Learning of Cooperative actions in multi-agent systems: a case study of pass play in Soccer

Hitoshi Matsubara, Itsuki Noda and Kazuo Hiraki

Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba, Ibaraki, 305 JAPAN
e-mail: {*matsubar, noda, khiraki*} @etl.go.jp

Abstract

From the standpoint of multi-agent systems, soccer (association football), which is just one of usual team sports, make a good example of problems in the real world which is moderately abstracted. We have chosen soccer as one of standard problems for the study on multi-agent systems, and we are now developing a soccer server, which provides a common test-bench to various multi-agent systems. We present an experiment on cooperative action learning among soccer-player agents on the server. Our soccer agent can learn whether he should shoot a ball or pass it.

Introduction

Soccer (association football) is a typical team game, in which each player is required to play cooperatively. And soccer is a real-time game in which situation changes dynamically. We have chosen soccer as one of standard problems for the study on multi-agent systems, and we are now developing a soccer server for the research. We present an experiment on cooperative action learning among soccer-player agents on the server.

Soccer as a Standard Problem

From the standpoint of multi-agent systems, soccer (association football), which is just one of usual team sports, make a good example of problems in the real world which is moderately abstracted. Multi-agent systems provides us with research subjects such as cooperation protocol by distributed control and effective communication, while having advantages as follows:

- efficiency of cooperation
- adaptation
- robustness
- real-time

Soccer has the following characteristics:

- robustness is more important than elaboration.
- adaptability is required for dynamic change of plans according to the operations of the opposing team.

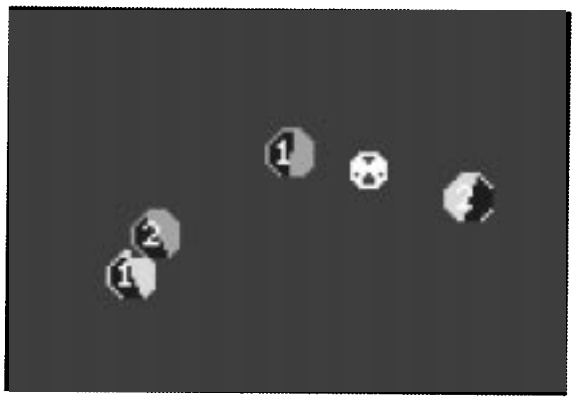

Figure 1: Soccer players and a ball

- Team play is advantageous.
- As precise communication by language is not expected, effectiveness must be provided by combining simple signals with the situation.

These characteristics show that soccer is an appropriate example for evaluation of multi-agent systems. Recently, soccer has been selected as an example on real robots as well as software simulators (Asada 1995; Sahota 1993; Sahota 1994; Stone 1995). Robo-cup, the robot world cup initiative, will be held in IJCAI-97 (Kitano 1995). Many of those experiments, however, have their own ways of setting, which makes it difficult to make a comparison among them. For satisfying the need for a common setting, we are developing a soccer server(Kitano 1995; Noda 1995). Adoption of this soccer server as a common sitting makes it possible to compare various algorithms on multi-agent systems in the form of a game.

Soccer Server

Fig.1 shows soccer players and a ball in our soccer server. Fig.2 shows an one-shot example of the server. This server will be used as one of the official servers of Robo-Cup in IJCAI-97.

