A Layered Approach to Learning Client Behaviors in the RoboCup Soccer Server

Peter Stone
pstone@cs.cmu.edu
http://www.cs.cmu.edu/~pstone
(412) 268-7123

Manuela Veloso *
veloso@cs.cmu.edu
http://www.cs.cmu.edu/~mmv
(412) 268-8464

Computer Science Department
Carnegie Mellon University
Pittsburgh PA 15213-3890

In *Applied Artificial Intelligence*, 12, 1998.

Abstract

In the past few years, Multiagent Systems (MAS) has emerged as an active subfield of Artificial Intelligence (AI). Because of the inherent complexity of MAS, there is much interest in using Machine Learning (ML) techniques to help build multiagent systems. Robotic soccer is a particularly good domain for studying MAS and Multiagent Learning. Our approach to using ML as a tool for building Soccer Server clients involves layering increasingly complex learned behaviors. In this article, we describe two levels of learned behaviors. First, the clients learn a low-level individual skill that allows them to control the ball effectively. Then, using this learned skill, they learn a higher-level skill that involves multiple players. For both skills, we describe the learning method in detail and report on our extensive empirical testing. We also verify empirically that the learned skills are applicable to game situations.

1 Introduction

In the past few years, Multiagent Systems (MAS) has emerged as an active subfield of Artificial Intelligence (AI) [20]. Focussing on how AI agents' behaviors can and do interact, MAS applies to a variety of frameworks ranging from information agents to real robots. Because of the inherent complexity of MAS, there is much interest in using Machine Learning (ML) techniques to help deal with this complexity [1, 26].

Robotic soccer is a particularly good domain for studying MAS. It has been gaining popularity in recent years, with international competitions, namely RoboCup and MIROSOT, planned for the near future [22, 9]. Robotic soccer can be used as a standard testbed to evaluate different MAS techniques in a straightforward manner: teams implemented with different techniques can play against each other.

The main goal of any testbed is to facilitate the trial and evaluation of ideas that have promise in the real world. A wide variety of MAS issues can be studied in robotic soccer [20]. In this article, we focus on the multiagent learning opportunities that arise in Noda's Soccer Server [12]. The properties of simulated robotic soccer that make it a good testbed for MAS include:

*This research is sponsored in part by the DARPA/RL Knowledge Based Planning and Scheduling Initiative under grant number F30602-95-1-0018. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the U. S. Government.

- Enough complexity to be realistic;
- Easy accessibility to researchers worldwide;
- Embodiment of most MAS issues, including [20];
  - Ability to support reactive or deliberative agents
  - Need for agents to model other agents
  - Need for agents to affect each other
  - Room for both cooperative and competitive agents
  - Possibility for stable or evolving agents
  - Need for resource management (stamina)
  - Need for social conventions
  - Opportunity for agents to fill different roles
  - Support for communicating agents
  - Opportunity to plan communicative acts
  - Room for exploring commitment/decommitment strategies
- Straightforward evaluation;
- Good multiagent ML opportunities.

Our approach to using ML as a tool for building Soccer Server clients involves layering increasingly complex learned behaviors. We call this approach *layered learning*. Because of the complexity of the domain, it is futile to try to learn intelligent behaviors straight from the primitives provided by the server. Instead, we identified useful low-level skills that must be learned *before* moving on to higher level strategies. Using our own experience and insights to help the clients learn, we acted as human coaches do when they teach young children how to play real soccer.

In this article, we describe two levels of learned behaviors. First, the clients learn a low-level individual skill that allows them to control the ball effectively. Then, using this learned skill, they learn a higher-level, more "social," skill: one that involves multiple players. For both skills, we describe the learning method in detail and report on our extensive empirical testing. Finally, we verify empirically that the learned skills are applicable to a game-like situation.

Although several more layers are needed, the two learned behavior levels described below will allow us to continue moving upward towards high-level strategy issues. Keeping in mind the many open research issues in Multiagent Learning, we plan to use ML techniques at all stages to help the clients develop their behaviors.

In Section 3, we describe our overall robotic soccer systems of which the strategic level is one part. Section 3 presents some previous work that relates to this article. The RoboCup Soccer Server is described in Section 4. Then, in the body of the paper, Sections 5– 7, we present two learned layers of behavior, verify that they are useful in game situations, and discuss future work. Section 8 summarizes our layered learning approach as presented in this article and concludes.

2 The Complete Robotic System

Though conducted in simulation, the work described in this article is intended to contribute to the high-level reasoning aspect of our physical robotic system. The architecture of our system addresses the combination of high-level and low-level reasoning by viewing the overall system as the conjunction of mini-robots, a vision camera over-looking the playing field connected to a centralized interface computer, and several clients as the minds of the mini-robot players. Figure 1 sketches the building blocks of the architecture.

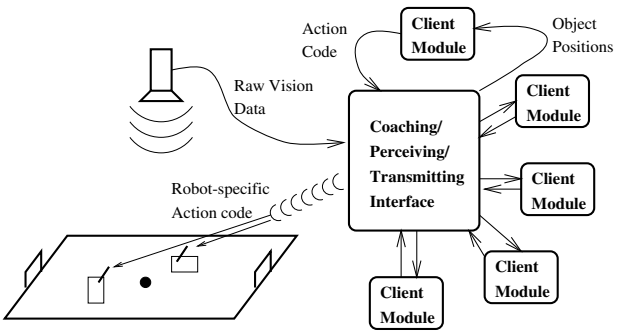


Figure 1: Our robotic soccer architecture as a distributed deliberative and reactive system.

Our architecture implements the overall robotic soccer system as a set of different platforms with different processing features. The mini-robots perform the physical navigation actions, decode commands, and can respond to positioning requests. Off-board computers perceive the environment through a vision camera, perform the high-level decision making and send commands to the mini-robots. Communication between the off-board computers and the robots in our current system is done by infrared radiation. The complete system is fully autonomous consisting of the following processing cycle: (i) the vision system perceives the dynamic environment, namely the positioning of the robots and the ball; (ii) the image is processed and transferred to the host computer that makes the perception available to the client modules; (iii) based on the perceived positioning of the agents and any other needed information about the state of the game (e.g. winning, losing, attacking), each client uses its strategic knowledge to decide what to do next; (iv) the client selects navigational commands to send to its corresponding robot agent; (v) these commands are sent by the main computer to the robots through wireless communication (infrared radiation in the current implementation) using the robot-specific action codes. Commands can be broadcast or sent directly to individual agents. Commands include positioning requests and navigation primitives, such as forward, backward, and turning moves at specific speeds. Each robot has a self identification binary code that is used in the wireless communication.

Figure 2 shows the architecture as a layered functional system. The protocols of communication between the layers are specified in terms of the modular inputs and outputs at each level. It is the layered strategic behaviors (Figure 2(b)) that we hope to enhance with the aid of the simulator client behaviors described in this article.

