Classifying the strategies of an opponent team based on a sequence of actions in the RoboCup SSL

Yusuke Adachi, Masahide Ito and Tadashi Naruse

Graduate School of Information Science and Technology,
Aichi Prefectural University

E-Mail Addresses: im161001@cis.aichi-pu.ac.jp,
{masa-ito, naruse}@ist.aichi-pu.ac.jp

Abstract. In this paper, we propose a new method for classifying the strategies of an opponent in the RoboCup soccer Small-Size League. Each strategy generates a sequence of basic actions selected from a kick action, a mark action, or other similar actions. Here, we identify strategies by classifying an observed sequences of basic actions selected by an opponent during a game. This method greatly improves our previous method[9] in the following two ways: the previous method was applicable mainly to set plays, whereas this restriction is lifted in our new method. Additionally, our new method requires a lower computational time than the previous method. Assuming that our team was the opponent team, our team's strategies were evaluated using the Rand Index, yielding a value exceeding 0.877 in 3 out of 4 games. A Rand index value exceeding 0.840 was obtained from an analysis of the 4 opponent teams (1 game for each opponent team). These Rand indices represent a high level of classification algorithm performance.

1 Introduction

The strategies used in the RoboCup Small Size League (SSL) have been extensively developed in recent years so that each team's robots take action in response to an opponent's predicted behavior. It has become increasingly important for teams to learn about their opponent's behavior. Some studies have developed approaches to learning an opponent's strategies in the SSL[2,9]; however, because these methods use robot trajectory data and require long computational times, they are mainly applied to set plays.

To overcome these problems, we propose a new method for classifying an opponent's strategies. We focus on a sequence of basic actions, or simply a sequence of actions, wherein the basic action is a 4-tuple defined as the <action name, start position, end position, duration>. A typical action is a kick action, a pass action, a shoot action, or other similar actions. Sequences of actions are clustered into several groups such that each group includes a sequences of actions derived from a strategy. The advantage of this method derives from the ease

with which this method predicts a future subsequent action, making it possible to take preemptive counter actions.

In the following sections, we describe a method of extracting robot actions and applying a clustering method by defining a dissimilarity measure of a sequence of actions. Finally, we provide experimental results and discuss the availability of the method.

2 Related work

Erdogan et al.[2] proposed a method for classifying an opponent's behaviors in the SSL, and they applied their classification method to the attacking behaviors in set plays during real SSL games. The opponent's behaviors were expressed as trajectories of offensive robots to which they applied a cluster analysis by computing the similarity of the behaviors. Yasui et al.[9] also proposed a method of classifying an opponent's behaviors using an approach similar to that of Erdogan. Yasui et al. applied their method to learn their opponent's behaviors during set plays as they occurred online and in real time. They demonstrated experimentally that an opponent's behaviors could be classified about 2 seconds before ball actuation. These studies demonstrated the effectiveness of learning an opponent's behaviors; however, because these methods use robot trajectory data and require significant computational times, they are mainly applied to set plays.

Trevizan et al.[6] proposed a method for comparing the strategies of two teams in the SSL. They divided a time series representing a game into non-overlapping intervals that they labeled episodes. They used 23 variables, including the distance between a robot and the ball and the distance between a robot and the defense goal, to characterize the episode. They used the mean and standard deviation of each variable over an episode to reduce the data size. Therefore, n episodes with f variables could be represented using a matrix of size $2f \times n$. They computed the matrix norms of two episode matrices for teams A and B and evaluated the similarities between the strategies of teams A and B. Their method then compared the similarities between the two teams' strategies. Their study's objective differed from the objective addressed in this paper.

Visser et al.[7] proposed a method of classifying an opponent's behaviors based on a decision tree constructed for use in the RoboCup simulation league. Time series data, consisting of the ball-keeper distance, ball speed, number of defenders in the penalty area, and other game parameters, were used to construct a decision tree that predicted the goalkeeper's (GK's) movements, including GK stays in goal, GK leaves goal, GK returns to goal, over several games. Learning based on a decision tree is a type of supervised learning. We propose an unsupervised learning algorithm for use in on-line real-time learning.

