

# Programming Robosoccer agents by modeling human behavior

Ricardo Aler<sup>a,1</sup>, Jose M. Valls<sup>a,1</sup>, David Camacho<sup>b,\*</sup>, Alberto Lopez<sup>a,1</sup>

<sup>a</sup> Computer Science Department, Universidad Carlos III de Madrid, Avenue Universidad, No. 30, 28911, Leganés, Madrid, Spain

<sup>b</sup> Computer Science Department, Universidad Autónoma de Madrid, C/ Francisco Tomás y Valiente, No. 11, 28049, Madrid, Spain

## Abstract

The Robosoccer simulator is a challenging environment for artificial intelligence, where a human has to program a team of agents and introduce it into a soccer virtual environment. Most usually, Robosoccer agents are programmed by hand. In some cases, agents make use of Machine learning (ML) to adapt and predict the behavior of the opposite team, but the bulk of the agent has been preprogrammed.

The main aim of this paper is to transform Robosoccer into an interactive game and let a human control a Robosoccer agent. Then ML techniques can be used to model his/her behavior from training instances generated during the play. This model will be used later to control a Robosoccer agent, thus imitating the human behavior. We have focused our research on low-level behavior, like looking for the ball, conducting the ball towards the goal, or scoring in the presence of opponent players. Results have shown that indeed, Robosoccer agents can be controlled by programs that model human play.

© 2007 Elsevier Ltd. All rights reserved.

*Keywords:* Learning to play; Imitation; Human modeling; Behavioral cloning; Machine learning; Robosoccer

## 1. Introduction

The Robosoccer simulator is a challenging environment for artificial intelligence, where a human has to program an agent and introduce it into a soccer virtual environment. Programming complex behaviors in software agents is usually a time-consuming and difficult task for human programmers. Machine learning (ML) is becoming a promising way of automatically endowing agents with complex skills. The main approach that has been followed so far is to let the agents learn the behaviors by themselves, totally or partially. For instance, in the case of the Robosoccer simulator, Luke, Hohn, Farris, and Hendler (1997) uses genetic programming to

evolve a complete team of agents whereas Stone and Veloso (1998) proposes an agent architecture where learning can be used at many levels, from acquiring skills to adapting to the opponent.

But ML can be used in another interesting way. Human players can learn to play many games well and quickly, so it makes sense to attempt to imitate them and transfer their experience to computer agents. Learning by imitation and modeling humans is a field common to many research areas like robotics (Kuniyoshi, Inaba, & Inoue, 1994), cognitive science (Brown & Burton, 1978), or user modeling (Webb, Pazzani, & Billsus, 2001). However, only recently imitation techniques have been applied to programming computer agents, specially in games. Sklar, Blair, Funes, and Pollack (1999, 2001) is the first reported work (to our knowledge) where data collected from players is used to train an agent that plays TRON. More recent research has produced quite remarkable human-like behavior in the Quake game (Thurau, Bauckhage, & Sagerer, 2003, 2004a, 2004c).

\* Corresponding author. Tel.: +34 91 4972288; fax: +34 91 4972235.  
E-mail addresses: [aler@inf.uc3m.es](mailto:aler@inf.uc3m.es) (R. Aler), [jvalls@inf.uc3m.es](mailto:jvalls@inf.uc3m.es) (J.M. Valls), [david.camacho@uam.es](mailto:david.camacho@uam.es) (D. Camacho).

<sup>1</sup> Tel.: +34 91 624 9418; fax: +34 91 624 9430.

The work reported in this paper follows this line of research and attempts to imitate a human player with the purpose of creating a Robosoccer player that performs well in the field. The Robosoccer type of domains is interesting for modeling humans, as skills range from low-level reactive behavior to high level strategic actions and team play. Our approach works as follows. First, we created an interface to allow a person to play Robosoccer just like any other video-game. Then, many input/output pairs were generated and recorded after the human player played several matches. The input is what the person can see in the field and the output is the action the person performed under that situation. Then, a ML technique is used to learn the mapping from inputs to outputs. Finally, the ML model is used to control a computer agent.

Imitation can be performed at different levels: reactive, tactical, or strategic. Different levels will raise different issues. As this is the first attempt at using human experience to control an agent in the Robosoccer domain, we have chosen to imitate the human player in low-level actions like running, turning, or kicking the ball. However, results show that learning when to perform such low-level actions allow to learn slightly more complex sequences of actions like conducting the ball to the goal, dribbling opponents, or stealing the ball from opponents.

Results have shown that indeed, agents can be programmed by modeling the experience of humans playing Robosoccer and performing behaviors like looking for the ball, conducting the ball towards the goal, or scoring in the presence of opponent players.

This paper has been divided into the following sections. First, Section 2 deals with work related to modeling other agents, human modeling, imitation, and modeling in games. Section 3 describes our modeling approach in the Robosoccer domain. Section 4 describes how the agent was trained from the instances generated by the human in different types of behavior and discusses the results. Finally, 5 draws the most important conclusions of this work and posits some future lines of research.

## 2. Related work

Currently, there is a lot of interest in automatic modeling of other agents and also of human users/players. In this Section, we will overview related work about agent modeling, considering whether models were reactive or had some form of internal memory, the power of the modeling language (propositional, first order, etc.), the task involved (classification, prediction, imitation, ...), or the domain type (continuous, noisy, ...). We will also focus on aspects related to human modeling.

One of the first attempts at modeling opponents was that of Carmel and Markovitch (1996). Here, the goal was to learn finite automata (DFA) consistent with the behavior of the opponent. The model was used to improve a mini-max search algorithm. Interestingly, models included an internal state. This approach was valid only

for discrete, round-based games, and required complete non-noisy information. Prediction was the goal, but such models could be used to imitate the opponent. Machine learning has been used extensively since then in classical and strategic games. Frnkranz and Kubat (2001) contains a good survey with some discussion of opponent modeling.

