Predicting Opponent Actions in the RoboSoccer

A. Ledezma, R. Aler, A. Sanchis and D. Borrajo

Departamento de Informática
Universidad Carlos III de Madrid
Avda. de la Universidad, 30, 28911, Leganés (Madrid). Spain
{ledezma, aler, masm}@inf.uc3m.es, dborrajo@ia.uc3m.es

Abstract— **A very important issue in multi-agent systems is that of adaptability to other agents, be it to cooperate or to compete. In competitive domains, the knowledge about the opponent can give any player a clear advantage. In previous work, we acquired models of another agent (the opponent) based only on the observation of its inputs and outputs (its behavior) by formulating the problem as a classification task. In this paper we extend this previous work to the RoboCup domain. However, we have found that models based on a single classifier have bad accuracy, To solve this problem, in this paper we propose to decompose the learning task into two tasks: learning the action name (i.e. kick or dash) and learning the parameter of that action. By using this hierarchical learning approach accuracy results improve, and at worst, the agent can know what action the opponent will carry out, even if there is no high accuracy on the action parameter.**

Keywords—**opponent's model, robosoccer, machine learning**

I. Introduction

In competitive domains, the knowledge about the opponent can give any player a clear advantage. This idea lead us to propose an approach to acquire models of another agent (the opponent) based only on the observation of its inputs and outputs (it's behavior) by solving a classification task [2]. A model of another agent was built as one classifier that would take the same inputs as the opponent and would produce its predicted output. In this paper we present an extension of this previous work to a well known test domain, the RoboCup [6]. Here, models based on only one classifier have poor results, so we have extended this idea with a structure of classifiers.

The behavior of a player in the robosoccer can be understood in terms of its inputs (sensors readings) and outputs (actions). Therefore, we can draw an analogy with a classification task in which each input sensor reading of the player will be represented as an attribute that can have as many values as the corresponding input parameter. Also, we can define a class for each possible output. Therefore, the task of acquiring the opponent model has been translated into a classification task.

In previous papers we have presented results for agents whose outputs are discrete [2], agents with continuous and discrete outputs [9] and an implementation of the acquired model in order to test its accuracy [8]. Here, we use the logs produced by another team's player to predict its actions using a hierarchical learning scheme.

Recent work also related with modeling opponent's model applied to the RoboCup domain can be found in [17], [15], [5].

The remainder of the paper is organized as follows. Section II gives an overview of the Robocup domain. Section III presents a short summary on our learning approach to modeling. Section IV explains the experimental setup. Actual results are detailed in Section V. The paper finishes with some concluding remarks and future works, Section VI.

II. RoboCup

- By the year 2050, develop a team of fully autonomous humanoid robots that can win against the human world soccer champion team -. This is the long-term ambitious challenge of the Robocup. This international initiative fosters the research and education in the artificial intelligence and robotics fields [7]. The main idea is to provide a standard problem in which a wide range of technologies can be applied and mixed. This initiative mainly promotes the development of technologies that use cooperation between agents in dynamic multi-agents environments.

The primary domain used by the Robocup organization as the standard problem is the soccer game and organizes RoboCup: The Robot World Cup Soccer Games and Conferences.

Despite the soccer game is the main underlying testbed, the Robocup also includes a disasters rescue competition. The soccer competition has four categories, simulation, small-size robots, mid-size robots and legged robots. In this work we focus on the simulation version that we will describe next.

The base of the simulation league of RoboCup is the Soccer Server System [11]. This is a server-client software system. The Soccer Server provides a virtual field in which two soccer teams can play a match. The server simulates all movements of the ball and players (clients) and when the clients send a request to execute an action (e.g. turn, run, kick the ball, etc.), the server modifies the current state of the world that it has. Additionally, the server sends periodically information about the environment to each agent (e.g. information about the position of other players, ball, goals, etc.). This information can be noisy and incomplete. The player only receives information about the objects in its vision range.

The team's members are eleven independent players as well as a coach agent. The brain of a each player on the field is a client that controls the actions of the player and communicates with the server through a standard network protocol with well-defined actions.

A standard game lasts for about 10 minutes, with each half being 5 minutes duration. The server work with discrete time intervals known as "cycles". A second has ten cycles, thus, a standard game have 6000 cycles . In each cycle, an agent can send an action to the server, and also receive information about the world from the server every 6 or 7 times per second.