3 Robot action detection

In this paper, we use data logged during previous RoboCup competitions. The logged data comprise a time series of robot positions and orientations, ball positions, referee commands, and other game parameters, logged every 1/60 seconds.

The strategies were classified by defining the following 8 actions: passer robot mark, shooter robot mark, ball-keeping robot mark, pass wait, kick ball, kick shoot, kick pass, and kick clear. A time series of logged data was converted to a sequence of these actions for use as an input to our classification process. The not available (NA) action was suitably inserted if a part of the time series could not be converted to any of the 8 available actions.)

In this section, we describe how the robots' actions were detected using the logged data. The basic method is one that we have proposed in [1]. We extend that method in this section.

3.1 Mark actions

In [1], mark actions consist of three actions: "passer mark", "shooter mark", and "ball-keeping robot mark". We improved the detection algorithms described in [1] and describe some of these improvements in the following subsections.

Passer mark In his passer mark algorithm, Asano did not consider whether a passing robot surely existed. In his algorithm, the passer mark was detected, even if a robot simply ran after the ball. We corrected this fault as follows.

Definition of symbols
$\overrightarrow{T_{i,f}}$: the position of the teammate robot T_i at time f.
$\overrightarrow{O_{j,f}}$: the position of the opponent robot O_j at time f.
$\overrightarrow{B_f}$: the position of the ball at time f.
$\overrightarrow{T_{S,f}}$: the position of the teammate robot with the shortest distance to the line connecting the ball $\overrightarrow{B_f}$ and the robot $\overrightarrow{T_{i,f}}$. We considered $T_{S,f}$ to be the receiver robot.

We computed the distance $D_{j,i,f}$ between $O_{j,f}$ and the line connecting $T_{i,f}$ and $T_{S,f}$, as shown in Figure 1. If either or both of the inner products $\overrightarrow{V_1} \cdot \overrightarrow{V_2}$ and $\overrightarrow{V_4} \cdot \overrightarrow{V_5}$ were negative, γ_p was added to $D_{j,i,f}$, where γ_p is a constant, because it was desirable to exclude non-mark cases (See Eq.(1)). Averaging the $D_{j,i,f}$s over the interval $[f, f+n-1]$ gave the following equation, and with it we judged whether or not O_j marked a passer robot. (O_j marked the passer T_i at time f if the $MarkPass_{j,i,f}$ variable were equal to 1,)

$$MarkPass_{j,i,f} = \begin{cases} 1, & if \ \ \frac{1}{n} \sum_{k=f}^{f+n-1} D_{j,i,k} \leq TH_p \\ 0, & otherwise \end{cases} \tag{1}$$

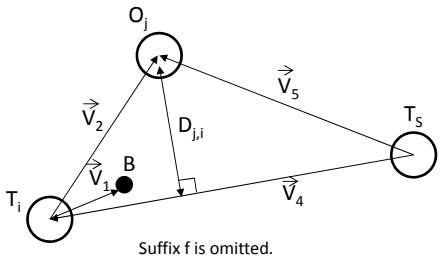

Fig. 1. Passer mark.

where TH_p is a given threshold and n is a given constant[1].

The detection algorithm is given below.

```
/*Passer mark*/
1   for(f = 0; f < f_end; f++)
2     for(j = 0; j < 6; j++)
3       for(i = 0; i < 6; i++) {
4         compute D_{j,i,f} and MarkPass_{j,i,f}
5         memorize MarkPass_{j,i,f} }
```

Shooter mark and ball-keeping robot mark

A shooter mark is often carried out near the goal area, and a mark robot usually stands some distance from a shooter. As an evaluation metric, we defined the distance between a (mark) robot and a line connecting the shooter and the center of the goal mouth. We then computed the $MarkShoot_{j,i,f}$ variable using an equation[2] similar to Eq. (1). (For a $MarkShoot_{j,i,f}$ variable of 1, O_j marks the shooter T_i at time f.)