Behavioral cloning is an attempt to imitate other agents behavior (Urbancic & Bratko, 1994; Bain & Sammut, 1999). Sammut, Hurst, Kedzierand, and Michie (1992) uses this technique to learn from a human piloting a simulator, from whom input/output traces are obtained. It learns a decision tree for each of the four plane controls. It is a non-deterministic, quickly changing, noisy domain. The complete fly is divided into seven stages, each one with a different goal. Otherwise, similar situations (inputs) in different stages would have very different outputs. Goals are taken into consideration, and it distinguishes between two kinds of goals: homeostatic (non-persistent) and standard (persistent). The authors identify a problem when learning from different persons, because they use very different piloting styles. It differs with our work because the testing domains is different and no information about the stage or the current goal is supplied to the learning system.

KNOMIC (Lent & Laird, 1999) is an approach that learns to imitate an expert from input/output traces in dynamic, non-deterministic, and noisy domains. It uses a rich knowledge representation based on SOAR rules. For instance, actions (operators) are decomposed into three kinds of rules: selection, application, and goal-detection. As in behavioural cloning, two kinds of goals are considered: non-persistent and persistent. The expert is required to decompose a task into operators and sub-operators (a hierarchy of them, actually). This eliminates ambiguity between traces belonging to different parts of the task, which usually aim to achieve different goals (some research that tries to work around this problem, by inducing the agent's subgoals and using them to build the model, is reported in Suc & Bratko (1997)). The expert is also required to annotate the traces by telling the system which operators are selected and unselected. The authors claim that KNOMIC can learn from observing a human in difficult domains such as air combat and Quake II. Successful results are only given from observing a hand-programmed agent. The authors identify a source of ambiguity for learning from humans: they are less systematic, more variable, and make errors. There are many similarities with our approach, but no annotations are obtained from the user in our case.

Bakker and Kuniyoshi (1996) present an overview of imitation in the field of robotics. They define imitation as "imitation takes place when an agent learns a behaviour from observing the execution of that behavior by a teacher" and summarize it as solving the problems of "seeing, understanding, and doing". However, most work is centered around the seeing and replicating sequences of actions (which in robotics is not a trivial task), and less about the understanding

part. Based on Piaget's work (1945), they bring up the important issue that "it is not possible to learn a new behavior unless one almost knows it already".

Kuniyoshi et al. (1994) fits into this framework: a robot observes a human performing a simple assembly task and then reproduces it. Here, the problem that was addressed was to recognize the human actions (in terms of the ones that it already knew). The robot could also adapt to small changes in the position of items on the table. In Hayes and Demiris (1994), a robot follows a teacher through a maze and learns to associate the environment, in terms of local wall positions, with the teacher's actions. In short, "if situation then action" rules are learned. This second study is similar to our input/output rule learning, but the domain used is simpler.

An area where human behavior ML modeling has been the focus is that of user modeling (Webb et al., 2001). Early work in this field was about student modeling, which seeks to model the internal cognition of a student's cognitive system (as in Brown & Burton (1978)). This can be used to make online learning adaptive to the student skills and background knowledge, as well as to predict the student's actions. However, Self (1988) casted some doubt on the tractability of the cognitive approach. Since then, many researchers have followed a different approach that models an agent in terms of the relationships between its inputs and outputs. This approach, called input–output agent modeling (IOAM), treats the operation of the cognitive system as a black box. Some of the systems include feature based modeling (Webb & Kuzmycz, 1996), relational based modeling Kuzmycz (1994), C4.5-IOAM Webb, Chiu, and Kuzmycz (1997), and FFOIL-IOAM Chiu, Webb, and Kuzmycz (1997), among others. Both propositional and relational learning systems have been used. An usual testing ground is the problem of substraction, where models are learned to predict the student response, including mistakes, when doing subtractions. In contrast with the Robosoccer, this is a discrete and static domain.

Currently, the demands of electronic commerce and the Web have led to a fast growth in research in information retrieval, where ML can be used for acquiring models of individual users interacting with information systems (Bloedorn, Mani, & MacMillan, 1996) or grouping them into communities with common interests (Paliouras, Karkaletsis, & Papatheodorou, 1999). These models can help users in selecting useful information from the Web. Although user models are obtained, their domain and purpose is very different to ours. Their aim is to learn user preferences to filter Web information or to build adaptive interfaces. Also, users can be classified into stereotypes, so that user likes or dislikes can be predicted.

ML techniques have also been applied to the prediction of user's actions, also called plan recognition. Kautz and Allen (1986) defined it as the problem of identifying a minimal set of top-level actions that are sufficient to explain the observed actions. Most work learns hierarchical plans from user logs,

and associate them with the goal the user was trying to achieve (Bauer, 1999, 1996). Later, plan libraries can be used to match actual user actions with a plan in the library and determine the user intentions. But their purpose is not to imitate the user, as in our research.

Some research deals with agents modeling opponents and using this knowledge to beat them. In some cases, models are not learned, but predefined and used to classify opponent teams (not individual agents) by means of a similarity metric (Riley & Veloso, 2000a). Models can be used, for instance, to select the best plan to beat the opponent in set plays (Riley & Veloso, 2000a, 2001b). However, although Riley's RectGrid models can classify adversaries, they cannot be used to imitate opponent behavior. Riley's work has focused mostly on the Robosoccer domain, as in our case.

Riley, Veloso, and Kaminka (2002) uses ML for a coach agent in the Robosoccer domain. That is, here learning takes place at a more strategic higher level. The coach can learn from previous games three kinds of models about teams: team formations and passing behaviour. These models can be used to predict and beat the opposite team, or more closely related to our research, so that the team imitates the modeled team. Differences with our research is that whole teams are modeled, and the coach agent is used to observe the playing field.