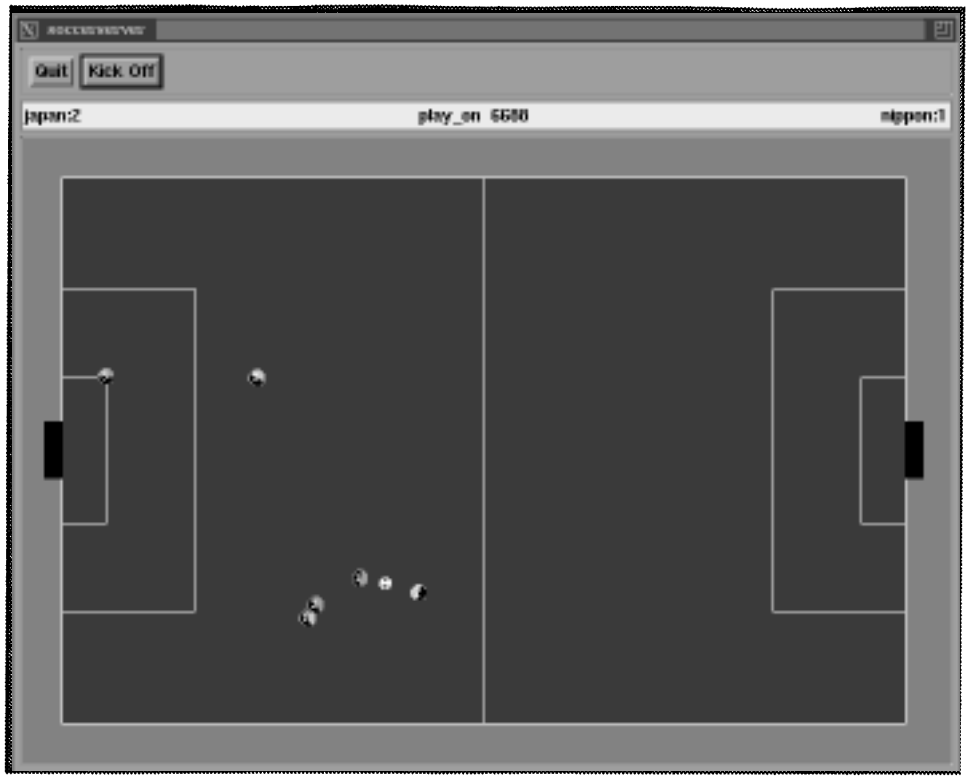

Figure 2: Window Image of Soccer Server

Overview

The soccer server provides a virtual field where players of two teams play soccer. Each player is controlled by a client program via local area networks. Control protocols are simple in that it is easy to write client programs using any kind of programming system that supports UDP/IP sockets.

- Control via Networks

  A client can control a player via local area networks. The protocol of the communication between clients and the server is UDP/IP. When a client opens a UDP socket, the server assigns a player to a soccer field for the client. The client can control the player via the socket.

- Physical Simulation

  The soccer server has a physical simulator, which simulates movement of objects (ball and players) and collisions between them. The simulation is simplified so that it is easy to calculate the changes in real-time, but the essence of soccer is not lost.

  The simulator works independently of communications with clients. Therefore, clients should assume that situations on the field change dynamically.

- Referee

  The server has a referee module, which controls each game according to a number of rules. In the current implementation, the rules are: (1) Check goals; (2) Check whether the ball is out of play; (3) Control positions of players for kick-offs, throw-ins and corner-kicks, so that players on the defending team keep a minimum distance from the ball.

  Judgments by the referee are announced to all clients as an auditory message.

Protocol

As described above, a client connects to the server by a UDP/IP socket. Using the socket, the client sends commands to control its player and receives sensory information of the fields. Command and sensor information consist of ASCII strings.

- Initialization

  First of all, a client opens a UDP socket and connects to a server host.(The server's port number is assumed to be 6000.) Then, a client sends the following string to the server:

  (init *TEAMNAME* (*POS-X POS-Y*)).

  The server assigns a player whose team name is *TEAMNAME*, and initializes its position at (*POS-X*,*POS-Y*). If the initialization succeeds, the server returns a following string to the client:

(init *SIDE NUMBER TIME*),

where *SIDE*, *NUMBER*, *TIME* indicates the side of the team ('l' or 'r'), the player's uniform number, and the current time respectively.

- Action Command

  Each client can control its player by 3 kinds of commands, 'turn', 'dash' and 'kick'. Clients also can communicate with other clients using the 'say' command. Command formats are as follows:

(turn *POWER*)	Change the direction of the player according to *POWER*. *POWER* should be $-180 \sim 180$.
(dash *POWER*)	Increase the velocity of the player toward its direction according to *POWER*. *POWER* should be $-30 \sim 100$.
(kick *POWER DIRECTION*)	Kick a ball to *DIRECTION* according to *POWER*. *POWER* should be $-180 \sim 180$, and *DIRECTION* should be $0 \sim 100$. This command is effective only when the distance between the player and the ball is less than 2.
(say *MESSAGE*)	Broadcast *MESSAGE* to all players. *MESSAGE* is informed immediately to clients using (hear ...) format described below.

  These commands may not cause expected effects because of physical situation (ex. collisions) and noise.

