Opponent Provocation and Behavior Classification: A Machine Learning Approach

Ramin Fathzadeh[1], Vahid Mokhtari[1], and Mohammad Reza Kangavari[2]

[1] Mechatronics Research Laboratory
Department of Computer Engineering, Islamic Azad University,
Qazvin Branch, Qazvin, Iran
{fathzadeh,mokhtari}@qazviniau.ac.ir
http://www.mrl.ir/
[2] Department of Computer Engineering, Iran University of
Science and Technology, Tehran, Iran
kangavari@iust.ac.ir

Abstract. Opponent Modeling is one of the most attractive and practical arenas in Multi Agent System (MAS) for predicting and identifying the future behaviors of opponent. This paper introduces a novel approach using rule based expert system towards opponent modeling in RoboCup Soccer Coach Simulation. In this scene, an autonomous coach agent is able to identify the patterns of the opponent by analyzing the opponent's past games and advising own players. For this purpose, the main goal of our research comprises two complementary parts: (a) developing a 3-tier learning architecture for classifying opponent behaviors. To achieve this objective, sequential events of the game are identified using environmental data. Then the patterns of the opponent are predicted using statistical calculations. Eventually, by comparing the opponent patterns with the rest of team's behavior, a model of the opponent is constructed. (b) designing a rule based expert system containing provocation strategies to expedite detection of opponent patterns. These items mentioned are used by coach, to model the opponent and generate an appropriate strategy to play against the opponent. This structure is tested in RoboCup Soccer Coach Simulation and MRLCoach was the champion at RoboCup 2006 in Germany.

1 Introduction

Multi Agent System is one of the sub-disciplines of artificial intelligence which was introduced for the purpose of defining the rules and principles for developing complex systems and provides a mechanism for cooperating the agents [1], [2]. In real-time environments, multi agent systems need agents that are able to act automatically as members of a team. Modeling opponent in our multi agent system environment predicts the future behaviors of opponent and proposes an appropriate counteraction [3]. RoboCup is an MAS environment and opponent modeling plays a crucial role in this context. In this domain, every team is defined as a group of autonomous agents which are connected to a server and play a simulated soccer [4]. Coach agent of the team, receives the complete and noiseless information from the field and in order to enhance the performance sends messages in format of the standard coach language, called CLang, to its players [5], [6].

U. Visser et al. (Eds.): RoboCup 2007, LNAI 5001, pp. 540–547, 2008.

Recently emphasizing on opponent modeling, coach competition has regulation changed been, so that coach becomes in charge of identifying the weaknesses and strengths of the opponent *(patterns)*, from other behaviors of the opponent *(base strategy)*. The 2006 coach competition rule defines pattern and base strategy as:

Pattern: A *simple behavior* that a team performs which is *predictable* and *exploitable* for the coaches.

Base strategy: The general strategy of the test team regardless of the pattern in it.

To exemplify this, a pattern may be a sequence of consecutive passes between some particular players, clearing the ball to the outside of penalty area by defenders, or any different formation of players between pattern and base strategy

Our work is focused on opponent modeling and online pattern identification. For this purpose, MRLCoach receives the previous plays of the opponent as two log files of pattern and base strategy, and by analyzing them identifies the events occurred such as pass, shoot, dribble, etc. Then for pattern recognition, *chi-square test* [7], [8], issued to analyze the possible relation between an event and a sequence of previously occurred events. The eventual model of the opponent could be a collection of multiple identified patterns. Now, using a *radix tree* [9], we compare constructed models from the pattern and base strategy log files and store the differences as the final model of the opponent in the *model repository*. Finally, coach makes models for each of the pattern and base strategy log files. *Opponent Provocation* could be considered as a new problem in MAS environment. In RoboCup coach domain, this goal means for each identified behavior of opponent, a strategy is constructed to activate this behavior in online game. Hence *Rule Based Expert System* is recruited to expose the appropriate strategies for opponent provocation. Thereupon, in online mode, by observing the opponent behavior, coach looks for an appropriate strategy for it in *Knowledge Base*. Once an instance is found, it is sent to the players. In online mode, observing the live game, coach exposes an online model of the opponent and compares it with the stored models in repository. When a matching online model is identified, coach reports it as the current opponent model to the server. The remainder of this paper is organized as follows: At first, we introduce the soccer server environment, second, a 3-tier learning architecture for predicting and exposing the opponent behavior is presented. Afterwards, we explain how rule based expert system proposes a proper strategy against the opponent team and how the process of learning is accomplished in online game. Continuing on, section 4 presents the results of our experiments in details. The final section of this paper is devoted to the future works.