In Stone (2000), a layered learning architecture is proposed. It is designed to use Machine learning in complex domains, where learning a direct mapping from sensors to actuators is not tractable, and a hierarchical task decomposition is given. Different learning mechanisms are used to learn behaviors from the bottom level (simple behaviors) to the highest level (more complex, strategic behaviors). The architecture is instantiated in the Robosoccer to learn an individual skill (ball interception), a multi-agent behavior (pass evaluation and selection), and adapting the team behavior (here, a new reinforcement learning algorithm called TPOT-RL is used Stone & Veloso (1999)). Although this work does not focus particularly on modeling other agents, it is relevant because if their working hypothesis is correct, it should be expected that learning the detailed models required to imitate other agents should require a layered architecture too (i.e. a direct mapping from inputs to outputs will be almost impossible to learn). This is very likely to be true in general, and should be taken into account in future research, but our work shows that learning input–output mappings yield some positive results. In the opponent-modeling context, Bowling (2003) uses reinforcement learning for multi-agent systems where other agents are also learning. There, the CMUDragons Robocup team is described, that is able to adapt online to an unknown opponent team. Finally, Ledezma, Aler, Sanchis, and Borrajo (2004) proposed a ML scheme to take advantage of prediction of opponents based on visual observations in the Robosoccer simulator. In all these cases, the goal is learning to play and predict/adapt to opponents, but not imitation.

Sklar et al. (1999, 2001) is the first reported work (to our knowledge) where data collected from players playing a dynamic video-game, is used to train an agent. In this case, data was collected from humans playing Tron over the internet and a neural network was trained. Although they managed to create effective controllers for this game, it is less clear that the resulting behavior imitated the humans. They bring up the issue that a person, under the same situation can produce different responses, which can confuse the learning process.

Spronck, Sprinkhuizen-Kuyper, and Postma (2004) uses a new technique called dynamic scripting in role playing games (RPG), to assign weights to rules in a reinforcement learning way. In Spronck, Sprinkhuizen-Kuyper, and Postma (2002) neural networks and genetic algorithms are used to online learning in RPG. In Ponsen and Spronck (2004) the same techniques are applied to real time strategy games.

Recent research has shown a lot of interest in first person shooter games (FPS), like Quake. In this kind of games, human players must react to situations very quickly. So, it is a very suitable environment for learning reactive behaviors. They are played in networks by hundreds of people and records of the best games are kept, so there are good opportunities for Machine learning. Thureau, Bauckhage, and Sagerer (2004a) provide a good summary of how imitation can be used at all levels (reactive, tactical, strategic, and motion modeling ) in FPS games. In Thureau et al. (2003), the authors report some initial research in learning running and aiming behaviors from recorded human games. They built a MATLAB interface to allow a human to play Quake II and record pairs of state-vectors and actions. Then, they used a self-organizing map to reduce the dimensionality of state-vectors and multi-layer neural networks to map state-vectors to actions. Initial results in learning this kind of reactive behavior seem positive. In Bauckhage, Thureau, and Sagerer (2003), neural networks are used to learn trajectories, aiming behavior and their combination. They bring up the important issue of taking into account the context and the past, in addition to the state-vector, to reduce ambiguity when deciding which action to perform. Also in the Quake game, recent research tries to improve imitation models by means of genetic algorithms (Priesterjahn, Kramer, Weimer, & Goebels, 2005).

In Bauckhage and Thureau (2004), tactical knowledge about which weapon to use is learned from human play by means of a mixture of experts. In Thureau, Bauckhage, and Sagerer (2004b), Neural Gas algorithms are used for learning the topology of the environment, and potential fields for learning human trajectories for picking up objects situated at different locations. This is strategic knowledge, because it tells which is the most important place to pick up the next object. The bot got stuck at some locations and temporal changes to the potential field had to be added (pheromone's trails). The author's claim that the bot imitated human behavior in simple setups and showed a mixture of intelligent behavior for more complex missions (i.e.

less imitation but still clever movements). In Thureau, Bauckhage, and Sagerer (2004c), principal component analysis is used to extract primitive movements (building blocks for more complex sequences of movements) and conditional probabilities on the state-vector and the last action are learned. The artificial movements learned seem realistic and even certain human habits are preserved.

### 3. Our modeling approach

#### 3.1. Modelling process

As Fig. 1 shows, a human player interacts with an interface (the GUI soccerclient) that allows to play Robosoccer as a video-game. This GUI sends the human commands to the Soccerserver and displays the state of the field to the human. The interface has been carefully designed so that the only information that is displayed to the user is the one available to the actual agent in the simulated field. The trainer is used to set the playing field in a particular state, because many different states are required to learn general models.

From the GUI, a trace is obtained by observing the human play. Records are obtained for every server cycle. This trace is made of many  $(s, a)$  such records, where  $s$  is the observation made by the agent sensors (distance to the ball, angle to the ball, etc). And  $a$  is the action carried out by the human player in that situation (for instance, kicking the ball, turning, etc.).

Then, Machine learning techniques can be used to obtain a classifier that determines which action has to be carried out in a particular situation. Then, the classifier will be translated into C code, which will be used to control a soccer agent. If the modeling process is correct, the soccer agent will play Robosoccer similarly to the human.

#### 3.2. The Robosoccer interface

In order to build a good model for the soccer agent, the information available to the human through the interface must be as close as possible to that available to the agent. The XClient programmed by Itsuki Noda<sup>2</sup> accomplishes this restriction. It is a first person interface, so the human player observes the objects in the field in 3D perspective. However, the version of the Soccerserver we have used is 2D and it is a bit confusing to observe a 2D world in a first person view. Therefore, we decided to program our own interface, that is displayed in Fig. 2.

This interface displays a complete 2D real time view of the field, just like the soccer monitor. Absolute positions are computed by means of the (Matellan, Borrajo, & Fernandez, 1998) library (trigonometry computations are used to obtain absolute positions of objects from the known position of the banners distributed along the field border).

