HELIOS2023: Team Description Paper

Hidehisa Akiyama[1][0000−0002−9793−9123],
Tomoharu Nakashima[2][0000−0002−1443−0816],
Kyo Hatakeyama[2], Takumi Fujikawa[2], and Akei Hishiki[2]

[1] Okayama University of Science, Okayama, Japan
hidehisa.akiyama@ous.ac.jp
[2] Osaka Metropolitan University, Osaka, Japan
tomoharu.nakashima@omu.ac.jp

Abstract. This team description paper presents an overview of previous work and recent research topics of Team HELIOS2023. We have been attempting to construct a method that facilitates the construction of tactical decisions that reflect the intentions of the team designers, intending to operate a variety of tactics flexibly. This paper outlines one of our recent efforts that use learning to rank to make tactical decisions about ball chasing behavior.

1 Introduction

Team HELIOS has participated in the RoboCup competitions since 2000, and has won five championships [6,7,9]. The team has always succeeded in being among the top four teams since the year 2005.

One of our recent research topics is the adaptation of team tactics. In the soccer simulation league, changing the team strategy and players' behaviors according to the opponent teams becomes increasingly essential. The contribution described in this paper is applying the learning-to-rank approach to make the best tactical decisions about ball chasing behavior.

2 Previous Works

We have released some of our team's source codes and debugging tools to help new teams participate in the competitions. Furthermore, to start the research of multiagent systems [5]. Currently, the released source codes are available at our project site[3]. We have proposed two essential methods for developing a (simulated) robotic soccer team: A formation model using triangulation [1] and a framework for action sequence planning [2]. These methods have already been implemented in the released sources, allowing us to effortlessly develop a working simulated soccer team.

[3] https://github.com/helios-base (Please cite [5] when you publish papers using the codes in this site.)

Acquiring a practical evaluation function for action sequence planning is still a problem to be solved. We have tried to apply machine learning methods to this problem (such as in [4,12]). As an application of evaluation function research, we also developed an automatic cheering system in [19]. Game analysis is also an important topic related to evaluation function modeling. We proposed several analysis methods using the information about the distribution of kicking actions during the game in [20,11]. To conduct a quantitative evaluation, we have also developed a team evaluation system [15] and a log analysis tool[4]. Analyzing the team name effect is one of the achievements of using this tool [13].

3 Ball Chasing Behavior using Learning to Rank

3.1 Problem Overview

The ball-chasing behavior is a frequently occurring behavior in soccer games. Usually, individual players estimate the predictions of the future ball positions based on the simulator's physics model. When it comes to catching up with the ball in the shortest distance, optimizing the ball-chasing behavior is not so difficult. For example, reinforcement learning has been successfully applied to acquiring the ball-chasing behavior in the early days [18]. However, more is needed for a pass receiver to receive the ball if they need to consider team tactics and make a more significant contribution to the team. The receiver player must find the ball-trap position and its posture that will be more advantageous to the team and select the movement actions according to the decision. In the ball-chasing behavior, it is not easy to determine a more advantageous position and posture for the team, considering the game situation. This task is generally performed in conventional team development by applying manually designed rules and handcrafted evaluation functions.

3.2 Ball Chasing Task in the Soccer Simulation

Figure 1 shows an example of the ball trap position predicted by Player #10. Each player predicts the position of the ball based on its observed information and predicts whether or not he can reach those positions. The figure shows the ball trajectory predicted by Player #10 and the possible positions where Player #10 can trap the ball as circles. It is easy to predict that the ball will pass right by the player and to find the trapping position with the shortest distance to travel. However, Player #10 is playing an offensive role who attacks from left to right, so trapping the ball more toward the opposition, i.e., more to the right, is expected to give his team an advantage. As shown in this case, the most effective ball-trap position is only sometimes apparent because the positional relationship with other players and the team's tactics must be taken into account.

[4] https://github.com/opusymcomp/loganalyzer3

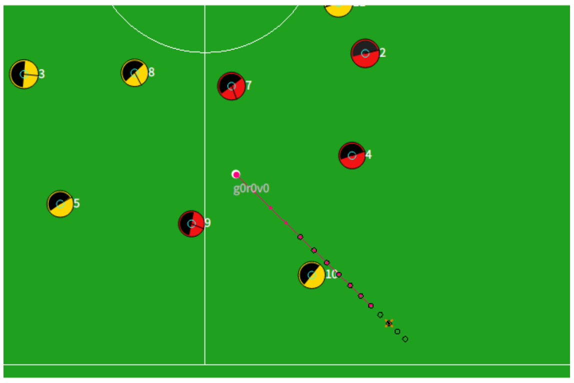


Fig. 1. Example scene showing the future ball positions predicted by Player #10 and candidate positions where it is possible to trap the ball.