III. Automatic acquisition of models

We will first present a description of a player behavior in terms of its inputs and outputs. If we think in all its possible inputs, we could represent them in terms of a set of input parameters. Therefore, there is a clear analogy with a classification task in which each input parameter of the player will be represented as an attribute that can have as many values as the corresponding input parameter. With respect to the output, since we have selected a set of tasks that have the characteristic of generating one-step decisions (solutions that do not require a set of steps, as in planning), we can think of them as atomic outputs. In terms of a classification task, this allows to define a class for each possible output. Then, the task of modeling (generating a declarative representation of a player behavior) has been translated into a classification task.

Once we have a classification task, any classification technique, c, could be employed for solving the task: instance based learning [1], learning decision trees [12], learning rules([10], [13]), or neural networks [16]. However, we would like to obtain a declarative representation so that it is easier to understand and debug the knowledge obtained. Therefore, we will use a technique that generates symbolic representations, such as decision trees, or rules.

The actual learning task can be described as follows:
• Inputs:
– Set of attributes that models the input parameters (sensors readings) of player p_1 (i.e. from the vision sensor: distance to the ball, ball direction, etc).
– For each attribute, the set of values that its corresponding input parameter can have (i.e. distance to the ball = 0.7 meters, ball direction = -21 degrees).
– Set of possible outputs in the case of discrete classes, and continuous range in the case of continuous classes (i.e. action: kick, dash, turn, etc; kick direction = -180 to 180 degrees).
– Set of training instances T.
• Output: a classifier that provides the same (or approximate) output as player p_1 would provide given the same input instances.

The general framework is described in Figure 1, which shows the interrelation among player p_1, player p_2 that tries to learn and reason about a model of p_1, the classification technique c used for modeling its behavior, and the obtained classifier m (model of p_1). This classifier m should model the behavior of player p_1, in such a way that if one presents the same set of input patterns to both p_1 and m the error between the output provided by p_1 and m should be minimal.

In this case, the model m is a hierarchical combination of several models as we will describe in Section IV.

IV. Experimental Setup

This section describes the experimental sequence to determine whether the knowledge generated by a player p_2 is able to model the behavior of a player p_1, taken as a black box. To do so we have carried out two phases: a player of a good Robocup team plays and we record its actions; and a training phase for obtaining a model m of p_1. Currently, the agent to be modeled has no opponents, so that we can determine whether models can be acquired in the simplest of cases. Its goal is to direct the ball and kick it to score a goal. The player receives environment information through the soccer server in form of aural, vision and body sensors (inputs). The player sends the actions through the soccer server (outputs).

Once the logs of player p_1 have been acquired, the knowledge that tries to model the behavior of p_1 is obtained by two different techniques: a rule modeler (C4.5 [13], discretized outputs) and a regression tree modeler (M5 [14], continuous outputs). The detailed steps for obtaining the opponent model are as follows:
1. The player p_1 plays alone several games. At every instant, the readings of the sensors (inputs) and the actions (outputs) are logged to produce a trace of the player behavior. From this trace it is straightforward to obtain a set of examples T so that another player can learn and model p_1.
2. Let T be the whole set of available examples from p_1 inputs and outputs. Each example $t_i \in T$ is made of two parts: an n-dimensional vector representing the attributes $a(t_i)$ and a value $c(t_i)$ representing the class it belongs to.
3. When the actions $c(t_i)$ in T are a combination of discrete and continuous values (e.i. dash 100), we create a set of instances $\hat{T}$ with only the discrete part of the actions, and a set $\hat{T}_j$ for each parameter of the action using only the examples corresponding to the same action. That is, the name of the action and the parameter of the action will be learned separately. For instance, if the action executed by the player is "dash 100" only *dash* will be part of $\hat{T}$ and the value 100 will be in $\hat{T}_{dash}$ with all the instances whose class is dash.
4. The set $\hat{T}$is used to obtain a model of the action names (i.e. predict the action that the player will execute). We used a rule inducer for this task. The $\hat{T}_j$ are used to generate the continuous values parameters associated to its corresponding action. We used a regression tree inducer for this task. Since the class part of the examples in $\hat{T}$ and $\hat{T}_j$ is generated from the p_1 outputs, the rules model the behavior of p_1.
5. Once all classifiers are built, in order to predict the behavior of the opponent, first the classifier that predicts the action is run in order to know which action should be taken. Second, the associated classifier that predicts the value of the action parameter is executed.

A general description of the hierarchical learning approach is shown in Figure 2.