<sup>2</sup> Xclient is available at [www.sserver.sourceforge.net](http://www.sserver.sourceforge.net)

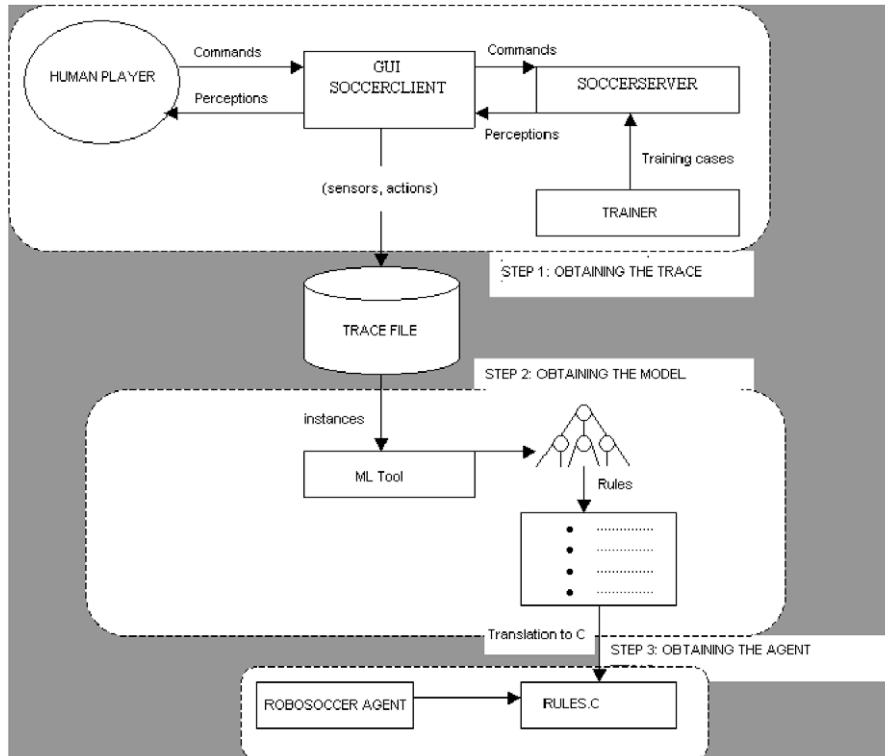


Fig. 1. Process to obtain a model from a person playing Robosoccer by using Machine Learning.

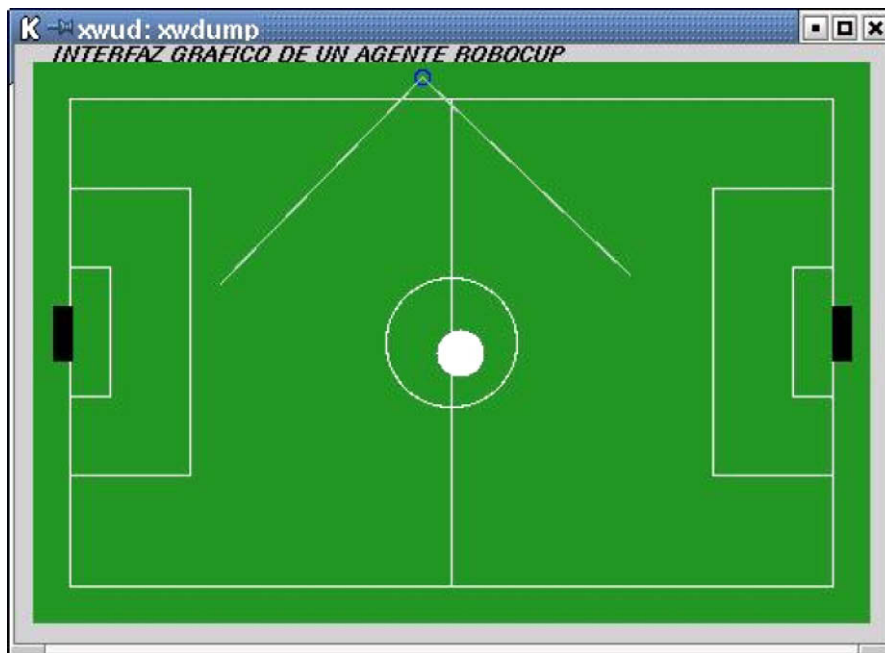


Fig. 2. 2D Interface. Objects are represented by probability circles.

Although the whole field is visible, only those objects within the vision cone of the agent are displayed. Also, in Robosoccer, perception of far away objects is noisy. Thus, objects are displayed as probability circles: the radius of the circle depends on the radius of the object and the distance to the object. In Fig. 2, the ball is represented as a

probability circle. Different colors<sup>3</sup> are used to differentiate the ball, the opponents, and the same team players. Those

<sup>3</sup> For interpretation of color in Fig. 2, the reader is referred to the web version of this article.

objects which are no longer visible are represented at the last position they were seen.

In order to improve playability, not all SoccerServer commands are available to the player. For instance, it is possible to kick the ball with any strength, but the interface only allows a standard kick. The commands allowed by the interface are:

- turn left: the player can turn the agent's body a  $10^\circ$  to the left. The player is only allowed to turn left (but not right) because in some preliminary experiments we found out that it was very difficult for the Machine learning algorithm to discriminate between turning left and right.
- run slow/fast: only these two kinds of dash are allowed. Their power is 60 and 99, respectively. These values were obtained experimentally.
- kick ball: kicks the ball in the direction of the agent's view line with a strength of 60.
- kick to goal: kicks the ball towards the goal, with a strength of 99.

### 3.3. The trainer (coach) agent

In order to have a diverse set of instances for learning behaviors, a diverse set of situations has to be presented to the human player. The trainer agent was used for this purpose. The trainer agent allows to position the agent, the ball, and same-team/opposite team agents in arbitrary positions in the field. Our trainer agent puts objects in random positions in the field according to Fig. 3. In this way, an initially defensive positioning of the opposite team is achieved.