3.3 Applying Learning to Rank

Enumerating multiple candidate behaviors and evaluating and ranking them can also be applied to the ball-chasing behavior. To obtain the evaluation function, Akiyama et al. applied a learning-to-rank method using SVMRank. It was also suggested that SVMRank could obtain evaluation functions that perform similarly to those designed by humans. However, the problem is that it is difficult to deal with in exceptional situations. In order to deal with this problem, it is inevitable to write some rules, and an approach using decision trees seems promising for this task.

Learning-to-rank is a machine learning technique widely used in the field of information retrieval. Originally, it was a method for ranking documents that satisfy a particular query in information retrieval. Let us consider the field state when each player generates an action as a query. The candidate action groups generated by the players correspond to the document groups.

This paper uses LightGBM [14] as an implementation tool for the learning-to-rank model. LightGBM is one of the gradient boosting decision trees and the implementation of LamdaMART [10]. LightGBM is a proven library widely used in machine learning competitions.

Gradient boosting decision tree is known as a decision tree-based ensemble learning method. Furthermore, a method for applying gradient boosting trees to learning-to-rank was proposed and has shown practical performance in [10]. We applied a gradient boosting tree-based ranking model to the ball-chasing behavior.

3.4 Feature Vector and Training Data

For the computational experiments in this paper, the feature information used by the players' decision making is dumped out as a log and used as training

Table 1. Features used in our model.

ID	Feature
0	Type of action(Omnidirectional, turn and dash)
1	Number of turn
2	Number of dash
3	Number of steps to trap
4	Distance to the ball when trapping
5	Remaining stamina when trapping
6	X-coordinate of the ball when trapping
7	X-coordinate of the offside line
8	Distance between trap position and opponent goal
9	The shortest steps reached by the opponent
10	The shortest steps reached by the teammate
11	My move distance
12	Radius of my kickable area
13	Current ball speed
14	Ball speed when trapping

data. We use 15 total features shown in Table 1. These features were manually selected and were actually used by HELIOS2021 in competitions.

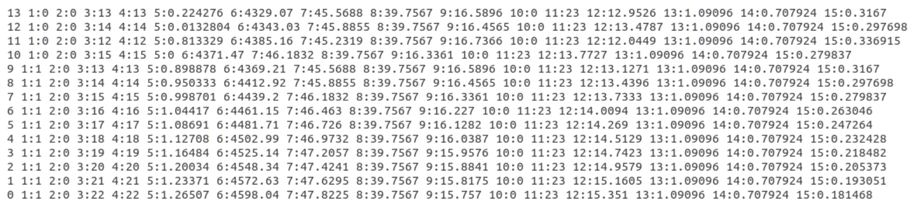
The player is programmed to create logs in which the ball trap position by the manually designed evaluation function is recorded. Therefore, the recorded information helps reproduce the manually designed evaluation function using a machine learning model. Even if the internal implementation of a player is unknown, it is possible to learn and mimic the evaluation function model from the game logs recorded by the simulator.

4 Experiments

4.1 Experimental Setup

A game of 3,000 cycles per half (i.e., 6,000 cycles in total for a single game) was performed, and each player recorded a log that served as training data. Figure 2 shows an example of training data where each line means one behavior candidate. The leftmost column means a value for each behavior, and the remaining columns mean the indices of the features and their values. In this example, 14 candidates are generated and ranked from 1 to 14. Similar data are generated for each game situation.