A ball-keeping robot mark is a mark other than the passer mark and the shooter mark. We used, as an evaluation metric, the weighted sum of two distances, that is, the distance $D1_{j,i,f}$ between the ball-keeping robot T_i and the (mark) robot O_j and the distance $D2_{j,i,f}$ between the (mark) robot O_j and a line connecting the ball-keeping robot T_i and the ball.

$$D_{j,i,f} = \alpha D1_{j,i,f} + \beta D2_{j,i,f} \tag{2}$$

We then computed the $MarkBall_{j,i,f}$ variable using the similarity equation[3], Eq. (1). (For a $MarkBall_{j,i,f}$ variable equal to 1, O_j marked the ball-keeping robot T_i at time f.)

[1] We used $TH_p = 400mm$ and $n = 3$ in our experiments.

[2] In the equation, the threshold is denoted by TH_s, and we used $TH_s = 400mm$ in our experiments.

[3] In the equation, the threshold is denoted by TH_b, and we used $TH_b = 800mm$ in our experiments. Both α and β in Eq. (2) were set to the value 0.5 .

3.2 Pass waiting action

The pass waiting action was not discussed in [1]. We defined it here for the first time.

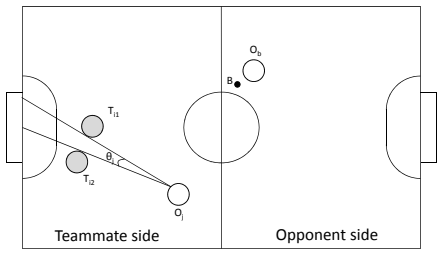

Fig. 2. Pass waiting action.

Let O_b be the opponent robot nearest to the ball. It was reasonable to assume that a candidate robot waiting to receive a pass was the one on the left side of O_b, as shown in Figure 2. O_j in Fig. 2 is one such candidate. The shootable angle θ_j could then be computed. If an opponent robot with a shootable angle exceeding a given threshold were present, the opponent was defined as being in a pass waiting action. To reduce the influence of noise, the shootable angle was averaged over some interval in time. The variable $WaitPass_{j,f}$ is given by

$$WaitPass_{j,f} = \begin{cases} 1 & if \quad \frac{1}{n} \sum_{k=f}^{f+n-1} \theta_{j,k} \geq TH_w \\ 0 & Otherwise, \end{cases} \tag{3}$$

where $\theta_{j,k}$ is the shootable angle of robot O_j at time k. As the threshold value of TH_w, we used a threshold of 8 degrees (= 0.14rad), and the length of the interval n was 3 in our experiments.

3.3 Kick actions

In RoboCup Soccer, the kick actions as well as the mark actions are important. We proposed a kick action detection algorithm using the logged data reported in [1] and its modification in [8]. For the purposes of this paper, we classified kick actions according to the kick purpose: kick for shoot, kick for pass, or kick for clear. A kick action that did not belong to any of these three purposes was also considered. This section describes the kick action detection algorithm.

Definition of symbols
Kick actions = {KickShoot, KickPass, KickClear, KickBall}. KickBall is a kick

action other than one of the first three actions.
L_b: a line segment that begins at the kick point P_s and ends at the last point P_e, along which the ball's trajectory is straight. Let $\overrightarrow{P_b}$ be its vector form.
$\overrightarrow{P_{oi}}$: a vector beginning at the kick point and ending at the opponent's robot O_i.
P_{gl}, P_{gr}, P_{gc}: edge points and center point of the teammate goal mouth.
d, D_G: a distance between P_e and P_{gc} and a given threshold.

The following algorithm predicts a kick action based on the location of the end point L_b.

/*Kick action classification*/
1 **if** L_b *crosses side line* **then** *kick is KickClear*
2 **else if** L_b *crosses teammate goal line* **then** *kick is KickShoot*
3 **else if** $\left|\overrightarrow{P_{oi}} \times \overrightarrow{P_b}\right| < D_1$ **then** *kick is KickPass*
4 **else if** P_e *is in* $\triangle P_s P_{gl} P_{gr}$ *and* $d < D_G$ **then** *kick is KickShoot*
5 **otherwise** *kick is KickBall*

In line 3 of the above algorithm, $\left|\overrightarrow{P_{oi}} \times \overrightarrow{P_b}\right| < D_1$ computes how close P_{oi} is to the line segment L_b.

