Team Description of opuCI_2D 2009

Takesuke Uenishi and Tomoharu Nakashima

Department of Computer Science and Intelligent Systems
Graduate School of Engineering, Osaka Prefecture University
Gakuen-cho 1-1, Naka-ku, Sakai, Osaka, 599-8531
uenishi@ci.cs.osakafu-u.ac.jp
nakashi@cs.osakafu-u.ac.jp

Abstract. This paper describes opuCI_2D , our soccer team that has been submitted to the qualification for the competition of the soccer 2D simulation league of RoboCup 2008. The main characteristic feature of this team is to use neural networks for team formation. The position of our players who are not in ball possession is determined by a neural network based on the current ball position. The position of opponent players are also estimated from the ball position by neural networks. We show a series of computational experiments to show that the use of neural networks gives a beneficial effect on our team.

1 Introduction

Team opuCI_2D is the new project in Osaka Prefecture University (OPU) after OPU_hana_2D project finished. CI in opuCI_2D means computational intelligence, which is the main research field of our laboratory.

So far we have applied various computational intelligence techniques [1] such as fuzzy logic [2], neural networks [3], evolutionary computation [4], and reinforcement learning.

2 Formation Learning by Neural Networks

In this paper, we use the term "team formation" to refer to the positioning of player agents based on the ball position. Although this definition is not exactly correct in all cases, it works for most of teams because the ball position is one of the most important factor to consider the game situation. Thus the problem here is to learn the mapping from the ball position to the player position. Three-layered neural networks are used to realize this mapping.

We show the architecture of the neural networks in Fig. 1. Since the input of the target mapping is the ball position, there are two units in the input layer and the output layer, where the coordinate (i.e., x and y) of the ball (b_x, b_y) and the player (p_x, p_y) is used as input/output information.

The number of hidden units indicates the complexity level of the input-output mapping that neural networks provide. That is, complex and non-linear mappings can be obtained if the number of hidden units is large. The disadvantage

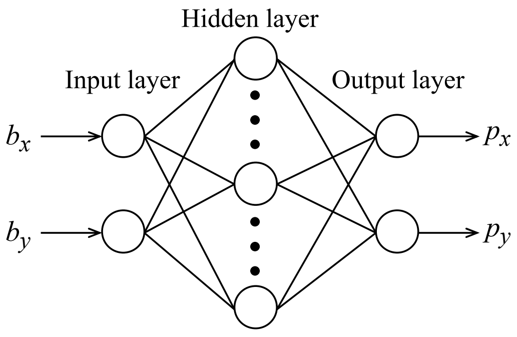

Fig. 1. Three-layered feed forward neural network.

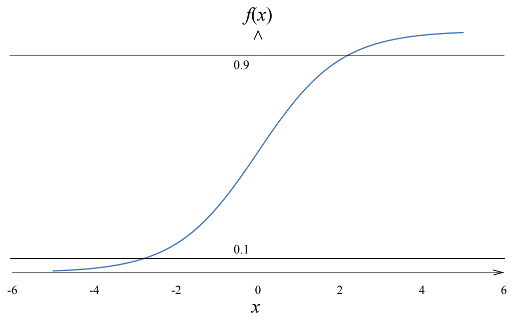

Fig. 2. Sigmoid function (Cutting lines at $f(x) = 0.1, 0.9$).

of having a large number of hidden units is that there will be too many weights to tune during learning and thus the learning speed will be very slow or often stuck in a local minimum. In this paper, nine hidden units are used for on-line learning and ten hidden units for off-line learning.

The activation function used to calculate the output of each unit is the sigmoid function (see Fig. 2).

While the output range of the neural networks is from 0 to 1 due to the nature of the sigmoid function, we modified the output values from the neural networks so that the minimum and the maximum value is 0.1 and 0.9 as follows:

$$o'_{NN} = \max\{0.1, \min(0.9, o_{NN})\}, \tag{1}$$

where o_{NN} is the output value from a neural network and o'_{NN} is the modified output. The reason for this modification is that we want the neural network to learn efficiently. It is obvious that the derivative of the activation function is not very sensitive around its extreme values (i.e., around the output values 0.0 and 1.0). This modification is simple but powerful for efficient learning for the valid output interval (i.e., from 0.1 to 0.9).

One neural network corresponds to the mapping for one player in this paper. Thus 11 neural networks are used for our team formation and another 11 neural networks for the opponent team formation.