3 Related Work

A ground-breaking system for Robotic Soccer, and the one that served as the inspiration for our work, is the Dynamo System developed at the University of British Columbia [18]. This system was designed to be capable of supporting several robots per team, but most work has been done in a 1 vs. 1 scenario. Sahota used this system to introduce a decision making strategy called *reactive deliberation* which was used to choose from among seven hard-wired behaviors [16]. Our system differs from the Dynamo system

Layer	Entity	Input	Output
Behavioral	Robots	Commands	Actual Moves
Perceptual	Vision Camera	View of Field	Robots & Ball Coordinates
Strategic	Computer	Robots & Ball Coordinates	Commands

(a)

Layered Strategic Level	Examples
Robot–ball	intercept
One-to-one player	pass, aim
One-to-many player	pass to a teammate
Action selection	pass, dribble, or shoot
Team collaboration	strategic positioning

(b)

Figure 2: (a) The functional layers of the architecture, and (b) the strategic level decomposition.

in several ways, most notably in that teams consist of several robots, thus necessitating the development of cooperative behaviors. We also hope to do minimal hard-wiring, instead learning behaviors from the bottom up.

The Robotic Soccer system being developed in Asada's lab is very different from both the Dynamo system and from our own [3, 25]. Asada's robots are larger and are equipped with on-board sensing capabilities. They have been used to develop some low-level behaviors such as shooting and avoiding as well as a RL technique for combining behaviors [3, 25]. While the goals of this research are very similar to our own, the approach is different. Asada has developed a sophisticated robot system with many advanced capabilities, while we have chosen to focus on producing a simple, robust design that will enable us to concentrate our efforts on learning low-level behaviors and high-level strategies. We believe that both approaches are valuable for advancing the state of the art of robotic soccer research.

Although real robotic systems, such as those mentioned above and the many new ones being built for robotic soccer tournaments [22, 9], are needed for studying certain robotic issues, it is often possible to conduct research more efficiently in a well-designed simulator. Several researchers have previously used simulated robotic soccer to study ML applications. Using the Dynasim soccer simulator [17, 16], Ford et al. used a Reinforcement Learning (RL) approach with sensory predicates to learn to choose among low-level behaviors [5]. Using a simulator based closely upon the Dynasim system, Stone and Veloso used Memory-based Learning to allow a player to learn when to shoot and when to pass the ball [19]. They then used Neural Networks to teach a player to shoot a moving ball into the goal [21]. In the RoboCup Soccer Server Matsubar et al. used a Neural Network to allow a player to learn when to shoot and when to pass [11] (as opposed to the Memory-based technique used by Stone and Veloso for a simlar task). The work described in this article uses Neural Networks and Decision Trees to learn different behaviors in the RoboCup Soccer Server.

A wide variety of MAS research is related to the layered learning approach espoused in this paper. Most significantly, Mataric uses Brooks' Subsumption Architecture [4] to build multiagent behaviors on top of a set of learned *basis behaviors* [10]. Mataric's basis behaviors are chosen to be necessary and sufficient for the learning task, while remaining as simple and robust as possible. Since Mataric's robots were to learn social behaviors such as flocking and foraging, they were equipped with basis behaviors such as the ability to follow each other and the ability to wander without running into obstacles. While our approach makes

similar use of multiple behavior layers, we are more committed to using ML whenever possible in order to study the interactions between learned behaviors at different levels.

One of the learned behaviors described in this article involves players that assume different roles. Although the roles are fixed in the current implementation, the players will eventually need to change roles as a match progresses. Tambe discusses a framework in which agents can take over the roles of other teammates in a helicopter-combat domain [23]. In the learning context, Prasad et al. have created design agents that can learn which role to fill [13]. We plan to combine role learning with dynamic role assumption as we progress to higher levels of learned behaviors (see Section 7).

In addition to reasoning about roles of teammates, Tambe's combat agents can also reason about the roles that opponents are playing in team behaviors [24]. By recognizing an opponent's action as a part of a larger team action, an agent is able to more easily make sense of the individual opponent's behavior with the goal of being able to predict the opponent's future actions. This work enhances previous work that aims at having agents deduce other agents' intentions through observation [8].

4 The Simulator

Extensive experimentation of the type described in this article is not feasible with physical robotic systems. Consequently, to conduct meaningful research in simulation that might apply to the real world, a well-designed simulator is needed. Though not directly based upon any single robotic system, Noda's *Soccer Server* [12], pictured in Figure 3, captures enough real-world complexities to be an indispensable tool. This simulator is realistic in many ways: (i) the players' vision is limited; (ii) the players can communicate by posting to a blackboard that is visible to all players; (iii) all players are controlled by separate processes; (iv) each player has 10 teammates and 11 opponents; (v) each player has limited stamina; (vi) actions and sensors are noisy; and (vii) play occurs in real time. The simulator, acting as a server, provides a domain and supports users who wish to build their own agents (clients).

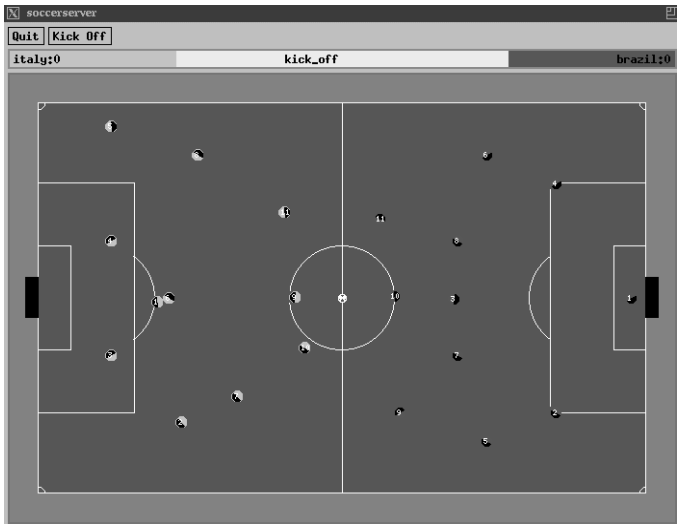
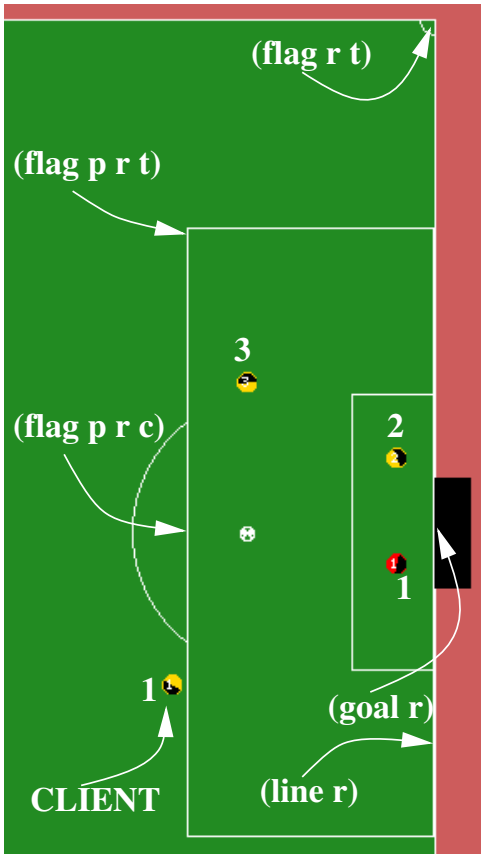