We employed HELIOS2021 in the computational experiments. The team participated in RoboCup2021 [5,8], and its binary files are publicly available online. In this study, we used only the logs recorded by Player #10 of HELIOS2021, who is in an offensive role in the team. We collected data for a total of 100 games to obtain enough amount of training data. Out of the 100 games, 2, 5, 10, 15, 50, and 100 games were used for training ranking models, and the learning effi-



```
13 1:0 2:0 3:13 4:13 5:0.224276 6:4329.07 7:45.5688 8:39.7567 9:16.5896 10:0 11:23 12:12.9526 13:1.09096 14:0.707924 15:0.3167
12 1:0 2:0 3:14 4:14 5:0.0132804 6:4343.03 7:45.8855 8:39.7567 9:16.4565 10:0 11:23 12:13.4787 13:1.09096 14:0.707924 15:0.297698
11 1:0 2:0 3:12 4:12 5:0.813329 6:4385.16 7:45.2319 8:39.7567 9:16.7366 10:0 11:23 12:12.0449 13:1.09096 14:0.707924 15:0.336915
10 1:0 2:0 3:15 4:15 5:0 6:4371.47 7:46.1832 8:39.7567 9:16.3361 10:0 11:23 12:13.7727 13:1.09096 14:0.707924 15:0.279837
9 1:1 2:0 3:13 4:13 5:0.898878 6:4369.21 7:45.5688 8:39.7567 9:16.5896 10:0 11:23 12:13.1271 13:1.09096 14:0.707924 15:0.3167
8 1:1 2:0 3:14 4:14 5:0.950333 6:4412.92 7:45.8855 8:39.7567 9:16.4565 10:0 11:23 12:13.4396 13:1.09096 14:0.707924 15:0.297698
7 1:1 2:0 3:15 4:15 5:0.998701 6:4439.2 7:46.1832 8:39.7567 9:16.3361 10:0 11:23 12:13.7333 13:1.09096 14:0.707924 15:0.279837
6 1:1 2:0 3:16 4:16 5:1.04417 6:4461.15 7:46.463 8:39.7567 9:16.227 10:0 11:23 12:14.0094 13:1.09096 14:0.707924 15:0.263046
5 1:1 2:0 3:17 4:17 5:1.08691 6:4481.71 7:46.726 8:39.7567 9:16.1282 10:0 11:23 12:14.269 13:1.09096 14:0.707924 15:0.247264
4 1:1 2:0 3:18 4:18 5:1.12708 6:4502.99 7:46.9732 8:39.7567 9:16.0387 10:0 11:23 12:14.5129 13:1.09096 14:0.707924 15:0.232428
3 1:1 2:0 3:19 4:19 5:1.16484 6:4525.14 7:47.2057 8:39.7567 9:15.9576 10:0 11:23 12:14.7423 13:1.09096 14:0.707924 15:0.218482
2 1:1 2:0 3:20 4:20 5:1.20034 6:4548.34 7:47.4241 8:39.7567 9:15.8841 10:0 11:23 12:14.9579 13:1.09096 14:0.707924 15:0.205373
1 1:1 2:0 3:21 4:21 5:1.23371 6:4572.63 7:47.6295 8:39.7567 9:15.8175 10:0 11:23 12:15.1605 13:1.09096 14:0.707924 15:0.193051
0 1:1 2:0 3:22 4:22 5:1.26507 6:4598.04 7:47.8225 8:39.7567 9:15.757 10:0 11:23 12:15.351 13:1.09096 14:0.707924 15:0.181468
```

Fig. 2. Example of training data.

Table 2. Number of collected data.

# of games	# of training data game	# of query	total # of data
2	1	447	5826
5	4	747	9687
10	9	1345	17896
15	14	2077	27729
50	49	8104	108061
100	99	15450	206974

ciency by the number of data was evaluated. Table 2 shows the detailed number of data.

The depth of the decision tree was not specified (thus, its depth is unlimited), and the maximum number of nodes is 31. If the performance is not improved by increasing the number of decision trees through cross-validation, the decision tree generation stops at that point. Normalized discounted cumulative gain (NDCG) is used as a performance indicator for the acquired models. NDCG is a commonly used metric to evaluate ranking models. In this experiment, we used NDCG@10, which is the degree of fit of the top 10 cases.

4.2 Results of Performance Evaluation

Figure 3 is a graph of NDCG@10 values for each number of data. As the number of data increases, the NDCG@10 value also increases, indicating that performance is improving. There was no significant increase in values for more than 50 games. Therefore, approximately 50 games, or about 100,000 data sets, are expected to be sufficient for training the ball-chasing task.

4.3 Analysis of the Contributions of the Features

From the trained model, we analyzed the contributions of the features. The analysis was performed on the model obtained for 50 games. One feature out of 15 was first removed, and then the model was re-trained using the remaining 14 to obtain the value of NDCG@10. This process was iterated for each removal of the features. Figure 4 shows the change in NDCG@10 as a result of removing each feature.

