Proc. The Fifteenth International Conference on Industrial & Engineering Application of Artificial Intelligence & Expert Systems (IEA/AIE 2002), Cairns, Australia, Jun. 2002, published in Lecture Notes in Computer Science, T. Hendtlass and M. Ali, Eds., vol. 2358, pp. 136-145.

Learning from Human Decision-Making Behaviors - An Application to RoboCup Software Agents

Ruck Thawonmas[1], Junichiro Hirayama[2], and Fumiaki Takeda[2]

[1] Department of Computer Science, Ritsumeikan University
1-1-1 Noji Higashi Kusatsu City, Shiga 525-8577, Japan
[2] Course of Information Systems Engineering, Kochi University of Technology
185 Miyanokuchi, Tosayamada-cho, Kami-gun, Kochi 782-8502, Japan
takeda.fumiaki@kochi-tech.ac.jp

Abstract. Programming of software agents is a difficult task. As a result, online learning techniques have been used in order to make software agents automatically learn to decide proper condition-action rules from their experiences. However, for complicated problems this approach requires a large amount of time and might not guarantee the optimality of rules. In this paper, we discuss our study to apply decision-making behaviors of humans to software agents, when both of them are present in the same environment. We aim at implementation of human instincts or sophisticated actions that can not be easily achieved by conventional multiagent learning techniques. We use RoboCup simulation as an experimenting environment and validate the effectiveness of our approach under this environment.

1 Introduction

It's a not-so-easy task to build software agents or multiagents (henceforth simply called agents) so that they could perform desired actions. To define proper condition-action rules, one may choose to hand code them using if-then rules or to use online learning techniques such as reinforcement learning [1]. Hand coding requires a set of rules that must cope with various kinds of conditions. This approach requires special skills and much effort to write complete agent programs. In addition, when conditions and desired actions are complicated, rules will also become intricate and the number of rules large. On the other hand, reinforcement learning can relatively easily make agents learn from their experiences a moderate number of condition-action rules. Compared to hand coding, this approach imposes the lesser burden on the user. However, in this approach it might be difficult to learn complex rules or take long learning time to reach them. Rather, this approach is useful for adding refinement to other methods.

The main objective of our study is to solve the aforementioned problems. In our approach, we derive condition-action rules from decision-making behaviors of humans. Namely, logs are taken that consist of conditions and the corresponding

actions while the latter are decided by a human who is present in the same environment as the agent. Condition-action rules extracted from these logs using C4.5 [2] are then applied to the agent. Thereby, the agent can be promptly trained to have human decision-making behaviors. Moreover, this approach has high potential to allow implementation of complex condition-action rules such as cooperative behaviors among agents or human instincts that can not be readily achieved by conventional agent learning techniques.

In this paper, we use RoboCup soccer simulation [3] as our agent platform. A system called OZ-RP [4] was recently proposed and developed by Nishino, et al., that enables human players to participate soccer games with agents. However, this system highly depends on the model for describing the environment and internal state of their agents. It thus is not compatible with our agents previously developed for both education [5] and competitions[1]. As a result, for our agents we originally develop a system called KUT-RP that has similar functions to the OZ-RP. Logs from the KUT-RP system are used for extracting human decision-making behaviors.

2 RoboCup Simulation

RoboCup[2]is an international initiative to foster artificial intelligence and intelligent robotics by providing a standard test bed (soccer) where several technologies can be integrated and examined. The grand challenge of the RoboCup initiative is to develop a team of fully autonomous humanoid robots that can win against the human world soccer champions by the year 2050. Research outcomes are to be applied to a wide range of fields such as rescuing robots, intelligent transportation systems, and other social infrastructures. At present, RoboCup consists of five leagues, namely, simulation, small size, middle size, Sony 4 legged, and humanoid leagues.

Fig. 1 shows the architecture of the simulation league system adopted in this study. They are three main components in the system, i.e., soccer clients (agents), the soccer server, and the soccer monitor. The soccer server and clients communicate with each others during a game based on a client/server model under the UDP/IP protocol. All soccer objects such as the soccer ball, the players, or the field are visualized on the soccer monitor.

In simulation, to maximally mimic real-world soccer games, a number of constraints have been introduced and applied to the system. For example, noises are intentionally imposed to sensory information (aural sensor, vision sensor, and body sensor) to be sent from the soccer server to each agent. On the other hand, three basic commands (kick, dash, and turn) to be sent from each agent to the soccer server always contain a certain degree of inaccuracy. The vision scope and stamina of each agent is also limited.