Figure 3: The Soccer Server system

Figure 4 illustrates the format of the communication between the server and a specific client. Since the client's vision is limited to 45^o on either side, not all objects are visible at each sensory step. For example, at the beginning of the trace in Figure 4, the client sees two teammates and one opponent (player brazil 1). However after dashing once, it is no longer able to see the opponent. Similarly, after dashing a second time,




```
(see 124 ((goal r) 20.1 34) ((flag r t) 47.5 -4) ((flag p r t) 30.3 -24) ((flag p r c) 10.1 -20)
 ((ball) 11 0) ((player usa 2) 21 19) ((player usa 3) 21 -11) ((player brazil 1) 17 35) ((line r) 40 -26))
**-> (dash 80)
(see 129 ((goal r) 16 43) ((flag r t) 42 -6) ((flag p r t) 25 -30) ((flag p r c) 5 -40) ((ball) 6 1)
 ((player usa 2) 16.3 24) ((player usa 3) 15.3 -17) ((line r) 32.8 -27))
**-> (turn 1)
**-> (dash 60)
(see 134 ((flag r t) 40 -9) ((flag p r t) 23.3 -35) ((ball) 3.7 2) ((player usa 2) 14.4 24)
 ((player usa 3) 13.3 -22) ((line r) 28.2 -30))
**-> (turn 2)
**-> (dash 30)
(hear 138 18 shoot the ball)
(see 139 ((flag r t) 38.1 -11) ((flag p r t) 22 -39) ((ball) 1.9 0) ((player usa 2) 12.8 27)
 ((player usa 3) 11.6 -27) ((line r) 25.5 -31))
**-> (say shooting now)
**-> (kick 53 51)
(hear 141 self shooting now)
(see 144 ((flag r t) 38.1 -11) ((flag p r t) 22 -39) ((ball) 8.1 42) ((player usa 2) 12.8 27)
 ((player usa 3) 11.6 -27) ((line r) 25.5 -31))
**-> (turn 42)
(see 149 ((goal r) 13.6 9) ((ball) 13.5 5 0) ((player usa 2) 12.8 -14) ((player brazil 1) 11 18)
 ((line r) 14 -73))
**-> (turn 5)
**-> (dash 81)
(hear 150 referee goal_l_1)
(hear 150 referee kick_off_r)
```

Figure 4: A trace of the simulator's input and output to the client controlling player 1 (indicated "CLIENT"). The player moves to the ball and then shoots it towards the goal. Commands from the player are indicated with "**-> " preceding them. Dashes are followed by a power (they are always in the direction that the player is facing), turns are followed by an angle, and kicks are followed by a power and an angle. Sensory information from the server comes in the form of audial and visual strings. In both cases, the number after the type indicator ("hear" or "see") indicates the elapsed time in the match. Audial information then indicates whether it is the referee speaking or else from what angle on the sound came. Visual information includes the distance followed by angle of the visible object.

it is no longer able to see the center of the penalty area: (flag p r c). After kicking the ball, when the client turns to face the goal, the opponent comes back into view.

The method of communication is illustrated by the message from teammate number 2 that is heard at time 144 ("shoot the ball"), and by the spoken response "shooting now." Two messages from the referee, indicating the successful goal and the subsequent restart, are also present at the end of the trace.

Both the sensors and the actions in the simulator are noisy. Notice that even though the player begins by facing directly at the stationary ball (the ball's angle is 0) and dashes straight toward it, the ball does not remain directly in front of the player. Before its next two dashes, the client turns to face the ball again. Although not apparent from this trace, when players are far enough away, their uniform numbers, or even their team, may not be visible.

The trace in Figure 4 begins at elapsed time 124 and continues through 150. Since each time increment occurs in 0.1 second of real time, visual sensor information arrives twice a second, and the entire trace, from the moment pictured in Figure 4 until the ball enters the goal, occurs in about 2.5 seconds. Thus, the action in the simulator occurs in real time.

All of these simulator features combine to make it a very challenging and realistic environment in which to conduct research. The lessons we learn in the simulator will help us greatly as we implement the strategic levels of our real robotic system.

5 Learning a Low-level Skill

Just as young soccer players must learn to control the ball before learning any complex strategies, Soccer Server clients must also acquire low-level skills before exhibiting complex behaviors: the most sophisticated understanding of how to act as part of a team is useless without the ability to execute the necessary individual tasks. Although general agents can be assumed to possess basic domain-independent skills such as moving and sensing, there are always new skills to learn in a new domain. Acting as human coaches for our clients, we identified a low-level skill that is needed in the Soccer Server. Isolating a situation that requires this skill, we drilled the clients, providing the appropriate reinforcement, until they were able to learn to execute this skill reliably.

The low-level skill we identified as being most essential to our Soccer Server clients was the ability to intercept a moving ball. This skill is ubiquitous in all soccer-type frameworks as indicated by the fact that we taught clients a similar skill in a different simulator [21]. Intercepting a moving ball is considerably more difficult than moving to a stationary ball both because of the ball's unpredictable movement (due to simulator noise) and because the client may need to turn and move in such a direction that it cannot see the ball (see Figure 5).

Intercepting a moving ball is a task that arises very frequently in the Soccer Server. Unless the ball has decelerated completely without a player collecting it, this skill is a prerequisite for any kicking action. In particular, defenders must intercept shots and opponents' passes, while players must frequently "intercept" passes to them from teammates. In many of these situations, the ball is moving roughly in the direction of the player that is trying to intercept it, the condition which causes the difficulty illustrated in Figure 5. The ball can move past the player as it goes to where the ball used to be. This problem arises primarily because the defender gets sensory information at discrete intervals (250 msec).

Faced with two possible methods for equipping our players with the ability to intercept a moving ball—empirical and analytical—we chose the empirical method for its appropriateness to our Machine Learning paradigm. Rather than providing the clients with the ability to perform sophisticated geometric calculations, we provided the clients with a large number of training examples and used a supervised learning technique: Neural Networks (NNs).

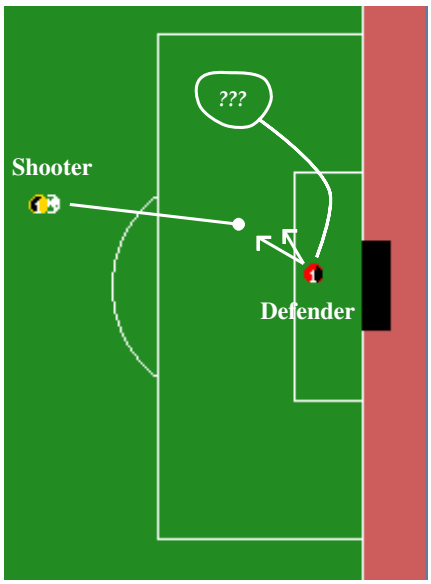


Figure 5: If the defender moves directly towards the ball (left arrow), it will miss entirely. If the defender turns to move in the appropriate direction (right arrow), it may no longer be able to see the ball.