### 3.4. Opposite agents

In the most complex situations, the human player will play within a team against a complete opposite team. In this paper we want to determine whether human input/output modeling of persons works in principle in the Robosoc-

cer domain. To achieve this, we have used a simpler situation where a single human-controlled agent plays against a defensive opposite team (Camacho, Fernández, & Rodelgo, 2006; Fernandez, Gutierrez, & Molina, 2000). This team is based on zones, where each of the team members is located. Among the agents, the player closer to the ball takes the role of leader. The rest of agents maintain a distance from the leader so as to maintain the formation. Agents determine who is the leader and pass this information to others. When the agent moves away from its zone, it tries to pass the ball to other agent, if available. Otherwise, it continues towards the goal, in order to score. The goalie follows a similar behavior: it stays within its zone and looks for the ball. If it is close (15 units), goes for it and kicks it towards the opposite goal.

### 3.5. Model representation

Human behavior can be modeled in many ways. Some of them have been used traditionally in AI: rules, trees, regression models, logic programs, etc. In this paper we intend to study how far first order representations can get. For this paper, we have chosen rules as a way of representing human behavior. They have a long tradition for representing knowledge and their conversion to C if-then-else structures is straightforward. Also, we have used the C4.5 algorithms for generating the rules, although any rule-based ML algorithm would work just as well. Most specifically, the PART algorithm (Revision 8 of C4.5) included in the Weka ML tool has been used (Iain, 2000). The rules follow a `if(situation) then action` structure, where the `situation` checks the values of the agent's sensors and the `action` tells what the agent should do next. Table 1 displays two actual rules obtained in the course of our research.

With respect to the `if` part of the rules, it is very important to select informative attributes so that they capture all information used by the human player to make decisions. Table 2 displays the attributes we have chosen. All positions

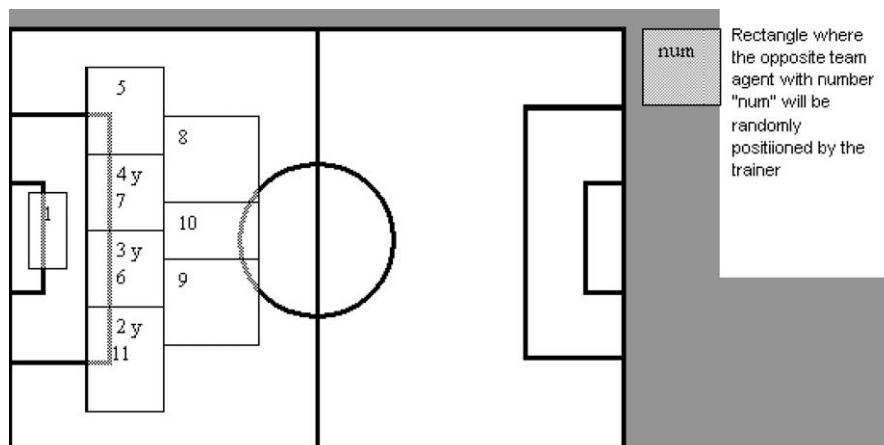


Fig. 3. Rectangles where opposite agents will be randomly positioned by the trainer.

Table 1  
Example of actual rules obtained from Weka's PART algorithm in the Robosoccer domain

---

```

IF (Angle_Ball>-37) AND
  (Distance_Ball>1.2) AND
  (Angle_Ball<=24)
THEN dash99
ELSE IF
  (Angle_Ball>19) AND
  (Angle_Ball<=42) AND
  (X_Ball<=33.9477)
THEN dash99
ELSE turn10

```

---

Table 2  
Attributes used in the left hand side of rules

Attribute	Meaning	Type
<i>X</i> , <i>Y</i>	Absolute agent location	Real
Angle	Angle of the agent's view line	Real
Angle_Ball	Angle between the ball and the agent's view line	Real
Distance_Ball	Distance from the ball to the agent	Real
Distance_Opposite1	Distance from the closest opposite player to the agent	Real
Angle_Opposite1	Angle between the agent and the closest opponent	Real
Valid_Opposite1	It indicates whether the closest opponent could be seen in the last server cycle	Boolean (0,1)
Distance_Opposite2	Distance from the second closest opposite player to the agent	Real
Angle_Opposite2	Angle between the agent and the second closest opponent	Real
Valid_Opposite2	It indicates whether the second closest opponent could be seen in the last server cycle	Boolean (0,1)
Angle_Opponent_goal	Angle between the opponent's goal and the agent	Real
Distance_Opponent_Goal	Distance to the opponent's goal	Real
Valid_Opponent_Goal	It indicates whether the opponent's goal could be seen during the last server cycle	Boolean (0,1)

---

of objects (ball, opponents, and goal) are relative to the view line of the agent and expressed in polar coordinates: distance and angle. If the object is too far away, it cannot be seen and the values of these attributes are meaningless. This is indicated by other attributes, prefixed by `valid` which tell whether the associated object was visible or not. Absolute attributes are only used for the *X* and *Y* coordinates of the agent. In this paper, only the two closest opponents are considered by the rules, although it would be easy to create new attributes so that more opponents can be taken into account.

The values that the right hand side of rules (the action part) can take are: `kick60`, `kick99`, `dash60`, `dash99`, `turn10`, and `turnminus10`. They have already been explained. Currently, the interface can only use these discretized actions, but in the future, it could be modified so that the human player can select a continuous value to turn, to kick, or to dash.

## 4. Training the agent

Some preliminary experiments were carried out for testing simple behaviors like looking for the ball and advancing with the ball in an empty field. As these behaviors were easily learned and properly performed by the agent, we proceeded to more complex behaviors, which involve playing against opponents.

### 4.1. Dribbling static opponents

This skill involves a striker advancing with the ball and scoring, after dribbling static opponent agents located near the goal. These opponents can only kick the ball when it comes close to them. Eleven opponent players are used. The training cases will be generated in such a way that the striker is forced to dribble opponents to find the ball, and then continue dribbling them until it scores.