3 Implementing Neural Networks for On/Off-Line Learing

We implemented two modes of the formation learning: On-line and Off-line learning. The on-line learning aims to learn the opponent team formation during the course of games. On the other hand, the off-line learning forms the formation of our teams by imitating that of other good teams. In the following subsections, we will explain these two implementations in detail.

3.1 On-line learning

The purpose of the on-line learning is to learn opponent team formation during a game. Since player agents cannot have full information on the team formation by regulation, a coach agent is used for the on-line learning of team formation.

Once a game starts, the coach agent records the ball position and the corresponding position of all the 11 opponent players. That is, an input-output pair is monitored for each opponent player per time step. When a certain amount of input-output pairs are obtained, the coach agent trains 11 neural networks using the input-output pairs. The training of neural networks are done as threads so that the coach agent can continue to collect input-output pairs immediately after it spawns the neural network threads.

In each thread, the neural network is trained until a pre-specified termination condition is satisfied. After the training of neural networks are terminated, the coach agent sends the weights of trained neural networks to player agents. The official rule allows the coach agent to send a free message only when the game is suspended (e.g., before_kickoff, kickin, etc.). Furthermore there is a limit on the length of free message (128 bytes). The coach agent first compress the information on the trained neural networks by converting the decimal system to the 72-base system because there are 72 characters that are allowed to use in the free message format (a-z, A-Z, 0-9, (,), ., +, *, /, ?, <, >, and _). By the above procedure, the information on the weights of a neural network with nine hidden units is coded into 256 characters. Since it takes $256/128 = 2$ time steps to send the information of a neural network, $2 \times 11 = 22$ time steps are necessary to send the information for 11 opponent player agents.

Even after sending the weight information to player agents, the coach agent has the right to spawn another neural network threads for additional learning. In Section 4, we investigate various settings of the on-line learinng.

3.2 Off-line learning

In contrast to the on-line learning in Subsection 3.1, log files are used for the off-line learning. The purpose of the off-line learning is to imitate a team formation from strong teams. For this purpose, we used log files of RoboCup 2008 which was held in Suzhou, China. While there are two types of log files (RCG and RCL), we used only RCG files because RCG files include all the necessary information on team formation.

First we generate a training data set from log files by extracting the relation between the ball position and the corresponding positions of player agents. The extraction process is performed only when the game mode is play-on mode in the log file. After preparing a training data set, a single neural network is trained for the learning of one ball-player mapping. In the off-line learning, there is no time limit because the learning takes place after games. Thus we can perform the learning much longer than in the case of on-line learning. Also there is no need for sending the information of trained neural networks in a limited message length as in on-line learning. So for the off-line learning, the number of hidden units is specified as ten so that the mapping realized by the trained neural networks can represent any complex relation between the ball position and the players' position.

4 Computational Experiments

In this section we show some computational experiments where the learning ability of neural networks are investigated.

4.1 On-line learning

In the on-line learning, it is important when to start the learning of neural networks and when to stop because the learning process is performed during the course of a game. Although we have to start the learning as fast as possible, there are not enough information to train neural networks at the beginning of the game. So we have to collect as many training patterns as possible. However, for example after 6000 training patterns are collected (i.e., 6000 time steps has passed), such collected training patterns are useless any more because the game is soon to end (or already finished). In this subsection, we investigate the learning performance of neural networks with various amount of training patterns. We also investigate the learning performance with various number of learning iterations. For this experiments, HELIOS2008 was used to evaluate the learning ability of the neural networks. In Fig. 3 we show the results of the experiments where the training of neural networks started after 1000 time steps have passed. It should be noted that training patterns are not generated if the game mode is not play-on. Thus the number of the collected training patterns is less than or equal to 1000. From Fig. 3, we can see that the mean square error (MSE) decreases as we continue to train the neural networks. We can also see that 1000 iterations for the learningis almost enough for practical use with a sufficiently low error.