[1] Our soccer simulation teams NoHoHoN and NoHoHoN G2 participated in the Japan Open 2000 and 2001, respectively.

[2] The official web site is www.robocup.org.

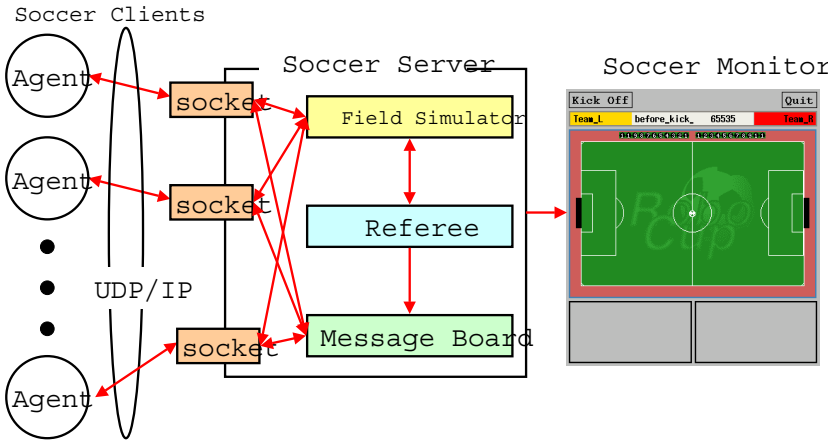

Fig. 1. Architecture of the simulation league system.

3 Architecture of KUT-RP

In this section, we describe the architecture of our originally developed KUT-RP (Kochi University of Technology - Ritsumeikan University's Player) system using Java. The KUT-RP system is developed based on our original agents. The objective of the KUT-RP system is to enable human players to participate soccer games by operating agents called KUT-RP agents. Fig. 2 shows the conceptual diagram of the KUT-RP system.

The KUT-RP system is composed mainly of three components, i.e., the agent core, the real-player monitor, and the standard input devices. The agent core has the following modules, namely,

communication module that copes with communication between the soccer server and the KUT-RP agent,
memory module that processes and stores the information sent from the soccer server as well as the real-player monitor, and
composite command execution module that performs composite commands composed of series of at least one of the three basic commands.

In practice, it is not feasible for a human player to keep on issuing the three basic commands. To cope with this problem, three composite commands are developed. Each of them is described as follows:

kick to specified position that makes the KUT-RP agent kick the ball toward the specified position,
dash to specified position that makes the KUT-RP agent dash to the specified position, and
dribble to specified position that makes the KUT-RP agent dribble the ball to the specified position.

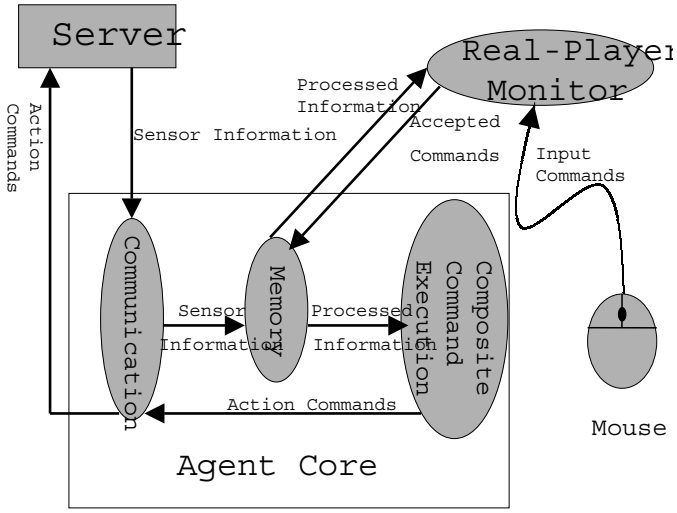

Fig. 2. Conceptual diagram of the proposed KUT-RP system.

In addition to these composite commands, the KUT-RP agent can be executed in one of the following three operation modes, namely,

full-auto mode that will play automatically and not take into account any composite commands issued by the human player,

semi-auto mode that will automatically trace the ball if any of the composite commands is not issued, and

manual mode that will wait until one of the composite commands is issued.