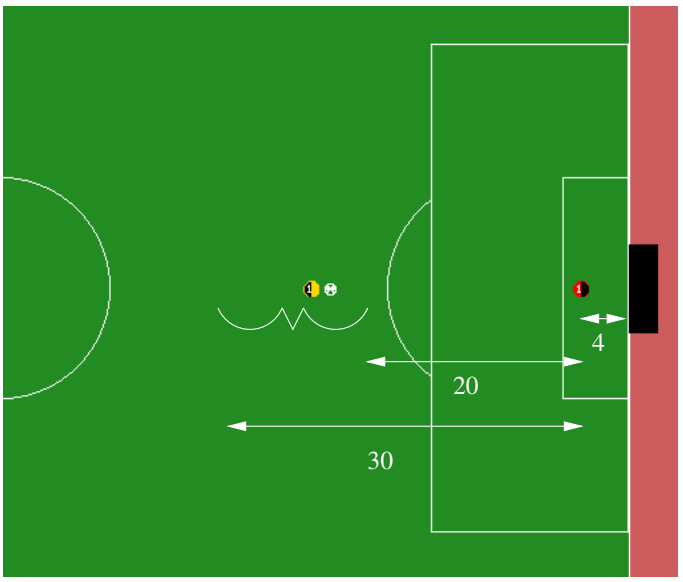


Figure 6: At the beginning of each trial, the defender starts 4 units from the goal, while the ball and shooter are placed randomly between 20 and 30 units from the defender.

The range of situations from which training examples were gathered is illustrated in Figure 6. For each training example, the *shooter* kicks the ball directly towards the *defender* with a fixed power. However, due to the noise in the simulator, the ball does not always move directly at the defender: if the defender remains still, the ball hits it only 35% of the time. Furthermore, if the defender keeps watching the ball and moving directly towards it, it is only able to stop the ball 53% of the time.

The defender's behavior during training is more complex than the shooter's. As we are using a supervised learning technique, it must first gather training data by acting randomly and recording the results of its actions. It does so as follows (BD = Ball's distance, BA = Ball's angle, TA = *Turn angle* — the angle to turn *after* facing the ball):

- While BD $>$ 14, TURN(BA)
- When BD $\leq$ 14, set TA = Random Angle between -45 and 45
- Record BD, BA, previous BD, and TA
- TURN(BA + TA)
- DASH()
- Record result (from coach)

Until the ball is within a given range, the defender simply watches and faces the ball. Then, once the ball is in range, the defender turns a random angle (within a range) away from the ball and dashes. Of course the defender misses most (76%) of the time, but after about 750 positive examples, it is able to learn to perform much better (see below).

In order to automate the training process, a coach client is used. The coach ends a trial when the ball gets past the defender or when it starts moving back towards the shooter. In the latter case, the trial is labeled a *SAVE*. In the former case, it is labeled a *GOAL* if the ball is still between the goal posts and a *MISS* if it is heading wide of the goal. Only saves are considered positive results and thus used for training. At the end of the trial, the coach resets the positions of both players and the ball for another trial.

The goal of learning is to allow the defender to choose the appropriate turn angle (TA) based upon the BD, BA, and previous BD. Thus, only the data from the saves during the training phase are useful (the NN is learning a continuous, not a binary, output). In order to learn the TA, we chose to use a Neural Network (NN). Other supervised learning techniques, such as memory-based learning, could also have worked (see below). After a small amount of experimentation with different NN configurations, we settled on a fully-connected net with 4 sigmoid hidden units and a learning rate of 10^{-6}. The weights connecting the input and hidden layers used a linearly decreasing weight decay starting at .1%. We used a linear output unit with no weight decay. We trained for 3000 epochs. This configuration proved to be satisfactory for our task with no need for extensive tweaking of the network parameters.

In order to test the NN's performance, we ran 1000 trials with the defender using the output of the NN to determine its turn angle. The behaviors of the shooter and the coach were the same as during training. The results for NNs trained with different numbers of training examples are displayed in Figure 7. The misses are not included in the results since those are the shots that are far enough wide that the defender does not have much chance of even reaching the ball before it is past. The figure also records the percentage of shots *on-goal* (Saves+Goals) that the defender saved. Reasonable performance is achieved with only 300 save examples, and examples beyond about 750 do not improve performance. The defender is able to save almost all the shots despite the continual noise in the ball's movement.

Training Examples	Saves(%)	Goals(%)	$\frac{\text{Saves}}{\text{Goals+Saves}}$(%)
100	57	33	63
200	73	18	80
300	81	13	86
400	81	13	86
500	84	10	89
750	**86**	9	**91**
1000	83	10	89
4773	84	9	90

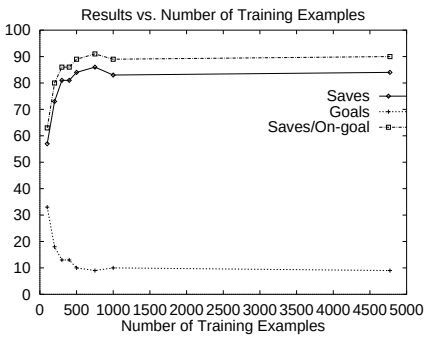


Figure 7: The defender's performance when using NNs trained with different numbers of positive examples. The last column of the table indicates the percentage of shots that were "on goal" that the defender saved.

In order to study the effect of noise in the ball's movement upon the defender's performance, we varied the amount of noise in the Soccer Server (the `ball_rand` parameter). Figure 8 shows the effect of varying noise upon the defender when it uses the trained NN (trained with 750 examples) and when it moves straight towards the ball. The default value of noise is .05, meaning that on every simulator step, the true position of the ball is perturbed by a random amount between -.05 and .05 with uniform probability distribution over the range. The "straight" behavior always sets TA=0, causing the defender to go directly towards where it last saw the ball. Notice that with no ball noise, both the straight and learned behaviors are successful: the ball and the defender move straight towards each other. As the noise in the ball's motion increases, the advantage to using the learned interception behavior becomes significant. The advantage of the NN can also be seen with no noise if the shooter aims slightly wide (by 4 degrees) of the goal's center. Then the defender succeeds 99% of the time when using the NN, and only 10% of the time when moving straight

Noise	Behavior	Saves(%)	Goals(%)	$\frac{\text{Saves}}{\text{Goals+Saves}}$(%)
0	NN	100	0	100
	Straight	100	0	100
.05	NN	**86**	9	**91**
	Straight	**53**	35	**60**
.06	NN	75	13	86
	Straight	47	35	57
.07	NN	68	14	83
	Straight	40	36	53
.08	NN	59	16	78
	Straight	34	36	49
.09	NN	53	17	75
	Straight	32	33	50
.1	NN	49	18	73
	Straight	28	32	47

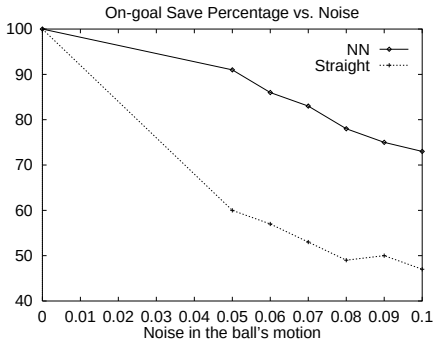


Figure 8: The defender's performance when using NNs and moving straight with different amounts of ball noise.

towards the ball.

It appears that the NN solution allows the defender to intercept a moving ball as well as can be hoped for given the discrete sensory events and simulator noise. However, as mentioned above and particularly because this problem may not be overly complex, there are other promising ways of approaching the problem. For example, since we notice that the NN weighs one of the inputs much more heavily than the other two, it appears that memory-based techniques would work quite well.

From a quick examination of the inputs and outputs of the trained NN, it appears that the NN focusses primarily upon the Ball's angle (BA) [1]. Consequently, we were curious to try a behavior that simply used a lookup table mapping BA to the typical output of the NN for that BA. We identified such outputs for BA's ranging from -7 to 7. Using this one dimensional lookup-table, the defender was able to perform almost as well as when using the full NN (see Table 1). Although the lookup table was built with the aid of a NN, the one-dimensional function could also be easily learned with memory-based methods.