2 The Environment

The RoboCup simulation league uses the Soccer Server System [4] to simulate the field and the players. Each player has to be a unique process that connects via a standard network protocol to the server. The players receive video and audio information every 150 msec over the network and can issue primitive actions like kick, dash, turn, turn-neck and say every 100 msec. The server processes the actions of the players and generates new visual information. The rest of information consists of the distances and angles to other players, the ball and landmarks. The players can only perceive objects that are in their field of vision and both the visual information and the execution of the actions are noisy. Additionally, the accuracy and amount of sensory information decreases with

the distance of objects. Communication between the clients is only allowed if it passes via the server, considering the fact that bandwidth and hear range are limited.

An extra client on each team can connect as a coach, who can see the whole field and send strategic information to clients when the play is stopped, for example for a free-kick. In the soccer server, a coach agent has three main advantages over a standard player. First, a coach has given a noise-free omniscient view of the field at all times. Second, the coach is not required to execute actions in every simulator cycle and can, therefore, allocate more resources to high-level considerations. Third, in competition, the coach has access to log-files of past games played by the opponent, which can provide to important strategic insights.

3 Coach Framework

Coach agent behavior comprises of two phases: In *Opponent Behavior Acquisition* phase, raw data is received from the environment, and events of the game are identified using statistical calculation. Then the opponent behaviors are classified as patterns. In *Opponent Provocation* phase, pattern recognition process is expedited. Here a rule based expert system is being used to opponent provocation. Figure 1 shows the general architecture of coach.

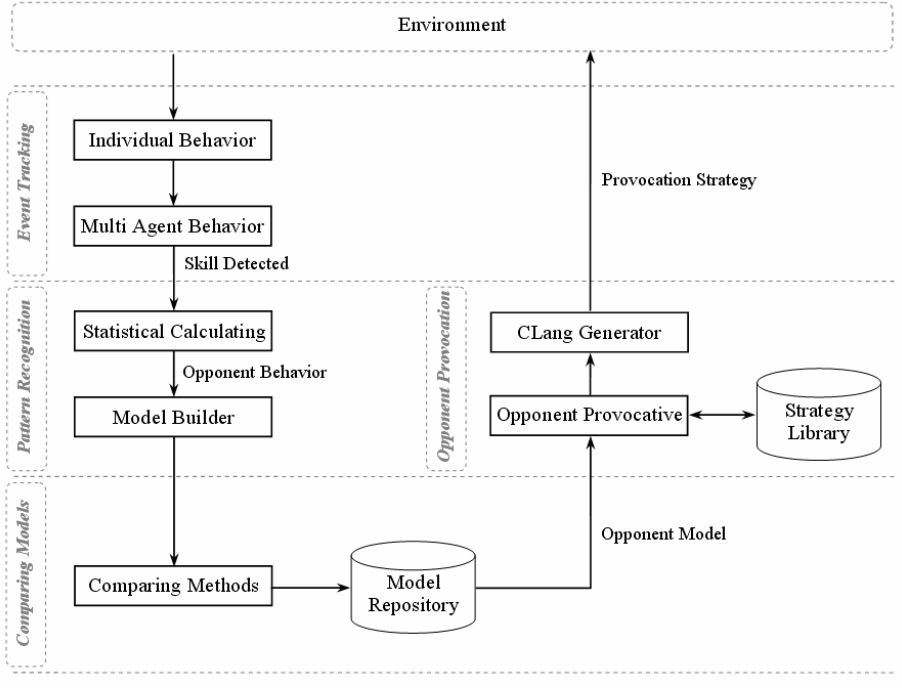

Fig. 1. MRLCoach architecture: event tracking, pattern recognition, comparing models and opponent provocation