A first attempt was done with 5370 training instances, obtaining a 95.88% 10-fold cross-validation accuracy. The agent is able to find the ball when there are no opponents and conduct it to the goal. The agent also does well when confronting the 11 opponents. However, in some cases it displays the following flawed behaviors:

- The agent tries to kick the ball when it is too far away, or when the angle is not appropriate.
- When the agent was close to the ball, it turns left again and again.
- The agent collides with an opponent and stops there.

Once the agent performs these flawed behaviors, it never gets out of these states. So for instance, the agent will try to kick the ball forever, or it will get into an eternal turning loop. In general, and this is a property of our reactive approach, if for some reason, the agent performs an action that does not change its environment, or that it changes it but this change is not perceived by the left hand side of the rules, the agent will get stuck in that behavior forever. For instance, when the agent tries to kick the ball, but it is not close enough, nothing has changed in the world, so the agent will repeat its kicking behavior again and again. Similarly, if a chain of rule actions gets the agent to do a complete 360° loop, this will be done forever. Perhaps a mechanism should be added on top of the rules, that realizes when the agent has got into such states and do some random actions until it gets out. However, in this paper we only want to study the pure learning approach, so we will leave that for the future.

In order to improve this behavior, the number of training instances was increased to 14,915, obtaining 172 rules, and a 95.11% accuracy. The behavior improved, but the agent still performs flawed behaviors. These flaws restrain the agent from fulfilling its objective. Our final agent displayed flawed behaviors in 12% of the trials (a trial involves letting loose the agent in the field, finding the ball, and

scoring). We found very hard to improve these results by adding more training instances. In the conclusions section, we will discuss why this is so and propose new lines of research to overcome these limitations.

Thus, results are not perfect but we considered it to be acceptable. Also, these behaviors happen in a world where the only agent that can initiate actions (and change the world) is the striker. When there are more active opponents in the field, the world will change independently of the agent, and the agent will get out of its static states more easily.

#### 4.2. Match with opponents

This is the most complex behavior learned: a striker must get to the ball and score against three defences and one goalie. For this task, it would seem that it would be desirable to choose very difficult opponents, like CMUnited or FC-Portugal, which were previous Robocup champions. However, the human player found impossible to beat them. This was due to these teams playing extremely well, but also to the interface not being responsive enough. This latter problem could not be solved, because the Robosoccer server was not designed with interactive play in mind.

As our aim is to show that human experience can be transferred to soccer agents, we have chosen a challenging but beatable Robosoccer team (Camacho et al., 2006; Fernandez et al., 2000), which has the advantage that although their players have a team behavior, we can use as many players as desired. In this case, only four of them were used. In any case, it must be remarked that, although the (Camacho et al., 2006; Fernandez et al., 2000) team is not a Robocup champion, it is still a very challenging situation, because:

- The opponents outnumber our agent and play cooperatively.
- The opponent can use more actions (turning the neck, turning left and right, kicking and dashing with any power and angle, ...).
- The opposite team has a goalie, whereas our agent must defend and attack.

By confronting a human player against this team, we were able to learn rules that could be transferred to an agent that performed very well, as it will be shown next.

16,124 instances were obtained and 234 rules were created, with a 93.44% cross-validation accuracy. Then, the agent had to play in six new testing matches. Although the agent did not win any of the six testing matches, it scored some goals. Results were: 2–5, 1–6, 0–5, 1–4, 1–5, 0–4 (where the first number indicates goals scored by the agent, and the second one, goals scored by the opponents). The agent incurs in previous flawed behaviors like trying to repeatedly kick the ball when it is not there. However, in this case the world is more dynamic and when an opponent

or the ball comes close, the agent gets out of the loop, and reacts.

In order to improve these results, we increased the number of instances to 24594. 332 rules were generated (93.65%). In this case, we also pruned the rules using WEKA's standard parameters. The number of rules was reduced to 164 (92.65%). Yet, the behavior was greatly enhanced: the agent was able to find the ball on the field, to conduct it towards the goal, to score, to dribble opponents, and to steal the ball from them. The scores in six matches display this improvement, as the agent won one of the games: 5–4, 2–4, 3–4, 4–5, 2–4, 3–4. The agent scored 19 goals versus 25 goals scored by the opponents. The learned classifier was further pruned to 69 rules (91.90%). Similar results were observed in six new matches: 3–4, 2–4, 3–3, 2–5, 3–1, 4–3. The agent scored 17 goals versus 20 goals scored by the opponents. Table 3 summarizes these results.

## 5. Conclusions and future work

In this paper we have applied an input–output modeling approach to model a human playing Robosoccer. First, an interface was built that displayed to the user the objects in the playing field that could be seen according to Robosoccer rules. This interface allowed the user to send low-level commands (dash, turn, and kick) to the Soccerserver. Input/output instances generated by the human player were used by a Machine learning algorithm (PART) to learn a model. This model was then introduced into a computer agent. Results show that in different low-level behaviors, like looking for the ball, conducting the ball to the goal, dribbling opponents, and scoring in the presence of other players, our approach works well. The final agent was able to score many goals against a computer team that the human found challenging. As far as we know, this is the first time that behavioral cloning techniques have been applied in the Robosoccer domain, with positive results. This shows that this is a very promising line of research whose results could be improved further, as discussed below.

Building a user-friendly and responsive interface is of great importance for the human play. Unfortunately, the Soccerserver was not thought as a video-game and it is difficult to construct a responsive enough interface for it. Our current interface is still not as good as commercial video-games. It would be possible to overcome this issue by replicating the functionality of the Soccerserver, but bearing in

Table 3

Summary of results in six testing matches for the “playing with opponents” behavior

Instances	Pruning	Rules	Accuracy	Goals scored	Goals against
16,124	No	234	93.44%	+5	–29
24,594	Yes	332	92.65%	+19	–25
24,594	Yes	69	91.90%	+17	–20

mind interactive play. Rules could be learned from the modified SoccerServer and transferred to agents playing the actual SoccerServer, after, perhaps, some small adaptation. Having a responsive interface is very important for learning low-level behaviors.