We also were curious about how well the NN would compare to analytical methods. As a basis for comparison, we used a behavior constructed by Mike Bowling, an undergraduate student in the project, whose goal was to create the best possible analytic behavior. The resulting behavior computed the ball's motion vector from its 2 previous positions and multiplied this vector by 3, thus predicting the ball's position two sensory steps (500 msec) into the future. The defender's TA was then the angle necessary to move directly towards the end of the lengthened vector. Using this technique, the defender was also able to perform almost as well as it did when using the NN (see Table 1).

In this section we verified that a supervised learning technique was useful for learning a low-level skill that is needed by clients in the Soccer Server. Although there were other possible methods of attaining this skill, the method we chose is at least as good as the other options. Furthermore, it is in keeping with our goal of building up learning clients by layering learned behaviors on top of each other.

[1] It was similarly noticed in [2] that the ball's angle is more useful for learning than is the ball's distance.

Defender Behavior	Saves(%)	Goals(%)	Saves/(Goals+Saves) (%)
NN	**86**	9	**91**
Lookup Table	83	8	91
Analytic	82	13	86

Table 1: The defender's performance when using a NN, a one-dimensional lookup table, and an analytic method to determine the TA.

6 Learning a Higher-level Decision

Once young soccer players have learned how to control the ball, they are ready to use their skills to start learning how to make decisions on the field and playing as part of a team. Similarly, our clients can use their learned ball-interception skill to exhibit a more complex behavior: passing. Passing requires action by two different clients. A *passer* must kick the ball towards the *receiver*, who must collect the ball. Since the receiver's task is identical to that of the defender in the previous section, the clients can (and do) use the same trained NN.

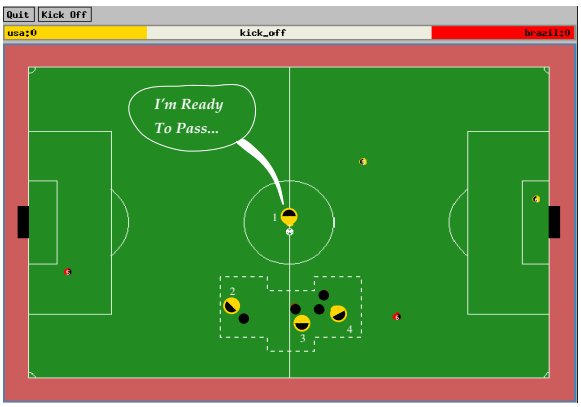
Although the execution of a pass in the open field is not difficult given the receiver's ball-interception skill, it becomes more complicated in the presence of defenders. If in the proper position, a defender (also equipped with the same ball-interception skill) may be able to intercept the ball before it reaches the receiver. Thus, the passer is faced with the task of assessing the likelihood that a pass to a particular receiver will succeed. For example, in Figure 9 the left-most teammate may be able to receive a pass, while the two directly to the right are much less likely to be able to do so. The higher-level decision that our clients learned was whether or not to pass to a given teammate.

Just as this behavior builds upon the interception skill, higher-level behaviors can be built upon this knowledge of when a pass will succeed. Such knowledge can contribute to the decision of which player to pass to or whether to pass, dribble, or shoot.

When deciding whether or not to make a pass, the passer has many possible features of the scenario at its disposal. When many features are available, it can be very difficult to pick out the ones that are relevant for building an analytical model. Rather than going through and filtering the attributes by hand, we chose to use a learning method that is capable of determining for itself which attributes to use. In particular, we used Decision Trees (DTs).

In order to gather the training data, we again defined a constrained situation and used a coach client to monitor the trials. Since passing requires coordination of the passer and the receiver, each trial was somewhat involved:

- The coach randomly placed the players (Figure 9).
- The passer announced its intention to pass (Figure 9).
- The receivers replied with their views of the field when ready to receive (Figure 10).
- The passer chose a receiver randomly during training, or with a DT during testing (Figure 11).
- The passer recorded a large number of attributes describing the trial (see below).
- The passer announced who it was passing to (Figure 12).
- The receiver and 4 defenders attempted to get the ball **using the learned ball-interception skill** (Figure 13).



1 Passer: 3 Receivers: 4 Defenders:

+2 other players per team

Figure 9: At the beginning of a trial, the passer is placed behind the ball. 3 teammates and 4 opponents are placed randomly within the region indicated by the dashed line, while 2 other players from each team are placed randomly on the field. In the following figures, the players involved in the play are enlarged for presentation purposes. When the passer sees that it has the ball, it announces its intention to pass. Its goal is to assess the likelihood of a pass to a given teammate succeeding.

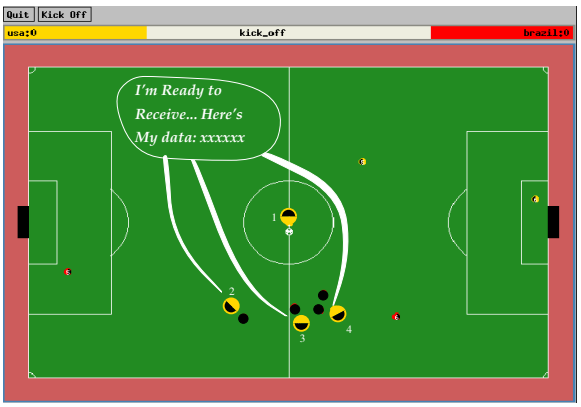
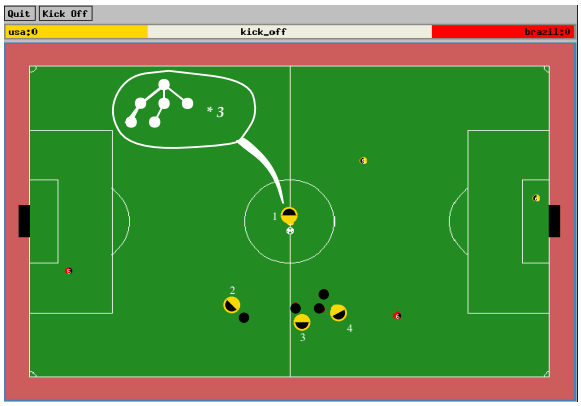


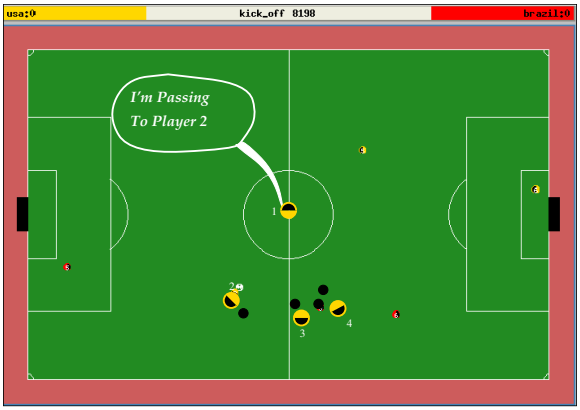
Figure 10: When the receivers are facing the ball, they tell the passer what the world looks like to them. The passer can use the transmitted data to help it assess the likelihood that each receiver would successfully receive a pass. The data includes distances and angles to the other players as well as some counts of players within given distances and angles.