3.1 Opponent Behavior Acquisition

Before starting the game, coach is provided with a set of prepared patterns and base strategies of the opponent's past games. By analyzing this information, coach exposes the occurred events. Then, using the chi-square test, expected patterns in a log file are identified. Constructed model of the opponent is a set of these patterns. This model is compared with the model created from base strategy. Their difference is stored as a final model of the opponent for online use. During the game, online coach receives the match's information and analyzes play of the opponent with similar methods used in offline phase and compares the created model with those already existing in the repository and reports a matching one as the current opponent model to the server.

The main goal of coach is to mine the opponent behavior. For this purpose, we classify the possible behaviors in a simulated match to different classes such as formation, pass, shoot, dribble, hold, etc., some of which are divided into subclasses. For instance, the pass class has the following 3 subclasses: *direct pass*, *pass graph* and *closed pass graph* [10], [11].

The opponent modeling process is comprised by *event tracking*, *pattern recognition* and *comparing models*.

Event Tracking. The first step in modeling the opponent is to detect the events occurred in a game. Event tracking consists of breaking problem down into two individual and multi agent behaviors [12]. For tracking these behaviors, raw data including play mode, positions and velocities of both the players and the ball are gathered from the field. Then by identifying the ball owner in every cycle, the individual behaviors, namely pass, shoot, dribble, hold and intercept are exploited. After identification of the individual behaviors, in a next higher level, multi agent behaviors such as formation, defending system, offending system and pass graphs are recognized.

Pattern Recognition. A pattern is a sequence of events appeared in a game sufficient number of times, which is predictable and exploitable. In order to treat the sequence of events identified at the previous section as patterns, we have recruited the chi-square test:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

Which O_i is the observed frequency of an event, E_i is the expected frequency of an event and k is the number of random variables.

As an example for recognition of pass pattern, 29 observed passes for the player 2 are shown in the frequency table 1.

Table 1. Use of chi-square test in recognition of pass pattern

	Player 3	Player 4	Player 5	Total
Observed Pass (O)	4	9	16	29
Expected Pass (E)	2.9	5.8	20.3	29
$(O-E)^2 / E$	0.417	1.765	0.910	3.092

The columns of this table contain players receiving the pass. The first and the second rows have respectively the numbers of observed and expected passes. Based on our experience in recognition of pass pattern, we consider the expected frequency to be at least 70% of the total of observed passes to the player with maximum number of receives. The total in the third row is the calculated chi-square value. Now, we should compare the chi-square value we calculated, χ^2=3.092, with the χ^2 value read from the *table of* χ^2 [13], with n-1 *degrees of freedom* (where n is the number of *categories* which is the number of pass receiver players, 3 in our case). So we have only 2 degrees of freedom. From the χ^2 table, we find the critical value of 5.99 with *probability*=0.05.

Because the calculated value of 3.092 is less than 5.99, our assumption is though acceptable. This means in 95% of the cases the calculation of pass pattern is significance.

Comparing Models. A model of the opponent is a set of detected patterns. We store this model using *radix tree ADT*. For each of the pattern or base strategy log files, a radix tree is created. Afterwards, it is necessary to compare these radix trees to suggest the final opponent model. The difference between *pattern radix tree* and *base radix tree* determines the final model of the opponent that is stored in a third radix tree in model repository. Actually each node of this tree is a node existing in pattern radix tree but not in base radix tree.

After all the possible models of the opponent are exposed from the log files in the offline section, in the online mode, we are to identify the current model of the opponent in real-time. To accomplish this, in each game, by receiving the information from the field, events and patterns of the opponent are identified with similar methods used in offline; then an online model of the opponent is created. Unlike the offline mode where model of the opponent is identified by comparing the log files of pattern and base strategy, in online mode, a current model is compared to a collection of previous models of the opponent. Hence, some similarities or conflicts between patterns are possible. To preclude erroneous reports, our policy is to store the similar or conflicting patterns in a specific table. To deal with these similarities or conflicts, we carry on computations until they are distinguished. In this case, if these conflicts are not settled until the end of the game, reporting is not allowed.

