In *Agents-2000*, 2000.
14. P. Riley and M. Veloso. Planning for distributed execution through use of probabilistic opponent models. In *AIPS-2002*, 2002. (to appear).
15. P. Riley, M. Veloso, and G. Kaminka. Towards any-team coaching in adversarial domains. In *AAMAS-02*, 2002. (extended abstract) (to appear).
16. RoboCup Federation, `http://sserver.sourceforge.net/`. *Soccer Server Manual*, 2001.
17. C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *ICML-92*, Aberdeen, 1992. Morgan Kaufmann.
18. P. Stone, P. Riley, and M. Veloso. The CMUnited-99 champion simulator team. In Veloso, Pagello, and Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, pages 35–48. Springer, Berlin, 2000.
19. P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.
20. D. Šuc and I. Bratko. Skill reconstruction as induction of LQ controllers with subgoals. In *IJCAI-97*, pages 914–919, 1997.