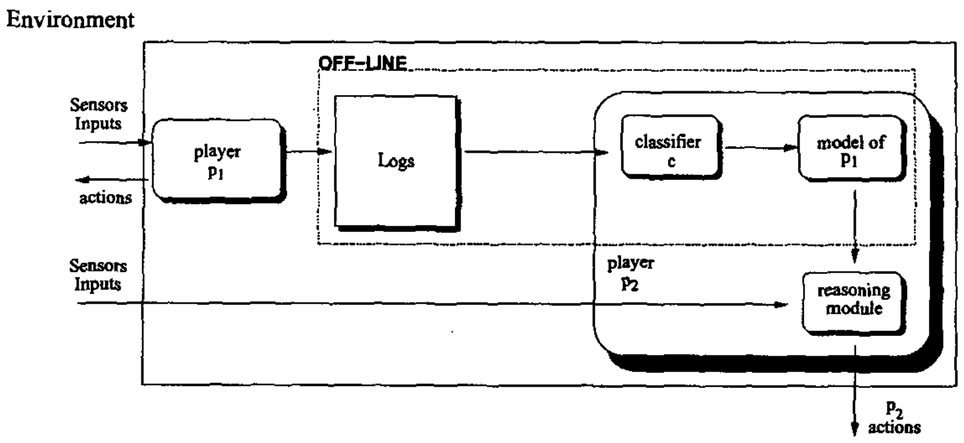

Fig. 1. Architecture of the modeling of opponent behavior.

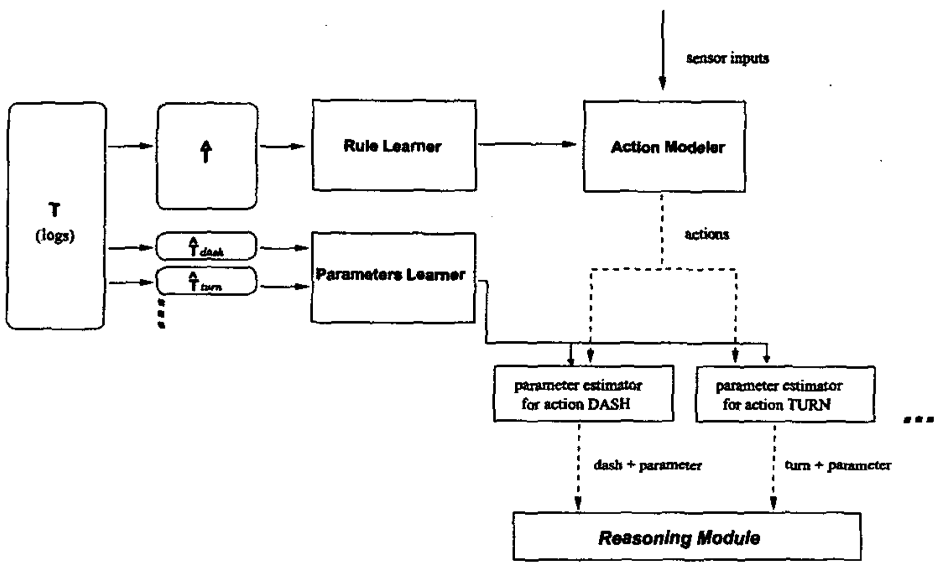

Fig. 2. Hierarchical learning approach.

V. Results

For the experimental evaluation of the approach presented in this paper, the player whose actions will be predicted is a member of TsinghuAeolus [18], the champion team of the 2001 RoboCup edition. The techniques to model this player have been C4.5 [13] and M5 [14]. C4.5 generates rules and M5 generates regression trees. The latter are also rules, whose then-part is a linear combination of the values of the input parameters. In the first series of experiments, we did not use the hierarchical learning scheme described in Section IV and in the second series we used this scheme.

A. Simple Step

As a first approximation to the problem, we saved the logs of one TsinghuAeolus's player in a raw manner to apply a machine learning technique in order to obtain a model. In this case we only saved the logs of a half time of the match (291 instances). We used 140 attributes including the information about the field flags and the ball (vision sensor), and the information from the sense body (stamina, speed, head angle, etc). The class was any combination of the actions that can be executed at the same cycle (e.g. *dash−turn_neck*) with the corresponding numeric value if it has parameters. (e.g. *dash*100−*turn_neck*103.13). These numeric values were obtained discretizing the original values using a variant of the generalized Lloyd algorithm [4]. This is referred as trial 01 in Table I. This configuration obtains more than 45% accuracy of prediction of action to be performed by the opponent plus its associated discretized parameters, which, considering there are 51 classes is not a bad percentage, but we felt we needed to obtain better results. In all the experiments that we have carried out, the accuracy column is a result of a ten-fold cross-validation.