Next, we fixed the number of learning iterations as 1000 but changed the collecting time step of training patterns. Figure 4 shows the experimental results

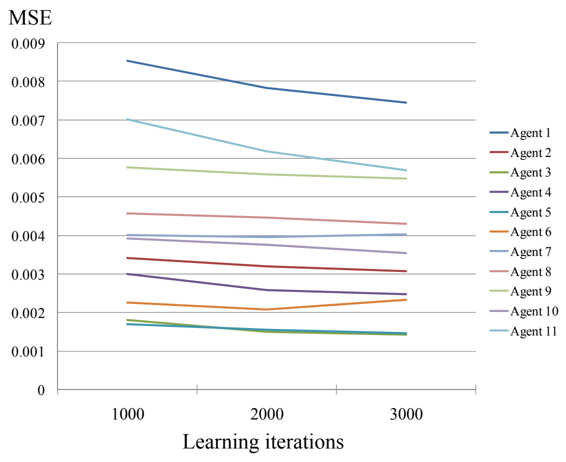

Fig. 3. Experimental results (Amount of data: 1000 time steps).

where we started the training of neural networks after 1000, 2000, and 3000 time steps. The training terminates by the learning iterations of 1000. From this figure we can see that the error increases as the collecting time step increases. This is because there are more training patterns when collecting time steps are longer and it became more difficult to find appropriate weights of the neural networks within 1000 iterations. In another experiments, we confirmed that the error decreases when we continued to train neural networks even after 1000 iterations.

4.2 Off-line learning

We also examined the performance of neural networks for the off-line learning. Because of the regulation of the qualification (TDP should be 4 pages to 6), we do not have enough space to present the experimental results for the off-line learning. To report the experimental results of the off-learing briefly, we observed that lower error rate was achieved and continued to decrease for a long learning iterations when the learing rate is 0.2 and inertia moment rate 0.2.

5 Conclusions

This paper described the development of this year's team opuCI_2D . The main feature of the team is the use of neural networks for team formation learning. The standard three-layered neural networks are used for this purpose. We showed on-line and off-line modes of the team formation learning. In the on-line mode, training patterns for the learning of the neural networks are incrementally stored by monitoring the ball position and the corresponding position of player agents. Log files are used to extract training patterns in the off-line mode. A series

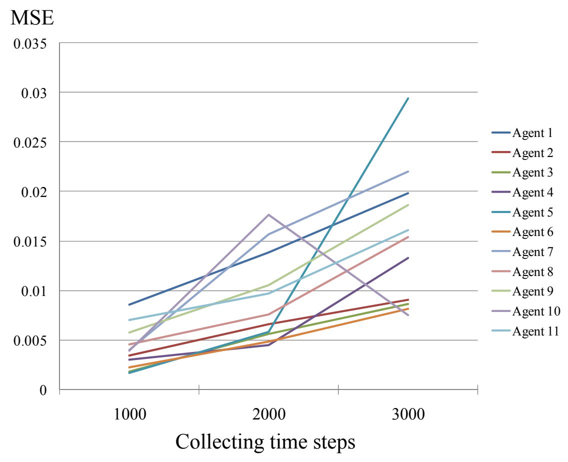

Fig. 4. Experimental results (Learning iterations: 1000).

of experiments showed the learning ability of the neural network, and we also showed some practical application where trained neural networks are successfully used.

It should be noted that in this paper we define the team formation as the relation between the ball position and the positions of player agents. Thus for those teams that do not actively use the ball position to determine players' position, this learning method does not necessarily work well. Nevertheless it is almost certain that the ball position is one of the most important factor for most of teams. Thus the approach taken in this paper should work in practice at the competition.

References

1. T. Nakashima and H. Ishibuchi, "Computational Intelligence in RoboCup Soccer Simulation", *Computational Intelligence: Principles and Practice*, G. Y. Yen and D. B. Fogel (eds.), IEEE Computational Intelligence Society, pp. 217–236, 2006.
2. T. Nakashima, M. Udo, and H. Ishibuchi, "A fuzzy reinforcement learning for a ball interception problem," *Proc. of RoboCup 2003 Symposium*, in CD-ROM (8 pages), Padova, Italy, July 10-11, 2003.
3. T. Nakashima, and H. Ishibuchi, "Mimicking Dribble Trajectories by Neural Networks for RoboCup Soccer Simulation," *Proc. of 2007 IEEE Multi-conference on Systems and Control*, pp.658-663, Singapore, 2007.
4. T. Nakashima, M. Takatani, M. Udo, H. Ishibuchi, and M. Nii, "Performance evaluation of an evolutionary method for RoboCup soccer strategies," *Proc. of RoboCup 2005 International Symposium*, in CR-ROM (8pages), Japan, 2005.