- Sensory Information

  Players can see objects in their views. The direction of view of a player is the direction of movement of the player, and the angle of view is 60 degrees. Players also can hear messages from other players and the referee. This information is sent to clients in the following format:

(see *TIME OBJINFO OBJINFO* ...)	Inform visual information. *OBJINFO* is information about a visible object, whose format is (*OBJ-NAME DISTANCE DIRECTION*). This message is sent 2 times per second. (The frequency may be changed.)
(hear *TIME SENDER MESSAGE*)	Inform auditory information. This message is sent immediately when a client *SENDER* sends (say *MESSAGE*) command. *TIME* indicates the current time.

Coach Mode

In order to make it easy to set up a certain situation, the server has a coach mode. In this mode, a special client called 'coach' can connect to the server, who can move any players and the ball. This mode is useful for learning client programs.

Current Status

The server is implemented by g++ and X window with Athena widgets.

WWW home page

Programs of the soccer server are available by FTP:

`ftp://ci.etl.go.jp/pub/soccer/server/sserver-1.8.tar.gz`

Home page of the soccer server is:

`http://ci.etl.go.jp/~noda/soccer/server.html`

We also have a mailing list about Robo-Cup:

`rjl@csl.sony.co.jp`

Learning of pass play

On this soccer server, we are conducting an experiment on cooperative-action learning among soccer players. Our first aim is to make the players learn how to kick a pass, which is supposed to be the fundamentals of cooperative action in soccer. The information that shooting the ball for the enemy's goal is the purpose of the game is already known to the player agents. It is easy for a successful shooting which leads to the team score when there are no enemy players in the area between him and the goal, that is, when he is a free player trying to shoot a goal.

When there are two players from his side and one player from the other, and the player of the enemy's team is defending against him, it is a wiser policy to pass the ball to a free player of his team than to make a shoot by himself. Evaluation of action is based on the score and the time taken for making the score.

We carried out an experiment on our server as follows:

- There are two offense players and one defense player in a penalty area.
- One offense player A is set at random. A ball is at his feet in the first place. He will shoot the ball to the goal or pass it to his teammate. He cannot move anywhere.
- Another offense player B and one defense player C are set at random. C keeps his position between the ball and the goal. In some situations (see Fig.3) C marks A (to pass the ball to B is a good strategy for A). In other situations (see Fig.4) C marks B (to shoot the ball to the goal directly is a good strategy).
- B is programmed to wait the pass from A and shoot the ball.

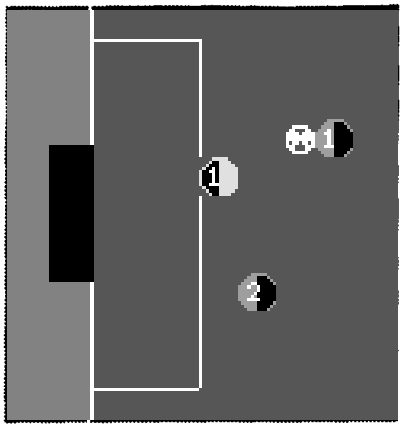

Figure 3: Situation where pass is better

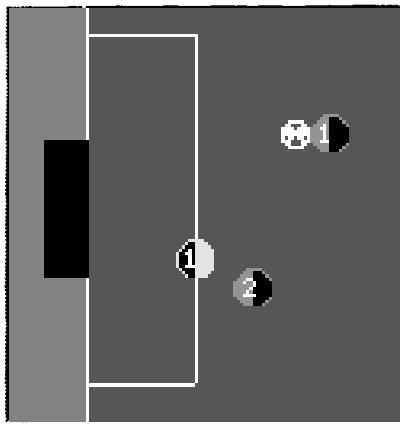

Figure 4: Situation where shoot is better

- Before learning, A either shoots the ball or passes it at random. A coach tells *nice* to A when the offense players get a goal, and tells *fail* when the ball is out of field or time is over.
- A learns his strategy by back propagation in a neural network. The neural network consists of 8 input-units, 30 hidden-units and 2 output-units. The network receives relative positions of B, C, the goal and the ball. And it outputs two values (O_{pass}, O_{shoot}), which indicate expected success rate of pass and shoot.
- A chooses pass or shoot according to the rate of O_{pass} and O_{shoot}: the probability of choosing pass is $O_{pass}/(O_{pass} + O_{shoot})$.

Fig.5 shows changes of outputs of the neural networks for direction to the opponent player C. The situation (from the viewpoint of A) is as follows:

- The goal: angle is 16 degrees on the left
  distance is 14.0
- B: angle is 54 degrees on the left
  distance is 4.5
- C: angle changes from 30 degrees on the left to 30 degrees on the right
  distance is 5.5

Intuitively speaking, it is better to shoot the ball directly when C is on the left. It is better to pass the ball when C is between A and the goal. And either shoot or pass is OK when C is on the right.