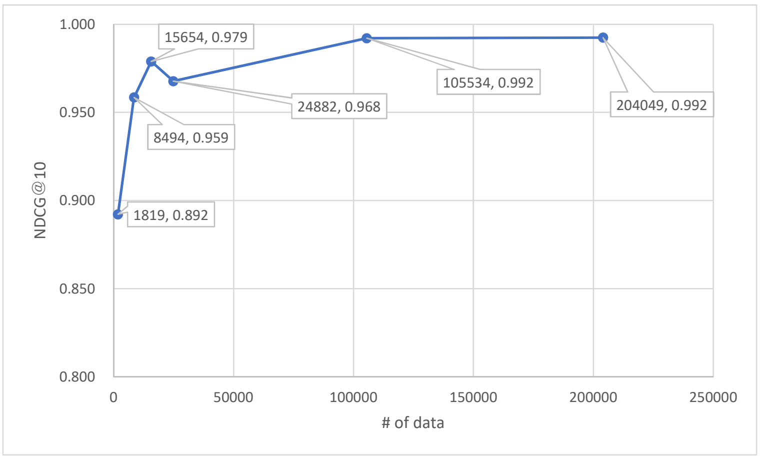


Fig. 3. Result of NDCG@10 for each number of data.

The original NDCG@10 value for the 50 games was about 0.992. This result shows that some features may strongly affect the performance. Feature 4 may be the most important, as the value of NDCG@10 is small. On the other hand, since there is little change in the NDCG@10, features 2, 5, 6, 10, 13, and 14 may not be necessary.

Reducing the number of features may also reduce the data size needed for training. Therefore, it is necessary to determine the balance between the cost of learning and the performance of the learned results.

4.4 Discussions

The performance of the model was evaluated by changing the number of training data. A comparison of NDCG@10 showed that as the number of data increases, the NDCG@10 value also improves. However, no significant increase in NDCG@10 was observed for data larger than about 100,000. Models with NDCG@10 values of approximately 0.99 or higher were obtained for 100,000 data sets. Based on these results, we estimate that approximately 100,000 data sets are sufficient for the ball-chasing task.

The results of feature removal imply that Feature 4, the distance to the ball when trapping, may be essential for the ball-chasing task. This result was not expected from the team developer's point of view. However, it is reasonable to trap the ball more reliably. It is expected that this was unconsciously adjusted during development.

The experimental results indicate that the performance was expected to be practical. However, it was impossible to verify whether the selected ball-chasing behavior was effective. We need to develop a system to judge the effectiveness of the actions and modify the training data.

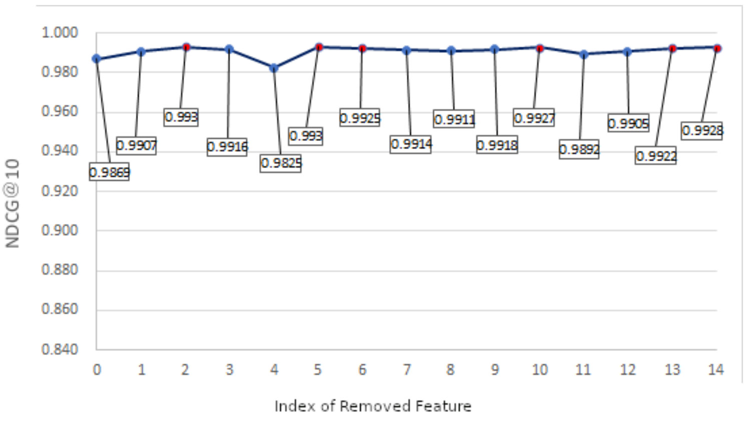


Fig. 4. Result of NDCG@10 when each feature is removed.

5 Conclusions

In this paper, we focused on the ball-chasing task, which has not been sufficiently studied in robotic soccer. We proposed applying a learning-to-rank method using gradient boosting trees. Numerical experiments were conducted using the RoboCup soccer simulator as the experimental environment. The experimental results showed that the proposed method could obtain a ranking model that reproduces the existing evaluation function. As a result of training and evaluation with various training data sizes, about 100,000 data are sufficient for the ball-chasing task in simulated soccer. The results of experiments to reduce the number of features suggested that some features may be unnecessary.

In future work, we must incorporate the obtained model into actual player agents and evaluate its performance in actual games. It is then also necessary to develop a mechanism to modify the model not only by imitation but also by external instructions.