In order to improve the results, we increased the number of instances by increasing the duration of the game. We also added two derived attributes: the X and Y coordinates. In this way, they do not have to be learned from the field flags, and the learning task should become easier. As a consequence, the number of attributes is greatly reduced by eliminating the attributes related to most of the field flags (only the main flags were left). In this trial the number of classes were increased because there are many more instances, and the numeric ranges are wider. We also applied the variant of the generalized Lloyd algorithm to the continuous class for discretizing those classes. Results can be seen in trial 02 of Table I. A 55.72% of accuracy is achieved, even though the number of classes was increased.

In the analysis of the resulting model of the previous experiment, we saw that the attributes with a large number of missing values generated a model which is very difficult to understand. Therefore, we substituted the unknown values of some attributes with large real values for attributes belonging to objects which were out of view (for instance, when a flag is too far to be seen). By doing so, the accuracy remains almost the same (55.57%) but the model makes more sense. Results can be seen in trial 03 of Table I.

We analyzed the resulting confusion matrix from trial 03 and found that there were many classes with very few instances, which made them very difficult to learn. We decided to test what accuracy could be obtained by trying to predict the actions only, without their parameters, therefore reducing greatly the number of classes. For instance, all classes like $dash100 - turn\_neck103.13$ would be compacted to $dash - turn\_neck$, leaving parameter learning for later. The number of classes is reduced now to 7: narrowhigh-turn-turnneck, wide-high, dash, dash-turnneck, kick, turn, and turnneck. Dash-turnneck means that both actions are performed concurrently. Like in the previous trials we did experiments with and without missing values. Classification accuracy improves to 69.66% (trial 04, with missing values) and 72.82% (trial 05, without missing values), as shown in Table I. Those prediction accuracies are reasonable enough for our purposes in such a noisy domain as Robocup is.

B. *Hierarchical learning*

After the first approximation to the problem described in the previous section, we decided to learn the actions and their parameters separately. We also decided to learn only the four main actions (view, dash, kick and turn) because those are the most relevant when predicting the opponent's behavior. In these experiments we used only 32 attributes (main field flags and sensor body information), like in trial 05 of Table I.

TABLE I
FIRST EXPERIMENTS. * MEANS NO MISSING VALUES

Trial	Instances	Attributes	Classes	Accuracy
01	291	140	51	45.36%
02	2595	32	69	55.72%
03*	2595	32	69	55.57%
04	2595	32	7	69.66%
05*	2595	32	7	72.82%

First, we used the complete set of instances to learn a model that can predict the action that the opponent will perform using C4.5. Results can be seen in trial 06 in Table II. A 72.74% accuracy is obtained, similarly to the previous section. Then, for every action, a regression tree was learned by the M5 algorithm, to model the parameters (which are continuous values). These parameters are: turn-angle (TA), dash-power (DP), kick-power (KP) and kick-direction (KD). Results can be seen in trials 07 to 10 in Table II. C means that the class is continuous and CC means that the accuracy is expressed as a Correlation Coefficient.

TABLE II
HIERARCHICAL LEARNING

Trial	Predict	Algorithm	Instances	Classes	Accuracy
06	Main action	C4.5	2594	4	72.74%
07	TA	M5	321	C	0.52CC
08	DP	M5	1331	C	0.94CC
09	KP	M5	929	C	0.83CC
10	KD	M5	929	C	0.58CC
11	KD	C4.5	929	5	62.10%
12	TA	C4.5	321	5	62.30%
13	KD	NBayes	929	5	45.64%
14	TA	NBayes	321	5	48.91%

Results for the Dash-power (DP) and Kick-power (KP) are quite good (0.98CC and 0.83CC, respectively). But for the angle-related parameters Turn-angle (TA) and Kick-direction (KD), the correlation coefficients are worse (0.52% and 0.58%, respectively). For this reason we discretized by hand the continuous values of these parameters and then we used two algorithms (C4.5 AND NAIVE BAYES) for discrete classes. After discretization, 5 classes resulted: center (50°to -50°), center-left (50° to 100°), center-right (-50° to -100°), left (100° to 180°), and right (-100° to -180°). We tried to have a finer discretization at those angles where predicting the opponent's behavior is more useful. For instance, we considered that it is more useful to predict the direction of the movement (or the ball kick direction) when the opponent moves forward than when it goes backward. Results can be seen in trials 11, 12, 13, and 14 (Table II). They are better when using C4.5 (62.10% KD and 62.30% TA) than Naive Bayes (45% KD and 48.91% TA). We also wanted to compare discretized results with previously obtained continuous results. This is difficult, because discretized results given by C4.5 are less precise than continuous values returned by M5, and therefore the predicted behaviour of the opponent is more uncertain. But assuming that predicting the 5 discrete classes (center, center-right, etc) is all we need, then M5 and C4.5 can be compared by discretizing M5's output into the 5 classes and computing the percentage accuracy. To achieve this, we splitted randomly the instances into a training and a test sets (80%/20%). With respect to the Turn-angle, C4.5 obtains 64% whereas M5 gives a 54% accuracy. Similar results are obtained for the Kick-direction: 61% for C4.5 and 44% for M5. So, if the discretized output is all we need, C4.5 is the best option.