The real-player monitor is different from the RoboCup soccer monitor where all available information and objects are visualized. In the real-player monitor, only the visual information available to the KUT-RP agent is displayed. The displayed information includes both the information sent from the soccer server and the information processed and enriched by the memory module. One example of the latter type of information is the predicted ball position when the ball can not be actually seen by the KUT-RP agent due to restricted vision scope. Fig. 3 shows a display of the real-player monitor before the ball is kicked off. In this figure, a KUT-RP agent is shown on the left side of the center circle, the ball in the middle. Two opponent players are shown in the right side of the soccer field. Our current version of the real-player monitor is also equipped with a bar showing the remaining stamina of the KUT-RP agent, a display box showing the most recently clicked button of the mouse (to be described soon below), and another display box showing the agent dashing speed.

Fig. 3. Display of the real-player monitor.

In this study, a mouse is used as the input device of the KUT-RP system[3]. The left, middle, and right buttons correspond respectively to **dash to specified position**, **kick to specified position**, and **dribble to specified position** composite commands. The specified position of these commands is defined by pointing the mouse cursor to the desired position in the real-player monitor while clicking one of the mouse buttons.

In addition to the components and modules described above, the KUT-RP system is also equipped with a function to record logs of the agent's commands and the sensor information when these commands are issued by the human player.

4 Experiments of Learning from Human Decision-Making Behaviors

We extensively conducted experiments in order to verify whether it is possible to learn the decision-making behaviors of the human player who operates a KUT-RP agent. In these experiments, a KUT-RP agent under the semi-auto mode played against two opponents (one defender and one goalkeeper), as shown in Fig. 3. Logs were then taken from each simulation game taking in total of 6000 simulation cycles (10 minutes in the default mode). Condition-action rules were extracted by C4.5 from the logs recorded with the KUT-RP system explained

[3] We are now developing a new user interface that can incorporate inputs from the keyboard

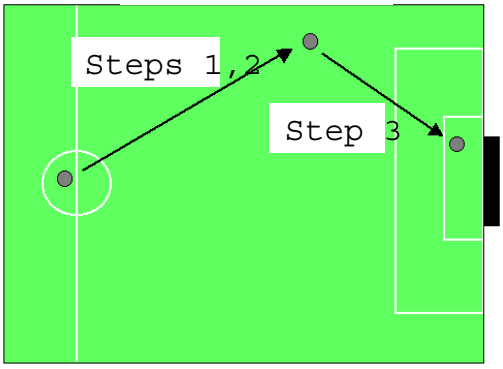

Fig. 4. Basic strategy from step 1 to 3.

above. The generated decision tree was then applied to an agent henceforth called the virtual human agent. The virtual human agent was designed by two very simple if-then rules:

If the virtual human agent holds the ball,

then consult the decision tree whether to kick or shoot,

and

If the virtual human agent does not hold the ball, then trace the ball.

We examined the scores obtained by the virtual human agent when it was applied with different decision trees generated by accumulated logs of different numbers of training games.

The opponent defender agent was programmed in such a way that it moves at the highest speed to and forwardly kick the ball at the highest power when the distance to the ball is relatively near (below 7 meters). Otherwise, it moves back to and waits at its initial position located in front of the goalkeeper but with some distance. This situation makes it difficult for an offending agent like our KUT-RP agent to dribble or to shoot the ball directly through this defender agent toward the center of the opponent goal. A very simple but effective strategy that the human player eventually adopted for playing the KUT-RP agent can be described below as follows:

1. kick the ball to either side (top or bottom),
2. trace the ball,
3. shoot the ball while keeping it away from the goalkeeper,
4. if the ball is blocked and kicked back by the goalkeeper, then trace the ball and repeat from step 1.

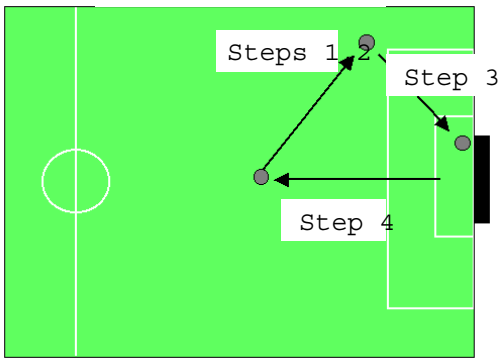

Fig. 5. Action sequences when the ball is blocked and kicked back by the goalkeeper.