- The coach classified the example as a SUCCESS if the receiver managed to pass the ball back toward the passer; a FAILURE if one of the defenders cleared the ball to a side; or a MISS if the receiver and the defenders failed to intercept the ball (Figure 13).

The key part of gathering training examples was the passer's recording of the attributes describing the trial. Rather than restricting the number of attributes, we capitalized on the DT's ability to filter out the irrelevant ones. Thus, we gathered a total of 174 attributes (in addition to the coach's label) for each trial, half each from the passer's and the receiver's perspective. The attributes from the receiver's perspective were communicated to the passer before it had to decide which player to pass to. The attributes—**all continuous**—available to the DT were:

- Distance and Angle to the receiver (2);
- Distance and Angle to other teammates (up to 9) sorted by angle from the receiver (18);
- Distance and Angle to opponents (up to 11) sorted by angle from the receiver (22);
- Counts of teammates, opponents, and players within given distances and angles of the receiver (45);
- Distance and Angle from receiver to teammates (up to 10) sorted by distance (20);
- Distance and Angle from receiver to opponents (up to 11) sorted by distance (22);
- Counts of teammates, opponents, and players within given distances and angles of the passer from the receiver's perspective (45);





Sample Decision Tree Output

Receiver 2 : Success with confidence 0.8
Receiver 3 : Failure with confidence 0.6
Receiver 4 : Success with confidence 0.3

Figure 11: During training, the passer chooses its receiver randomly. During testing, it uses a DT to evaluate the likelihood that a pass to each of the teammates would succeed. It passes to the most likely receiver (Receiver 2 in this case).

Figure 12: After choosing its receiver, the passer announces its decision so that the receiver knows to expect the ball and the other teammates can move on to other behaviors. In our experiments, the non-receivers remain stationary.

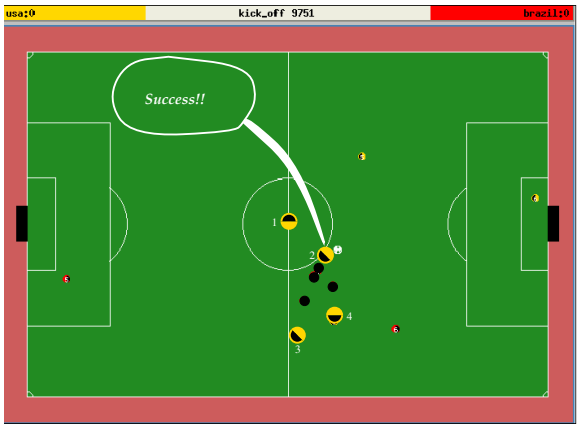


Figure 13: Finally, the receiver records the result of the pass.

Whenever fewer than the maximum number of players were visible, the remaining attributes were marked as *unknown*.

The goal of learning is to use these attributes to predict whether a pass to the given receiver will lead to a SUCCESS, a FAILURE, or a MISS. For training, we used standard off-the-shelf C4.5 code with all of the default parameters [14]. We gathered a total of 5000 training examples, 51% of which were successes, 42% of which were failures, and 7% of which were misses.

Training on this data produced a pruned tree with 87 nodes giving 26% error on the training set. The tree is shown in Figure 14. All of the attributes starting with "passer" are from the passer's perspective. Notice that these are used much more frequently than the attributes from the receiver's perspective. Thus the trained tree is comparably effective when the passer must decide without any input from the potential receivers. The first node in the tree tests for the number of opponents within 6 degrees of the receiver from

the passer's perspective. If there are any, the tree predicts that the pass will fail. Otherwise, the tree moves on to the second node which tests the angle of the first opponent. Since the passer sorts the opponents by angle, this is the *closest* opponent to the receiver in terms of angle from the passer's perspective. If there is no opponent within 13 degrees of the receiver, the tree predicts success. Otherwise it goes on to deeper nodes in the tree.

In order to test the DT's performance we ran 5000 trials with the passer using the DT to choose the receiver. All other behaviors were the same as during training. Since the DT returns a confidence estimate in its classification, the passer can choose the best receiver candidate even if more than one is classified as likely to be successful. If the tree predicts a failure for all three receivers, the one with the lowest confidence reading can be selected. Notice that during testing, the passer *must* pass, while in a game situation the passer would be given the option to dribble or shoot instead.

We compiled results sorted by the DT's confidence in the success of the pass to the chosen receiver (see Table 2). The largest number of passes were classified as successes with confidence between .7 and .8, with another large portion classified as successes with confidence between .8 and .9. Overall, the success rate of 65% is much better than the 51% success rate obtained when a receiver was chosen randomly. However, this result was obtained under a condition of forced passing: the passer was required to pass the ball during all trials. Notice that if the passer wanted to be fairly sure of success, it could pass only when the DT predicted success with confidence greater than .8. The resulting 79% success rate approaches the limit imposed by the success rate of the ball-interception skill. When the testing is repeated with no defenders to intercept the ball, the success rate is 86%.

		Success Confidence:		
Result	Overall	.8–.9	.7–.8	.6–.7
(Number)	(5000)	(1050)	(3485)	(185)
SUCCESS (%)	65	**79**	63	58
FAILURE (%)	26	15	29	31
MISS (%)	8	5	8	10

Table 2: The results of 5000 trials during which the passer used the DT to choose the receiver. Overall results are given as well as a breakdown by the passer's confidence prior to the pass. The passer was forced to pass even if it predicted failures for all 3 teammates. In that case, it passed to the teammate with the lowest likelihood of failure. Results are given in percentages of the number of such cases (shown in parentheses).

With all the different attributes to choose from, it was not obvious how to construct an analytic heuristic for the passer to use when choosing a receiver. However, we needed some comparison other than the passer's random choice during training. A reasonable improvement over the random choice is to pass to the closest teammate. For this reason, we compared the DT decision with the closest teammate heuristic.

Over 5000 trials, the closest teammate heuristic produced a success rate of 64%. Although this number compares favorably with the overall DT success rate, it is significantly lower than the 79% success rate the passer can achieve with the DT when given the option of not passing. Furthermore, the closest teammate heuristic gives no way of estimating the likelihood that a pass will succeed. It simply postulates that given a choice, the passer should pass to the closer teammate. Since the likelihood estimation is the true goal of our learning in this section, there is a clear advantage to using the DT method. When deciding whether to pass, dribble, or shoot, the knowledge of whether or not a given pass is likely to succeed will be extremely useful.