3.2 Opponent Provocation

One of the other policies applied in online section is the selection of a suitable strategy to motivate the opponent players to disclose the expected patterns. For this purpose, we have used rule based expert system architecture to provide a provoking strategy for opponent players. Rule-based systems are computer systems that use rules to provide recommendations or diagnoses, or to determine a course of action in a particular situation or to solve a particular problem [14], [15]. To design such rule-based architecture, the patterns identified in offline mode are considered as antecedents and the consequents are strategies for opponent provocation which are built with the assistance of a human expert. We store these ordered pairs of patterns and strategies as rules in our *Strategy Library*. To present an appropriate strategy, the *Forward Chaining* method is used. In a way that by receiving observations from the environment, we search in strategy library for a rule whose condition part is identical

to these observations. In this case, this rule is triggered and its action part is sent to the players as provoking strategy. Therein one of the outstanding problems of this system is *Conflict Resolution*. This means that it is probable that more than one rule are qualified to be fired. There are several conflict resolution strategies, such as choosing the most recent activated, the least frequently triggered, etc. The most suitable conflict resolution strategy is priority-based: assigning a priority value to every rule and selecte the fired rule with the highest priority. In the case that several rules have the highest priority value, a random selection is performed. To deal with this problem we have benefited from prority-baseed conflict resolution.

With this method, coach can select the best possible strategy to provoke opponent players. This increases the accuracy of reporting pattern and speeds up pattern recognition. Figure 2 illustrates some used provocation strategies.

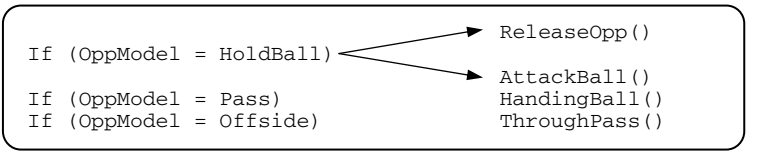

Fig. 2. Example of provocation strategies

What we have mentioned here about opponent provocation could be considered as a novel approach in Opponent Modeling.

Two examples are given here to clarify the idea.

- Let's assume that the opponent pattern is offside trap. In this case, we should put our players in offside situation. Therefore the candidate strategies which have the properties and can be used to activate opponent behavior could be "move forward" or "through pass". Meanwhile the game, by observing the first occurrence of offside trap, if the offside fact is found in strategy library, "move forward" or "through pass" strategy is activated.
- Let the opponent behavior be a simple direct pass e.g. a pass from player 9 to player 10. For this case, our strategy is "handing ball to player 9 of opponent".

Eventually these strategies are advised to players in the format of standard coach language. This structure is completely implemented and tested at the RoboCup competitions. In the following section, our experiments are explained in full detail.

4 Experimental Result

The MRL team acquired the 1st place among 10 participated teams in RoboCup 2006 competition. This competition had 3 rounds, each consisting of 4 iterations. In every round, nearly 15 to 20 patterns are fed to the log analyzer. The participants are responsible for creating these patterns. According to coach regulation, teams should provide at least 3 patterns for each round and active patterns for the iterations are also selected by them. Log analyzer has an average of 5 minutes to process a pattern.

Meanwhile the game, online coach should identify and report the activated patterns within 10 minutes. The score of a team depends on both the number of correct detected patterns and the time of report. In the first round, our team placed second. After making some slight modification to the parameters in the algorithms used to reduce the noise, we attained the first place in the second round and could pass to the final round without any wrong reports. Although most of the patterns in the final round were chosen from the patterns which other teams had prepared, our team took the greatest score. MRL detected 10 correct patterns from 18 activated patterns that equal the sum of all identified patterns of other teams in this round. The final round results and the ranks of teams are depicted in table 2.