Attribute Name	Attribute Type
x	continuous
y	continuous
body_angle	continuous
ball_direction	continuous
ball_distance	continuous
opponent_direction	continuous
opponent_distance	continuous

Table 1. Attribute names and types of the input data.

Figs. 4 and 5 visually show this strategy.

In order to generate decision trees, we have to define classes and input attributes as well as their types. In this study, they are summarized in Tables 1 and 2. The input attributes in Table 1 are selected from the information sent to the agent by the soccer server. The attributes x and y altogether represent the absolute coordinate of the agent in the soccer field. The attribute *body_angle* indicates the relative direction of the body of the agent toward the center of the opponent goal. The attributes *ball_direction*, *ball_distance*, *opponent_direction*, and *opponent_distance* are the relative direction and distance to the ball and to the nearest opponent, respectively. The classes in Table 2 represent agent actions. The class *kick* is divided into 19 subclasses according to different kicking directions relative to the agent's body direction. The class *shoot* is not explic-

Class Name	Subclass Name
kick	-90,-80,-70,-60,-50,-40,-30,-20,-10,0 ,10,20,30,40,50,60,70,80,90
shoot	Top,Bot,Mid

Table 2. List of classes and their subclasses.

itly provided by the KUT-RP system, but can be derived from the **kick to specified position** composite command by checking whether the position inside the opponent goal area is specified or not.

For comparisons, we conducted an experiment in which a programmer who has good understanding of Java but is relatively new to RoboCup simulation domain was asked to hand code an agent for playing against the two opponents described above. In addition, we also conducted an experiment using an agent trained with reinforcement learning for the same competition. For reinforcement learning, we used the Q learning with 3 states, each having 4 possible actions, namely,

state 1: far from opponent goal in which the distance between the agent and the opponent goal is between 20 and 40 meters,
state 2: near to opponent goal in which the distance between the agent and the opponent goal is below 20 meters,
state 3: near to opponent player in which the distance between the agent and the nearest opponent player is below 7 meters,
action 1: shoot to the near-side goal post by which the ball is shot to the near-side goal post,
action 2: shoot to the far-side goal post by which the ball is shot to the far-side goal post,
action 3: dribble to the near-side goal post by which the ball is dribbled to the near-side goal post,
action 4: dribble to the far-side goal post by which the ball is dribbled to the far-side goal post.

In addition, when the reinforcement learning agent is not in one of the three states above, it will trace the ball and dribble the ball toward the center of the opponent goal.

5 Experimental Results

Fig. 6 shows the scores obtained by the KUT-RP agent for five games. At the first game, the human player was still not familiar with the system. As a result, the obtained scores were relatively low. More stable scoring could be seen from the second game.

Fig. 7 summarizes the averaged and highest scores from five games obtained by the hand-coded agent, the KUT-RP agent, the reinforcement learning agent and the virtual human agent, when the last two were trained by various numbers of games from one to five. The number of games used for training the virtual human agent indicates the number of games that the human player plays the KUT-RP agent; more games means more logs or training samples for extracting condition-action rules.

From the results in Fig. 7, it can been seen that the performance of the virtual human agent improves as the number of training games increases. When five games were used for training, its performance become comparable to that

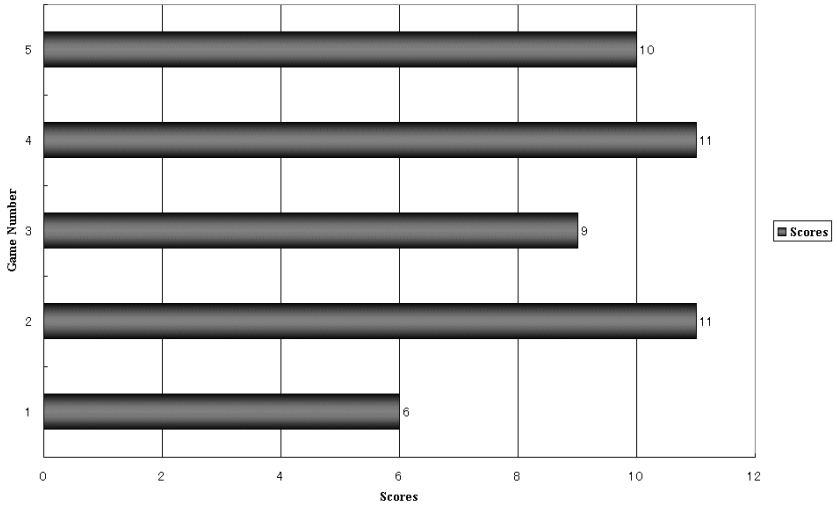

Fig. 6. Scores obtained by the KUT-RP agent.

of the KUT-RP agent. Visual comparisons of the movements of both agents also confirmed that they had similar behaviors.