Finally, an action other than any of the above 8 actions was expediently classified as a NA action.

4 Action decision algorithm

The previous section described the action detection algorithm. Next, a sequence of actions was calculated for each opponent robot. Multiple actions could be predicted simultaneously for a robot. In this case, we selected an action according to the priority of the action, where the priority of a kick action was the highest, followed by a pass waiting action, and finally by a mark action. The time series of logged data was then converted to a time series of actions[4] and was finally converted into a sequence of actions:

$$A_P[n] = \left[\begin{pmatrix} action_{n1} \\ \overrightarrow{p_{sn1}} \\ \overrightarrow{p_{en1}} \\ frame_{n1} \end{pmatrix}, \cdots, \begin{pmatrix} action_{ni} \\ \overrightarrow{p_{si1}} \\ \overrightarrow{p_{eni}} \\ frame_{ni} \end{pmatrix}, \cdots, \begin{pmatrix} action_{nt} \\ \overrightarrow{p_{snt}} \\ \overrightarrow{p_{ent}} \\ frame_{nt} \end{pmatrix} \right], \tag{4}$$

where $A_P[n]$ is a sequence of actions for robot n, $\overrightarrow{p_{si1}}$ and $\overrightarrow{p_{eni}}$ are the start and end times of the $action_{ni}$, respectively, and $frame_{ni}$ is the duration of $action_{ni}$. The ith element of a sequence of actions is denoted by

$$A_P[n][i] = \begin{pmatrix} action_{ni} \\ \overrightarrow{p_{sni}} \\ \overrightarrow{p_{eni}} \\ frame_{ni} \end{pmatrix} \tag{5}$$

[4] This series of actions is defined over each time frame.

A note on the time series of actions

Any time series of actions usually contains false actions. Preprocessing is needed to remove such actions.

- An action that only continues over a couple of frames should be classified as a false action and, as a result, is replaced by the succeeding action. The kick actions are an exception because short kick actions can occur at the edge of the field.
- If an action is broken into two actions by a false action, the two actions should be unified into a single action.
- If a false action cannot be replaced by any of the 8 actions, the time series is padded with an NA action.

5 Dissimilarities between the action sequences

We defined a dissimilarity metric of two sequences of actions using Eq. (4). To do so, we first defined a dissimilarity measure d_0 of two actions $A_{P_1}[n_1][t_1]$ and $A_{P_2}[n_2][t_2]$ as follows,

$$
\begin{aligned}
&d_0\ (\ A_{P_1}[n_1][t_1], A_{P_2}[n_2][t_2]) = \\
&\quad \begin{cases}
\alpha \cdot \text{frame\_diff} + \beta \cdot \text{p\_dist} + \gamma \cdot \text{diff\_size\_cost} \\
\qquad \text{if } \mathit{action}_{n_1 t_1} = \mathit{action}_{n_2 t_2} \\
\alpha \cdot 2.0 + \beta \cdot \text{p\_dist} + \gamma \cdot \text{diff\_size\_cost} \\
\qquad \text{if } (\mathit{action}_{n_1 t_1} \in \text{Kick}, \mathit{action}_{n_2 t_2} \notin \text{Kick})\ \mathit{or} \\
\qquad (\mathit{action}_{n_1 t_1} \notin \text{Kick}, \mathit{action}_{n_2 t_2} \in \text{Kick}) \\
\alpha \cdot 1.0 + \beta \cdot \text{p\_dist} + \gamma \cdot \text{diff\_size\_cost} \\
\qquad \mathit{otherwise}
\end{cases}
\end{aligned}
\tag{6}
$$

where α, β, γ are the weights, and frame_diff, p_dist, and diff_size_cost are explained in the following paragraph.

The value of frame_diff is given by the following equation,

$$
\text{frame\_diff} = \left| \frac{\mathit{frame}_{n_1 t_1}}{\mathit{frame\_play}_{n_1 t_1}} - \frac{\mathit{frame}_{n_2 t_2}}{\mathit{frame\_play}_{n_2 t_2}} \right|, \tag{7}
$$

where $\mathit{frame\_play}$ is the duration of the sequence of play $A_P[n]$. Frame_diff takes a value between 0 and 1.