References

1. Hidehisa Akiyama, Itsuki Noda, "Multi-Agent Positioning Mechanism in the Dynamic Environment", *RoboCup 2007: Robot Soccer World Cup XI*, pp. 377–384, 2008.
2. Hidehisa Akiyama, Tomoharu Nakashima, Shigeto Aramaki: "Online Cooperative Behavior Planning using a Tree Search Method in the RoboCup Soccer Simulation", *Proc of INCos2012*, 2012.
3. Hidehisa Akiyama, Masashi Tsuji, Shigeto Aramaki: "Learning Evaluation Function for Decision Making of Soccer Agents Using Learning to Rank", *Proc. of SCIS & ISIS 2016*, 2016.
4. Hidehisa Akiyama, Masashi Fukuyado, Toshihiro Gochou, Shigeto Aramaki, "Learning Evaluation Function for RoboCup Soccer Simulation using Humans' Choice", *Proc. of SCIS & ISIS 2018*, 2018.

5. Hidehisa Akiyama, Tomoharu Nakashima: "HELIOS Base: An Open Source Package for the RoboCup Soccer 2D Simulation", *RoboCup 2013: Robot World Cup XVII. Lecture Notes in Computer Science, vol 8371. Springer, Berlin, Heidelberg*, pp.528–535, 2014.
6. Hidehisa Akiyama, Tomoharu Nakashima, "HELIOS2012: RoboCup 2012 Soccer Simulation 2D League Champion", *RoboCup 2012: Robot Soccer World Cup XVI*, pp. 13–19, 2013.
7. Hidehisa Akiyama, Tomoharu Nakashima, "HELIOS2018: RoboCup 2018 Soccer Simulation 2D League Champion", *RoboCup 2018: Robot Soccer World Cup XVII*, pp. 18–22, 2018.
8. Masaki Yamaguchi, Ryota Kuga, Hiroki Omori, Takuya Fukushima, Tomoharu Nakashima, Hidehisa Akiyama, "HELIOS2021: Team Description Paper", *RoboCup 2021*, 2021.
9. Hidehisa Akiyama, Tomoharu Nakashima, Kyo Hatakeyama, Takumi Fujikawa, "HELIOS2022: RoboCup 2022 Soccer Simulation 2D Competition Champion", *RoboCup 2022: Robot Soccer World Cup XVIII*, in appear, 2023.
10. Christopher J.C. Burges: "From RankNet to LambdaRank to LambdaMART: An Overview", *Microsoft Research Technical Report, MSR-TR-2010-82*, 2010.
11. Takuya Fukushima, Tomoharu Nakashima, Hidehisa Akiyama, "Similarity Analysis of Action Trajectories Based on Kick Distributions", *RoboCup 2019: Robot World Cup XXIII*, pp. 58–70, 2019.
12. Takuya Fukushima, Tomoharu Nakashima, Hidehisa Akiyama, "Evaluation-function modeling with multi-layered perceptron for RoboCup soccer 2D simulation", *Artificial Life and Robotics*, Vol. 25, issue 3, pp.440–445, 2020.
13. Kyo Hatakeyama, Takuya Fukushima, Yoshifumi Kusunoki, Tomoharu Nakashima, Hidehisa Akiyama, "A Study on the Effect of Team Names on the Team Strategy", *Proc. of RoboCup Asai-Pacific 2021 Symposium*, 2021.
14. Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu : "LightGBM: A highly efficient gradient boosting decision tree", *Advances in Neural Information Processing Systems*, pp.3147–3155, 2017.
15. Ryota Kuga, Yudai Suzuki, Tomoharu Nakashima, "An Automatic Team Evaluation System for RoboCup Soccer Simulation 2D", *2020 Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems (SCIS-ISIS)*, pp. 1–4, 2020.
16. Thorsten Joachims: "Optimizing Search Engines Using Clickthrough Data", *Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02*, pp. 133-142, 2002.
17. Itsuki Noda, Hitoshi Matsubara: "Soccer Server and Researches on Multi-Agent Systems", *Proc. of IROS-96 Workshop on RoboCup*, pp.1–7, 1996.
18. M. Riedmiller, T. Gabel, R. Hafner and S. Lange: "Reinforcement Learning for Robot Soccer", *Autonomous Robots*, 27(1), pp.55-–74, Springer, 2009.
19. Yudai Suzuki, Takuya Fukushima, Lea Thibout, Tomoharu Nakashima, Hidehisa Akiyama, "Game-Watching Should be More Entertaining: Real-Time Application of Field-Situation Prediction to a Soccer Monitor", *RoboCup 2019: Robot World Cup XXIII*, pp 439–447, 2019.
20. Jiarun Zhong, Tomoharu Nakashima, Hidehisa Akiyama, "A Study on the Analysis of Soccer Games Using Distributed Representation of Actions and Players", *ICIC Express Letters*, Vol. 13, No. 4, pp. 303–310, 2019.