In this section, we demonstrated that a higher-level decision could be built upon the low-level skill learned in the previous section. Using a DT, our clients learned to judge the likelihood that a pass to a given receiver would be successfully received. This judgement represented a second layer in our quest to build intelligent Soccer Server clients by layered learning.

```
C4.5 [release 8] decision tree interpreter Wed Jul 10 11:31:56 1996
-------------------------------------------
Decision Tree:

passer opponents ang6 > 0 : F (1266.0/251.3)
passer opponents ang6 <= 0 :
|   passer opponent1 angle > 13 : S (1054.6/290.5)
|   passer opponent1 angle <= 13 :
|   |   passer teammates ang6 <= 0 :
|   |   |   passer receiver distance <= 22 :
|   |   |   |   passer opponent1 distance <= 20.9 :
|   |   |   |   |   passer players dist8 ang12 <= 3 : S (162.0/72.9)
|   |   |   |   |   passer players dist8 ang12 > 3 : F (15.0/5.8)
|   |   |   |   passer opponent1 distance > 20.9 :
|   |   |   |   |   passer opponent2 distance <= 21.1 :
|   |   |   |   |   |   passer opponents dist12 ang8 <= 1 :
|   |   |   |   |   |   |   passer teammates dist8 ang8 <= 1 :
|   |   |   |   |   |   |   |   passer opponents dist12 ang4 <= 0 :
|   |   |   |   |   |   |   |   |   passer receiver distance <= 20.3 : S (52.2/8.6)
|   |   |   |   |   |   |   |   passer opponents dist12 ang4 > 0 :
|   |   |   |   |   |   |   |   |   passer opponents dist4 ang12 <= 1 : S (60.5/20.0)
|   |   |   |   |   passer opponent2 distance > 21.1 :
|   |   |   |   |   |   receiver teammates dist8 ang12 <= 1 : S (704.3/139.6)
|   |   |   passer receiver distance > 22 :
|   |   |   |   passer opponent1 distance <= 23.1 :
|   |   |   |   |   passer opponents dist12 ang12 <= 0 :
|   |   |   |   |   |   receiver players dist8 ang12 <= 1 : S (87.5/44.7)
|   |   |   |   |   passer opponents dist12 ang12 > 0 :
|   |   |   |   |   |   passer opponents dist12 ang8 > 0 : F (191.0/63.0)
|   |   |   |   |   |   passer opponents dist12 ang8 <= 0 :
|   |   |   |   |   |   |   passer opponents dist4 ang12 > 1 : S (14.0/6.8)
|   |   |   |   |   |   |   passer opponents dist4 ang12 <= 1 :
|   |   |   |   |   |   |   |   passer teammate1 distance <= 19.5 : S (15.0/6.8)
|   |   |   |   |   |   |   |   passer teammate1 distance > 19.5 : F (234.0/91.7)
|   |   |   |   passer opponent1 distance > 23.1 :
|   |   |   |   |   passer opponents dist12 ang12 <= 1 : S (665.9/259.2)
|   |   |   |   |   passer opponents dist12 ang12 > 1 :
|   |   |   |   |   |   passer players dist4 ang12 <= 1 : F (11.0/5.6)
|   |   |   |   |   |   passer players dist4 ang12 > 1 :
|   |   |   |   |   |   |   passer opponents dist4 ang8 <= 0 : S (49.0/19.9)
|   |   |   |   |   |   |   passer opponents dist4 ang8 > 0 :
|   |   |   |   |   |   |   |   passer opponent2 distance <= 23.1 : F (85.0/26.5)
|   |   |   |   |   |   |   |   passer opponent2 distance > 23.1 :
|   |   |   |   |   |   |   |   |   passer opponents dist12 ang12 <= 2 :
|   |   |   |   |   |   |   |   |   |   receiver opponent3 angle <= 48 : F (9.6/2.2)
|   |   |   |   |   |   |   |   |   |   receiver opponent3 angle > 48 : S (108.4/43.0)
|   |   passer teammates ang6 > 0 :
|   |   |   passer teammates ang5 > 0 : F (21.0/7.0)
```

Figure 14: The trained decision tree. Some subtrees with fewer cases covered have been removed for purposes of presentation. Attributes starting "passer" are from the passer's perspective. Attributes starting "receiver" are from the receiver's perspective. For example, "receiver players dist8 ang12" is the number of players that the receiver sees within a distance of 8 and angle of 12 from the passer.

7 Scaling Up to Team-level Strategies

Once able to judge the likelihood that a pass will succeed, a real or simulated soccer player is ready to start making decisions in game-like situations. When considering what to do with the ball, the player can pass to a strategically positioned teammate, dribble, or shoot. To verify that the second level of learning could be incorporated into game-like situations, we implemented a set play that uses the passing decision described in the previous section.

As illustrated in Figure 15, a player starts with the ball in front of it and dribbles towards the opponent's goal. When it notices that there is an opponent in its path, it then decides to stop dribbling and considers its options. Noticing that it is too far away to shoot and that dribbling forward is no longer an option, it decides to pass. Thus, in accordance with the sequence laid out in the previous section, it announces its intention to pass and gets responses from the two nearest players. It then uses the DT to decide which teammate is the more likely to successfully receive the pass.

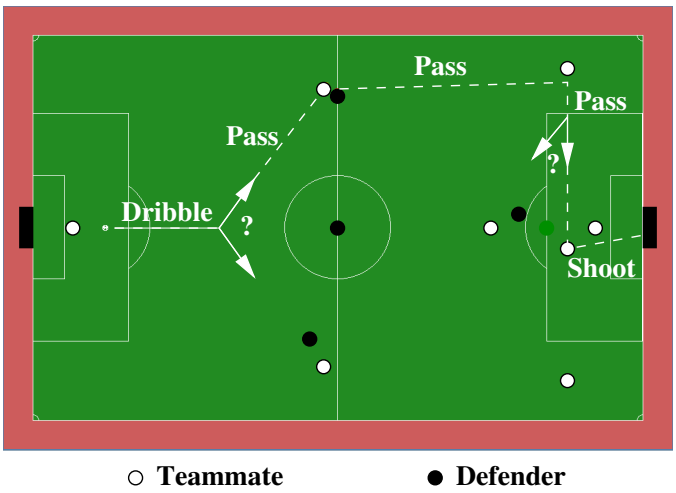


Figure 15: An illustration of the implemented set play. Players are emphasized for improved visibility. Every player uses at least one of the learned skills described earlier in the article.

In Figure 15, the passer chooses the topmost receiver and passes the ball. The receiver and the adjacent defender then both try to intercept the ball using the trained NN ball-interception skill. If the defender gets the ball, it kicks it back towards the left goal and the play starts over. However, if the receiver gets the ball, it immediately kicks the ball to its teammate on the wing. Since the winger is not covered, it can easily collect the ball and begin dribbling towards the goal. Using the same behavior as its teammate that began the set play, the winger notices defenders in its path and decides that it is not at a good angle to shoot. So rather than shooting or dribbling, it uses the trained DT to choose one of the two nearby teammates to pass to. If the chosen receiver is able to get to the ball before the defenders, it immediately shoots towards the goal.

We ran this set play several times in order to verify that the learned behaviors are both robust and reliable. Since the defenders are all equipped with the same ball-interception skill as the receivers, the defenders are sometimes able to break up the play. However, the fact that the attacking team can sometimes successfully string together three passes and a shot on goal when using the learned behaviors demonstrates that these behaviors are appropriate for game-like situations. Furthermore, this implemented set play suggests a number of possibilities for the next layer of learning.

In the set play described above, the player that starts with the ball dribbles until it sees an opponent at a predetermined distance. This distance was chosen so as to allow the player to pass without the

central opponent taking the ball. However, a more flexible and powerful approach would be to allow the dribbling player to learn when to continue dribbling, when to pass, and when to shoot. With these three possibilities as the action space and with appropriate predicates to discretize the state space, TD-lambda and other reinforcement learning methods will be applicable. By keeping track of whether an opponent or a teammate possesses the ball next, a player can propagate reinforcement values for each decision made while it possesses the ball.