In addition, the performance of the virtual human agent is superior to that of the reinforcement learning agent for all numbers of training games. Though one can argue that the design of states and actions of the reinforcement learning agent were not optimal, optimizing such designs is not an easy task and might even take longer training time (more training games) due to higher degree of complexity.

Compared with the hand-coded agent, the virtual human agent has subtly lower performance. It turned out that the programmer adopted a similar strategy to the one used for the KUT-RP agent, but the hand-coded agent had more precise shooting ability. However, it took at least five hours of coding time including the time for laying out the strategy. To train the virtual human agent, it took approximately one hour for taking the logs of five games and extracting the rules, noting that 10 minutes are required for playing one game.

6 Conclusions

Applying human decision-making behaviors to software agents is an effective approach for implementing complicated condition-action rules. The experimental results given in this paper confirm this conclusion. In addition, because humans tend to incorporate prediction using their instincts into decision-making process, it can be expected that agents with such ability can be readily built using the presented approach. Validation of this conjecture, optimization of the rule ex-

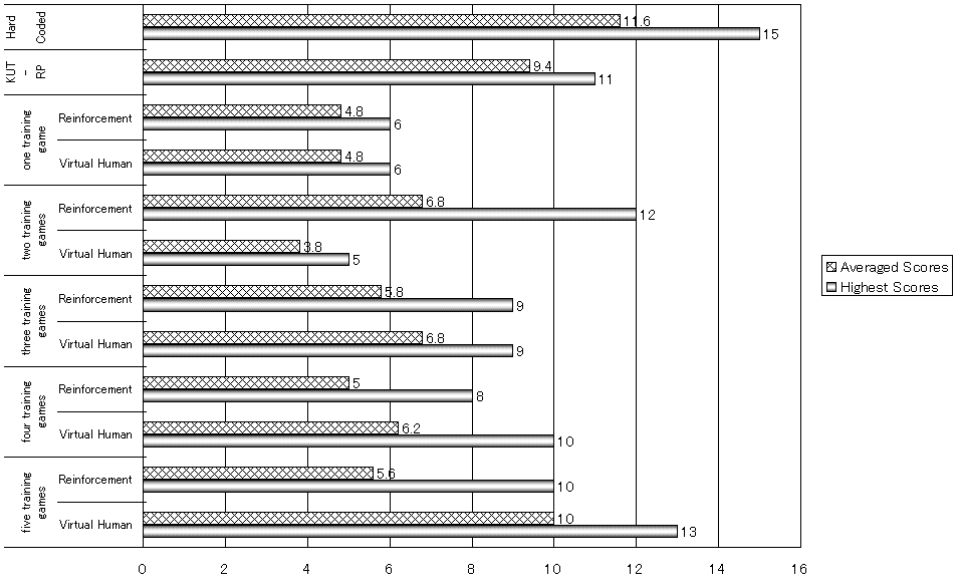

Fig. 7. Comparisons of the highest and averaged scores obtained by various kinds of agents used in the experiments.

traction methodology, and applications of the approach to industrial uses are left as our future studies.

References

1. Sutton, R.S., Barto, A.G.: Reinforcement Learning (Adaptive Computation and Machine Learning). MIT Press (1998)
2. Quinlan, J.R.: C4.5 Programs for Machine Learning. San Mateo, Morgan Kaufmann (1993)
3. Kitano, H., Kuniyoshi, Y., Noda Y., Asada M., Matsubara H., Osawa, E.: RoboCup: A Challenge Problem for AI. AI Magazine, Vol. 18, No. 1 (1997), 73-85
4. Nishino, J, et al.: Team OZ-RP: OZ by Real Players for RoboCup 2001, a system to beat replicants. (2001) (submitted for publication)
5. Thawonmas, R.: Problem Based Learning Education using RoboCup: a Case Study of the Effectiveness of Creative Sheets. International Symposium on IT and Education (InSITE 2002), Kochi, Jan. (2002)