Fig.6 shows the rates of successes and failures of shoot and pass before and after learning. We can see that the success rate of shoot is improved. After learning, A does not make fruitless shoots. The success rate of pass is not so improved. If success rate of B's shoot was higher, success rate of A's pass would be improved by learning.

Concluding remarks

In this article, we reported our soccer server and an experiment of pass-play learning on the server. The result shows that our server is useful for researches of this kind and that our soccer agent can learn his strategy. We are making further experiments now. For example, we are trying *ID-3* as a learning method instead of back propagation.

References

M. Asada, S. Noda, S. Tawaratsumida and K. Hosoda: Vision-based reinforcement learning for purposive behavior acquisition, Proc. of IEEE Int. Conf. on Robotics and Automation (1995)

H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda and E. Osawa: RoboCup: The robot world cup initiative, IJCAI-95 Working Notes of Entertainment and AI/Alife Workshop (1995)

I.Noda: Soccer server: a simulator of RoboCup, AI symposium '95, Japanese Society for Artificial Intelligence (1995)

M.K. Sahota: Real-time intelligent behavior in dynamic environments: soccer-playing robot, Master Thesis, The University of British Columbia, Canada (1993)

M.K. Sahota: Reactive deliberation: an architecture for real-time intelligent control in dynamic environments, Proc. of AAAI-94, pp.1303-1308 (1994)

P. Stone and M. Veloso: Learning to pass or shoot: collaborate or do it yourself, unpublished manuscript (1995)

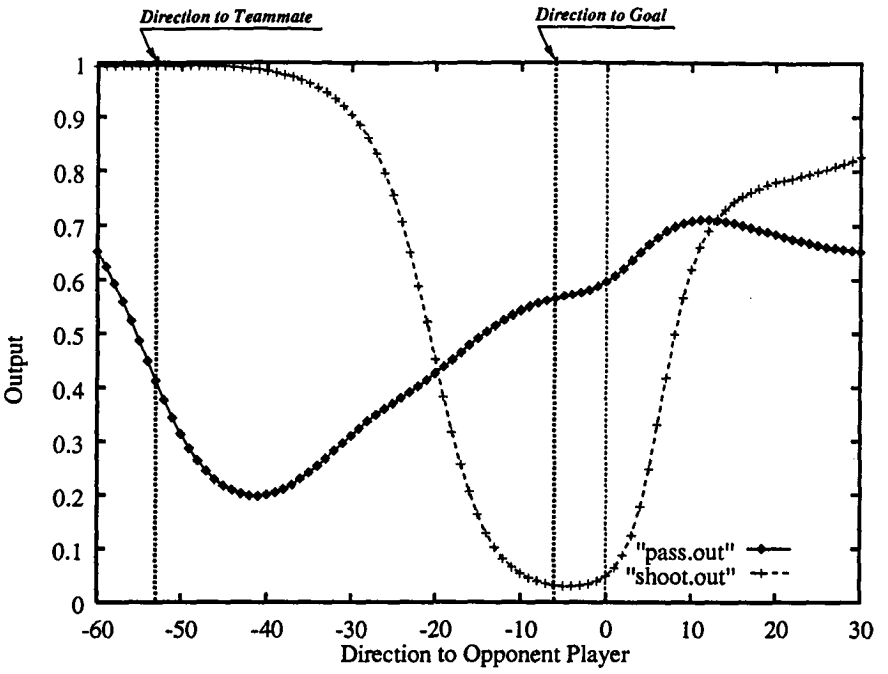

Figure 5: Change of output for direction to opponent player

[learning data]

	pass	shoot	total
nice	341(24.6%)	527(36.1%)	868(30.5%)
fail	1045(75.4%)	931(63.9%)	1976(69.5%)
total	1386	1458	2844

[before learning]

	pass	shoot	total
nice	435(29.9%)	665(44.1%)	1100(37.1%)
fail	1020(70.1%)	844(55.9%)	1864(62.9%)
total	1455	1509	2964

[after learning]

	pass	shoot	total
nice	506(31.4%)	708(52.1%)	1214(40.9%)
fail	1105(68.6%)	651(47.9%)	1756(59.1%)
total	1611	1359	2970

Figure 6: Success rate before and after learning