We have found out that the agent displayed some flawed behaviors, although not very frequently. This problem could be reduced by increasing the size of the dataset and pruning the model. However, the problem did not disappear completely. We believe that the underlying assumptions of a purely input–output behavioral approach may be the culprit. Our approach works well when the behavior to be cloned is reactive (i.e. the behavior is an input–output map). But if the action of the human depends on hidden variables, in addition to what the human can see on the field, the model of the human will degrade. For instance, the human can make use of memories and predictions about the opponents, even when he is not watching them. But these variables are hidden to the modeling algorithm (i.e. it is not possible to see what the human is thinking). Therefore, our approach worked well because the behaviors to be learned are mostly reactive, but even in this case, there are probably some hidden variables that could help to improve the results.

In the future, we plan to add estimations of some hidden variables to the agent, via new attributes, computed by special purpose algorithms. If human-models are to be used, we should delve more into the cognitive functions applied by a person when playing Robosoccer (like planning, opponent prediction, trajectory computation, ...). These cognitive abilities could be supplied to the agent, and used in the left hand side of the rules via new attributes. For instance, humans use memory to keep track in their minds of opponents and the ball, even when these objects are out of view. In the same way, tracking algorithms could be used to generate attributes that estimate where the ball might be at some particular time.

Imitating humans in the Robosoccer can be done at many levels. Inspired in computer games like FIFA 2006, we intend to let the human use higher level actions like passing the ball, shooting to the goal, pushing the ball, dribbling, etc. In this case, the human player will only have to press a key, and the computer will carry out a pre-programmed behavior (for passing the ball, etc.). Thus, the human can focus on a more strategic level and leave the low-level details to the computer. Modeling can be done at even higher levels, as the team level, or the coach agent. We would also like to test the approach in more complex situations like real matches and real team play.

### Acknowledgements

We would like to thank Vicente Matellan for letting us use his ABC2 routines, and Fernando Fernandez for providing us with his useful Robosoccer team. Agapito Ledezma has been very helpful with his knowledge of Robosoccer.

### References

- Bain, M., & Sammut, C. (1999). A framework for behavioral cloning. In *Machine intelligence agents* (pp. 103–129). Oxford University Press.
- Bakker, P., & Kuniyoshi, Y. (1996). Robot see, robot do: An overview of robot imitation. In *AISB' 96 workshop in robots and animats* (pp. 3–11).
- Bauckhage, C., & Thureau, C. (2004). Towards a fair'n square aimbot – Using mixtures of experts to learn context aware weapon Handling. In *Proceedings of the GAME-ON* (pp. 20–24).
- Bauckhage, C., Thureau, C., & Sagerer, G. (2003). Learning human-like opponent behavior for interactive computer games. In B. Michaelis & G. Krell (Eds.), *Pattern recognition. Lecture notes in computer science* (Vol. 2781, pp. 148–155). Springer-Verlag.
- Bauer, M. (1999). From interaction data to plan libraries: A clustering approach. In *International joint conference on artificial intelligence* (pp. 962–967).
- Bauer, M. (1996). Machine learning for plan recognition. In *Machine learning meets human computer interaction. Workshop of the international conference on machine learning* (pp. 5–16).
- Bloedorn, E., Mani, I., & MacMillan, T. R. (1996). Machine learning of user profiles: Representational issues. In *Thirteen national conference on artificial intelligence* (pp. 433–438).
- Bowling, M. (2003). Multiagent learning in the presence of agents with limitations. PhD thesis, Computer Science Department. Pittsburgh, PA: Carnegie Mellon University, May 2003. Available as technical report CMU-CS-03-118.
- Brown, J. S., & Burton, R. R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, 2, 155–192.
- Camacho, D., Fernández, F., & Rodelgo, M. A. (2006). Roboskeleton: An architecture for coordinating robot soccer agents. *Engineering Applications of Artificial Intelligence*, 19(2), 179–188.
- Carmel, D., & Markovitch, S. (1996). Opponent modeling in multi-agent systems. In *Adaptation and learning in multiagent systems. IJCAI' 95 Workshop. Lecture notes in computer science* (Vol. 1042, pp. 40–52). Springer.
- Chiu, B. C., Webb, G. I., & Kuzmycz, M. (1997). A comparison of first-order and zeroth-order induction for input–output agent modelling. In *Proceedings of the sixth international conference*. Springer.
- Fernandez, F., Gutierrez, G. & Molina, J. M. (2000). Coordinacion global basada en controladores locales reactivos en la robocup. In *Workshop Hispano-Luso de Agentes Fisicos* (pp. 73–85). Tarragona, España.
- Frnkranz, J., & Kubat, M. (Eds.). (2001). *Machines that learn to play games*. Nova Science Publishers.
- Hayes, G., & Demiris, J. (1994). A robot controller using learning by imitation. In *Proceedings of the second international symposium on intelligent robotic systems* (pp. 198–204).
- Kautz, H., & Allen, J. F. (1986). Generalized plan recognition. In *Proceeding of the AAAI national conference on artificial intelligence* (pp. 32–37).
- Kuniyoshi, Y., Inaba, M., & Inoue, H. (1994). Learning by watching: Extracting reusable task knowledge from visual observation of human performance. *IEEE Transaction on Robotics and Automation*, 10(6), 799–822.
- Kuzmycz, M. (1994). A dynamic vocabulary for student modeling. In *Proceedings of the fourth international conference on user modeling* (pp. 185–190).
- Ledezma, A., Aler, R., Sanchis, A. & Borrajo, D. (2004). Predicting opponent actions by observation. In *RoboCup international symposium 2004 (RoboCup2004)*, Lisbon (Portugal).
- Luke, S., Hohn, C., Farris, J. Jackson, G., & Hendler, J. (1997). Co-evolving soccer softbot team coordination with genetic programming. In *Proceedings of the first international workshop on RoboCup, at the international joint conference on artificial intelligence*, Nagoya, Japan.
- Matellan, V., Borrajo, D., & Fernandez, C. (1998). Using ABC2 in the Robocup domain. In *Robocup-97: Robot soccerworld cup I. Lecture notes in artificial intelligence* (pp. 475–483). Springer-Verlag.