Another candidate for the next layer of learned behavior is on the part of the receivers. When it appears that a teammate might be getting ready to pass (or when a teammate directly communicates that it is), a player that might be able to receive the pass could learn to move into a better position. In the current implementation of the set play, as in the DT learning phase, the passer chooses a receiver from among *stationary* teammates. However, if the receivers are given the goal of being chosen as the receiver as often as possible, they can learn a moving behavior that is built upon the learned passing DT. In effect, they will learn to satisfy the preconditions of a successful pass.

Finally, the next level of learning could be the one that introduces adversarial issues. In addition to learning to cooperate with teammates, players can learn to thwart the opponents. For example, the counterpart to the receivers learning to move to receive a pass is to have the defenders learn to move so as to prevent a pass. However, this defensive behavior could potentially become even more complex if the defender is given the goal of actually intercepting a pass, rather than simply preventing it. Then the defender's optimal behavior would be to move to a position such that the passer *thinks* that the pass will succeed, yet such that the defender is still able to intercept the ball.

Once adversarial behaviors are introduced, some additional issues must be considered. First, if the adversaries are allowed to continually adjust to each other, they may evolve increasingly complex behaviors with no net advantage to either side. This potential stumbling block in *competitive coevolution* has been identified and addressed by several researchers who work with genetic algorithms [6, 7, 15]. Second, since a robotic soccer team must be able to play against many different opponents, often for only a single match, it must be able to adapt quickly to opponent behaviors without permanently harming performance against other opponents. We anticipate addressing these and other adversarial learning issues while continuing to build our soccer playing agents.

As we move to higher-level behaviors, we will continue to consider a wide range of learning methods. In addition to NNs and DTs, we are hoping to test TD-lambda and genetic type learning methods. Continuing to build one learned layer at a time, we aim to eventually reach team-level strategies that consider the perceived strategies of opponents.

8 Conclusion

This article describes our initial steps towards creating complete Soccer Server clients using ML techniques. Starting with the ability to intercept a moving ball, we used a NN to teach the client this low-level skill which is a prerequisite for executing more complex behaviors. This individual skill is an example of the most basic form of Multiagent Learning. Although the action is executed by a single agent, it only makes sense in an environment in which other agents exist: without other agents to kick the ball, the client would never have to intercept the ball head-on.

Building upon this individual behavior, we then used a DT to teach the client to evaluate the likelihood that a pass to a particular teammate would succeed. This evaluation represents a more conventional form of Multiagent Learning since several agents (both teammates and opponents) directly affect the passing action.

Finally, we implemented a set play involving several players and several uses of the learned behaviors. This set play verified that the learned behaviors are useful in a game-like situation.

We chose both the low-level skill and the higher-level decision because of their usefulness in building up higher levels of behavior. Just as the ball-interception skill was used by many of the participants in the passing behavior, the knowledge of whether or not a teammate is likely to be able to receive a pass will be useful for clients when deciding whether to pass, dribble, or shoot. As we continue to layer learned behaviors on top of each other, it will be interesting to study how the different learning methods interact with each other. Keeping MAS, and particularly Multiagent Learning, as our research focus, we will continue moving up to higher-level behaviors until we have created a complete team of Soccer Server clients with learned behaviors. At the same time, we will continually apply methods that succeed in simulation to our physical robotic system.

References

[1] AAAI. *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01. Sandip Sen–Chair.

[2] Minoru Asada, Shoichi Noda, and Koh Hosoda. Action-based sensor space categorization for robot learning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS '96)*, pages 1502–1509, 1996.

[3] Minoru Asada, Shoichi Noda, Sukoya Tawaratumida, and Koh Hosoda. Purposive behavior acquisition for a real robot by vision-based reinforcement learning. *Machine Learning*, 23:279–303, 1996.

[4] Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2:14–23, 1986.

[5] Roger Ford, Craig Boutilier, and Keiji Kanazawa. Exploiting natural structure in reinforcement learning: Experience in robot soccer-playing, 1994. Unpublished Manuscript.

[6] John Grefenstette and Robert Daley. Methods for competitive and cooperative co-evolution. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 45–50, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.

[7] Thomas Haynes and Sandip Sen. Evolving behavioral strategies in predators and prey. In Gerhard Weißand Sandip Sen, editors, *Adaptation and Learning in Multiagent Systems*, pages 113–126. Springer Verlag, Berlin, 1996.

[8] Marcus J. Huber and Edmund H. Durfee. Deciding when to commit to action during observation-based coordination. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 163–170, Menlo Park, California, June 1995. AAAI Press.

[9] Hiroaki Kitano, Yasuo Kuniyoshi, Itsuki Noda, Minoru Asada, Hitoshi Matsubara, and Eiichi Osawa. RoboCup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, Spring 1997.

[10] Maja J. Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1), December 1995.

[11] Hitoshi Matsubara, Itsuki Noda, and Kazuo Hiraki. Learning of cooperative actions in multi-agent systems: a case study of pass play in soccer. In *Adaptation, Coevolution and Learning in Multiagent*

Systems: Papers from the 1996 AAAI Spring Symposium, pages 63–67, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.

[12] Itsuki Noda and Hitoshi Matsubara. Soccer server and researches on multi-agent systems. In *Proceedings of the IROS-96 Workshop on RoboCup*, November 1996.

[13] M V Nagendra Prasad, Victor R. Lesser, and Susan E. Lander. Learning organizational roles in a heterogeneous multi-agent system. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 72–77, Menlo Park,CA, March 1996. AAAI Press. AAAI Technical Report SS-96-01.

[14] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[15] Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In Stephanie Forrest, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Mateo,CA, July 1995. Morgan Kaufman.

[16] Michael K. Sahota. Real-time intelligent behaviour in dynamic environments: Soccer-playing robots. Master's thesis, University of British Columbia, 1993.

[17] Michael K. Sahota. Dynasim user guide. http://www.cs.ubc.ca/nest/lci/soccer, January 1996.

[18] Michael K. Sahota, Alan K. Mackworth, Rod A. Barman, and Stewart J. Kingdon. Real-time control of soccer-playing robots using off-board vision: the dynamite testbed. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 3690–3663, 1995.

[19] Peter Stone and Manuela Veloso. Beating a defender in robotic soccer: Memory-based learning of a continuous function. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 896–902, Cambridge, MA, 1996. MIT press.

[20] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. Technical Report CMU-CS-97-193, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, December 1997.

[21] Peter Stone and Manuela Veloso. Towards collaborative and adversarial learning: A case study in robotic soccer. *International Journal of Human-Computer Systems*, 48, 1998.

[22] Peter Stone, Manuela Veloso, and Sorin Achim. Collaboration and learning in robotic soccer. In *Proceedings of the Micro-Robot World Cup Soccer Tournament*, Taejon, Korea, November 1996. IEEE Robotics and Automation Society.

[23] Milind Tambe. Teamwork in real-world, dynamic environments. In *Proceedings of the Second International Conference on Multi-Agent Systems (ICMAS-96)*, Menlo Park, California, December 1996. AAAI Press.

[24] Milind Tambe. Tracking dynamic team activity. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, California, 1996. AAAI Press.

[25] Eiji Uchibe, Minoru Asada, and Koh Hosoda. Behavior coordination for a mobile robot using modular reinforcement learning. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems 1996 (IROS '96)*, pages 1329–1336, 1996.

[26] Gerhard Weiß and Sandip Sen, editors. *Adaptation and Learning in Multiagent Systems*. Springer Verlag, Berlin, 1996.