The value of p_dist is given by the following equation,

$$
\text{p\_dist} = \min\left\{ \frac{|(\overrightarrow{p_{sn_1 t_1}} - \overrightarrow{p_{sn_2 t_2}})|}{\mathit{FieldLength}}, 1.0 \right\} + \min\left\{ \frac{|(\overrightarrow{p_{en_1 t_1}} - \overrightarrow{p_{en_2 t_2}})|}{\mathit{FieldLength}}, 1.0 \right\}, \tag{8}
$$

where $\mathit{FieldLength}$ is the length of the side line of the field. P_dist takes a value between 0 and 2.

The diff_size_cost is given by the following equation,

$$\text{diff\_size\_cost} = \min\left\{\frac{1}{3}\left(\frac{\text{long\_size}}{\text{short\_size}} - 1.0\right), 2.0\right\}, \tag{9}$$

where long_size= $\max(frame_{n_1t_1}, frame_{n_2t_2})$ and short_size= $\min(frame_{n_1t_1}, frame_{n_2t_2})$. Diff_size_cost takes a value between 0 and 2.

Next, we defined a dissimilarity $d_1(A_{p1}[n_1], A_{p2}[n_2])$ between two sequences of actions $A_{P_1}[n_1]$ and $A_{P_2}[n_2]$ of robots n_1 and n_2.

Action sequences do not always have the same length; therefore, we defined the dissimilarity as the degree of overlap between the shorter sequence of actions and the longer sequence of actions. The computational algorithm is given below.

Step 1 Let *short* be the shorter sequence, and *long* be the longer sequence. Let the length of *short* and *long* be *short_size* and *long_size*, respectively. Let *kick_num* be the number of kick actions in *long*. Let i and j be counter variables with an initial value of 1. Let *start_j* and *limit_j* be the start of the search pointer and the end of the search pointer, with an initial value of 1. Initialize d_1 to 0.

Step 2 For the ith action in *short* sequence, decide the search range in the *long* sequence as follows,

$$\begin{aligned} ls &= \text{long\_size}/\text{short\_size} \\ limit\_j_1 &= i + ls \\ limit\_j_2 &= \min(start\_j + ls, \text{long\_size}) \\ limit\_j &= \max(limit\_j_1, limit\_j_2). \end{aligned} \tag{10}$$

For the ith action in the *short* sequence, search coincident actions over the range *start_j* and *limit_j* within the *long* sequence.

Step 3 If a coincident action is found, compute

$$d_1 = d_1 + d_0(A_{P_1}[n_1][i], A_{P_2}[n_2][j]),$$

and $start\_j = j + 1$. If such an action is not found, compute

$$d_1 = d_1 + d_0(A_{P_1}[n_1][i], A_{P_2}[n_2][i]).$$

If $i <$ short_size, then $i = i + 1$, and go to Step 2; otherwise, go to Step 4.

Step 4 Out of *kick_num* kick actions in *long* sequence, remove actions that match the kick action in *short* sequence. Let the number of remaining kick actions be *kick_unused*. Add *kick_unused* to d_1 as an additional cost,

$$d_1 = d_1 + kick\_unused.$$

Finally, we defined a dissimilarity d_2 between plays. A play includes six robot action sequences, so we considered the correspondence between any 2 sequences.

The dissimilarity d_2 was defined by

$$d_2(A_{P_1}, A_{P_2}) = \min_{\sigma \in S_6} \{\mathrm{Tr}(FP_\sigma)\} \tag{11}$$

$$F = [f_{ij}] \tag{12}$$

$$f_{ij} = \{d_1(A_{P_1}[i], A_{P_2}[j])\}, \tag{13}$$

where P_σ is a permutation matrix and Tr(A) is the trace of matrix A.

The team's behavior was classified using the group average method[3] to cluster the sequences of actions under the dissimilarity metric d_2.