Table 2. Total scores in the final round of the competition for the top 4 finishers

Team	Iteration 1 Score	Rank1	Iteration 2 Score	Rank2	Iteration 3 Score	Rank3	Iteration 4 Score	Rank4	Final Rank
MRL	67583.59	1	40857.6	1	51311.25	1	91578.75	1	4
UT Austin	44649.99	2	-4739.40	4	9548.5	3	9543.75	2	11
Caspian	11200.0	3	-4000.0	3	15000.0	2	-8000.0	4	12
Pasargad	-9282.0	4	2475.199	2	-9282.0	4	5418.0	3	13

The results of RoboCup 2006 competitions showed that MRL had well-deserved victory for being champion. And despite of lots of similarities and conflicts between the patterns, we had the least wrong reports number among all the teams, in the way that in the final round from a total of 26 wrong reports just one of them was ours.

5 Conclusion and Future Work

In this paper, we presented a novel architecture for modeling the opponent in coach competition. MRLCoach is an agent that is fully implemented and has been successfully tested in RoboCup competition. Providing this learning structure, MRL team took the 1st place in RoboCup 2006 coach competition. Our unparalleled performance in the competition has convinced us that the recipe for our success had been our capability in the handling of the noises and conflicts. Pattern categorization and noise handling are of prominent factors in our success in the competition. The trick in advising our players to motivate the opponent players to demonstrate the patterns had also assisted us in identifying the opponent behaviors simpler and sooner.

Opponent provocation could be considered as a novel approach in MAS for achieving a specific goal. This method can be applied to arenas such as military applications and other adversarial domains in order to give strategies to provoke and trap the enemy. In the future, our study will be focused on optimization of opponent provocation system to expedite opponent modeling and make this process more accurate.

References

1. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice-Hall, Inc. (1995)
2. Weiss, G.: Multiagent Systems a Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
3. Riley, P.: Coaching: Learning and Using Environment and Agent Models for Advice., Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891 (2005)
4. Murray, J., Noda, I., Obst, O., Riley, P., Stiffens, T., Wang, Y., Yin, X.: RoboCup Soccer Server User Manual for Soccer Server 7.07 and later (2002)
5. Riley, P., Veloso, M.: An Overview of Coaching with Limitations. In: Proceedings of the Second Autonomous Agents and Multi-Agent Systems Conference, pp. 1110–1111 (2003)
6. Riley, P., Veloso, M.: Coaching Advice and Adaptation. In: Polani, D., Bonarini, A., Browning, B., Yoshida, K. (eds.) RoboCup 2003: The Sixth RoboCup Competitions and Conferences. Springer, Berlin (2004)
7. Rohlf, F.J., Sokal, R.R.: Statistical Tables, 3rd edn. W. H. Freeman and Company, New York (1995)
8. Banks, J., Carson, J.S.: Discrete-Event System Simulation. Prentice-Hall Inc. (1984)
9. Sedgewick, R.: Algorithms. Addison-Wesley (1983)
10. Fathzadeh, R., Mokhtari, V., Shahri, A.M.: Coaching With Expert System towards RoboCup Soccer Coach Simulation. In: Bredenfeld, A., Jacoff, A., Noda, I., Takahashi, Y. (eds.) RoboCup 2005. LNCS (LNAI), vol. 4020. Springer, Heidelberg (2006)
11. Kuhlmann, G., Stone, P., Lallinger, J.: The Champion UT Austin Villa 2003 Simulator Online Coach Team. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003: Robot Soccer World Cup VII. Springer, Berlin (2004)
12. Stone, P.: Layered Learning in Multi-Agent Systems., Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891 (1998)
13. Shannon, R.E.: Systems Simulation: The art and science, p. 372. Prentice Hall, Englewood Cliffs (1975)
14. Buchanan, B.G., Shortliffe, E.H.: Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison Wesley (1984)
15. Biondo, S.J.: Fundamentals of Expert System Technology: Principles and Concepts, Intellect (1990)