- Paliouras, G., Karkaletsis, V., Papatheodorou, C., & Spyropoulos, C. D. (1999). Exploiting learning techniques for the acquisition of user stereotypes and communities. In *Seventh international conference on user modelling* (pp. 169–178).
- Piaget, J. (1945). Play, dreams and imitation in childhood. Heinemann.
- Ponsen, M., & Spronck, P. (2004). Improving adaptive game ai with evolutionary learning. *Computer games: Artificial intelligence, design and education*, 389–396.
- Priesterjahn, S., Kramer, O., Weimer, A., & Goebels, A. (2005). Evolution of reactive rules in multi player computer games based on imitation. In *International conference on natural computation (ICNC' 05)* Changsha, China.
- Riley, P., & Veloso, M. (2000a). On behavior classification in adversarial environments. In *Distributed autonomous robotic systems 4*. Springer-Verlag.
- Riley, P., & Veloso, M. (2001b). Coaching a simulated soccer team by opponent model recognition. In *Proceedings of the agents international conference* (pp. 155–156). ACM Press.
- Riley, P., Veloso, M., & Kaminka, G. (2002). An empirical study of coaching. In *Proceedings of DARS-2002, the seventh international symposium on distributed autonomous robotic systems*.
- Sammur, C., Hurst, S., Kedzierand, D., & Michie, D. (1992). Learning to fly. In D. Sleeman (Ed.), *Proceedings of the ninth international conference on machine learning* (pp. 385–393). Morgan Kaufman.
- Self, J. A. (1988). Bypassing the intractable problem of student modelling. In *Proceedings of the intelligent tutoring systems conference* (pp. 107–123).
- Sklar, E., Blair, A. D., Funes, P., & Pollack, J. (1999). Training intelligent agents using human internet data. In *Proceedings of the first Asia-Pacific conference on intelligent agent technology (IAT-99)* (pp. 354–363).
- Sklar, E., Blair, A. D., & Pollack, J. B. (2001). Training intelligent agents using human data collected on the internet. In *Agent engineering* (pp. 201–226). World Scientific.
- Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E. (2002). Improving opponent intelligence through machine learning. In *Proceedings of the fourteenth Belgium–Netherlands conference on artificial intelligence* (pp. 299–306).
- Spronck, P., Sprinkhuizen-Kuyper, I., & Postma, E. (2004). Online adaptation of computer game opponent ai. *International Journal of Intelligent Games & Simulation (IJIGS)*, 3(1), 45–53.
- Stone, P. (2000). Layered learning in multiagent systems: A winning approach to robotic soccer. MIT Press.
- Stone, P., & Veloso, M. (1998). A layered approach to learning client behaviors in the RoboCup soccer server. *Applied Artificial Intelligence*, 12, 165–188.
- Stone, P., & Veloso, M. (1999). Team-partitioned, opaque-transition reinforcement learning. In M. Asada & H. Kitano (Eds.), *RoboCup-98: Robot soccer world cup II*. Berlin: Springer-Verlag, *Proceedings of the third international conference on autonomous agents*.
- Suc, D., & Bratko, I. (1997). Skill reconstruction as induction of lq controllers with subgoals. In *Proceedings of the 15th international joint conference on artificial intelligence* (Vol. 2, pp. 914–920).
- Thurau, C., Bauckhage, C., & Sagerer, G. (2003). Combining self-organizing maps and multilayer perceptrons to learn bot-behavior for a commercial computer game. In *Proceedings of the GAME-ON* (pp. 119–123).
- Thurau, C., Bauckhage, C., & Sagerer, G. (2004a). Imitation learning at all levels of game-AI. In *Proceedings of the international conference on computer games, artificial intelligence, design and education* (pp. 402–408).
- Thurau, C., Bauckhage, C., & Sagerer, G. (2004b). Learning human-like movement behavior for computer games. In *Proceedings of the eighth international conference on the simulation of adaptive behavior (SAB'04)*.
- Thurau, C., Bauckhage, C., & Sagerer, G. (2004c). *Synthesizing movements for computer game characters. Lecture notes in computer science* (Vol. 3175). Heidelberg, Germany: Springer-Verlag.
- Urbancic, T., & Bratko, I. (1994). Reconstructing human skill with machine learning. In *European conference on artificial intelligence (ECAI 1994)* (pp. 498–502).
- van Lent, M., & Laird, J. (1999). Learning hierarchical performance knowledge by observation. In *Proceedings of the sixteenth international conference on machine learning* (pp. 229–238). Morgan Kaufmann Publishers Inc.
- Webb, G., Pazzani, M., & Billsus, D. (2001). Machine learning for user modeling. *User modeling and user-adapted interaction*, 11(19–20).
- Webb, G. I., Chiu, B. C., & Kuzmycz, M. (1997). Comparative evaluation of alternative induction engines for feature based modelling. *International journal of artificial intelligence in education*, 8, 97–115.
- Webb, G. I., & Kuzmycz, M. (1996). Feature based modelling: A methodology for producing coherent, dynamically changing models of agents' competencies. *User modeling and user-adapted interaction*, 5(2), 117–150.
- Witten, I. H., & Frank, E. (2000). *Data mining: Practical machine learning tools and techniques with java implementations*. Morgan Kaufman.