VI. Conclusion and Future works

A very important issue in multi-agent systems is that of adaptability to other agents, be it to cooperate or to compete. This can be achieved by learning techniques. In this paper we have studied several issues related to predicting the behavior of opponent's actions in the Robosoccer domain. Several experiments have been carried out to determine how many attributes should be used, how the readability of the model can be improved, and how useful is discretization in this domain. But the most important contribution is to separate the opponent's prediction in two parts. First, we learn what the other agent is going to do, then we learn the numerical parameter of its action. By doing this, the prediction of the opponent's action is increased. This is important because, even if we do not know the strength of the kick, the agent knows at least that it is going to kick, instead of dashing. We have called this schema hierarchical learning. Other learning levels could be set on top of them, like learning the sequence of actions, etc.

In the future, we would like to get better accuracies for the Turn-angle and Kick-direction classes. We observed that the center class includes most of the instances, so misclassifying instances belonging to other classes is less important for the algorithm. This could be solved by carrying out cost-based learning [3], so that classes with few instances have a higher cost when misclassified. Also, although cross-validation results reported here give a good approximation about the quality of the prediction, actual results can only be obtained when the resulting model is used by an agent to predict the opponent's behavior. We plan to do this in the near future.

VII. Acknowledgments

This work was partially supported by a grant from the Ministerio de Ciencia y Tecnología through project TAP1999-0535-C02-02.

References

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, jan 1991.

[2] Ricardo Aler, Daniel Borrajo, Inés Galván, and Agapito Ledezma. Learning models of other agents. In *Proceedings of the Agents-00/ECML-00 Workshop on Learning Agents,*, pages 1–5, Barcelona, Spain, June 2000.

[3] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)*, pages 155–164, 1999.

[4] Fernando Fernández and Daniel Borrajo. VQQL. Applying vector quantization to reinforcement learning. In *RoboCup-99: Robot Soccer World Cup III*, number 1856 in Lecture Notes in Artificial Intelligence, pages 292–303. Springer Verlag, 2000.

[5] G. Kaminka and M. Tambe. Robust multi-agent teams via socially-attentive monitoring. *Artificial Intelligence Research (JAIR)*, 12(105-147), 2000.

[6] H. Kitano, M. Tambe, P. Stone, M. Veloso, S. Coradeschi, E. Osawa, H. Matsubara, I. Noda, and M. Asada. The robocup synthetic agent challenge. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI97)*, pages 24–49, San Francisco, CA, 1997.

[7] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 340–347, New York, 1997. ACM Press.

[8] Agapito Ledezma, Antonio Berlanga, and Ricardo Aler. Automatic symbolic modelling of co-evolutionarily learned robot skills. In José Mira and Alberto Prieto, editors, *Connectionist Models of Neurons, Learning Processes, and Artificial Intelligence (IWANN, 2001*, pages 799–806, 2001.

[9] Agapito Ledezma, Antonio Berlanga, and Ricardo Aler. Extracting knowledge from reactive robot behaviour. In *Proceedings of the Agents-01/Workshop on Learning Agents*, pages 7–12, Montreal, Canada, 2001.

[10] Ryszard S. Michalski. A theory and methodology of inductive learning. *Artificial Intelligence*, 20, 1983.

[11] Itsuki Noda. Soccer server: a simulator of robocup. In Japanese Society for Artificial Intelligence, editor, *Proceedings of AI symposium'95*, pages 29–34, 1995.

[12] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

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

[14] J. Ross Quinlan. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 236–243, Amherst, MA, June 1993. Morgan Kaufmann.

[15] Patrick Riley and Manuela Veloso. Recognizing probabilistic opponent movement models. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer Verlag, Berlin, 2002. (extended abstract).

[16] D.E. Rummelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing Foundations*. The MIT Press, Cambridge, MA, 1986.

[17] Peter Stone, Patrick Riley, and Manuela Veloso. Defining and using ideal teammate and opponent agent models. In *Proceedings of the Twelfth Innovative Applications of AI Conference*, 2000.

[18] Cai Yunpeng, Chen Jiang, Yao Jinyi, and Li Shi. Global planning from local perspective: An implementation of observation-based plan coordination in robocup simulation games. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup-2001: The Fifth RoboCup Competitions and Conferences*. Springer-Verlag, 2002.