6 Deciding on the number of clusters

Determining the number of clusters was important. If the range of the number of clusters was given in advance, we could use the Davies–Bouldin index[4]. By contrast, Yasui et al. proposed a method for deciding the number of clusters independently of the range[10]. Their method is given by the following procedure. First, compute

$$W(K) = \sum_{i=1}^{K} \sum_{X_k \in C_i} \sum_{X_l \in C_i} d_2(A_{P_k}, A_{P_l}). \tag{14}$$

This equation computes the sum of the distances for any two elements in a cluster, summated over all clusters, assuming that the number of clusters is K. Then, using $W(K)$, compute

$$W'(K) = W(K)/W(1), \tag{15}$$

and

$$\arg\max_{1 \le K \le N} (W'(K) \le h), \tag{16}$$

where h is a threshold value determined in advance. The number of clusters is decided by Eq. (16).

7 Experiment: our team's strategy classification

In RoboCup 2015, we competed in 4 official games and recorded logged data from each game. These data were then used in a classification experiment, assuming that our team was the opponent. In the experiment, we used $\alpha = \beta = \gamma = 1/3$ in Eq. (6) and $h = 0.06$ in Eq. (16)[5].

In this section, we classified our team's strategies experimentally. The set play data were used in the experiment. A set play began at ball re-placement and ended at ball interception or ball-out-of-field. The clustering results obtained from the 4 games[6] are shown in Figures 3−6.

[5] For the parameter h, we ran the program over the range $0.03 - 0.07$ and found that $h = 0.06$ gave the best results.

[6] The number of clusters was not known in advance in this experiment, so the k-means method could not be used. Ward's method and the group average clustering

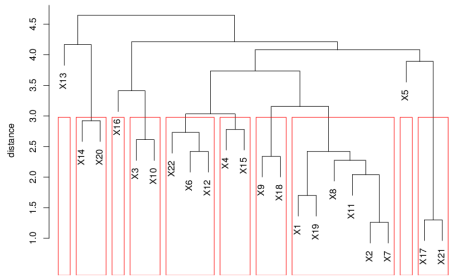

Fig. 3. Dendrogram for match No.1.

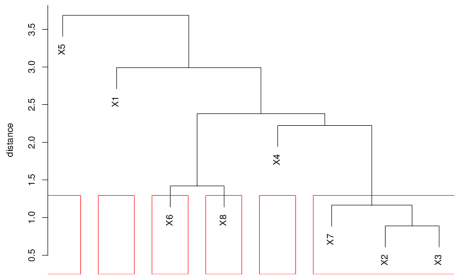

Fig. 4. Dendrogram for match No.2.

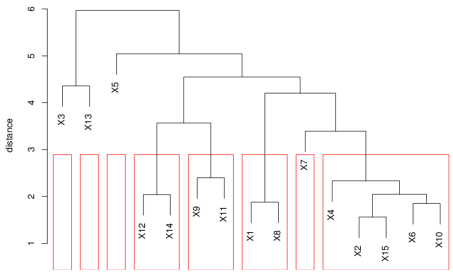

Fig. 5. Dendrogram for match No.3.

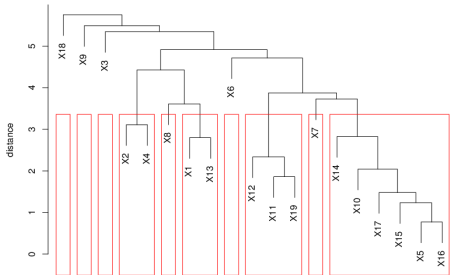

Fig. 6. Dendrogram for match No.4.

Table 1. Rand Index (RoboDragons)

	No.1	No.2	No.3	No.4
Rand Index	0.892	0.750	0.924	0.877

Table 2. Rand Index (opponents).

	No.1	No.2	No.3	No.4
Rand Index	0.901	0.889	0.874	0.840

The Rand index[5] was used to evaluate the classification results. We determined the correct classification for each game, as determined in comparison with the clustering results obtained by inspection (the human clustering method). The Rand index for each game is given in Table 1. The Rand index values were high for each game except for the No. 2 game. In the No. 2 game, the opponent team malfunctioned so that the detection of mark actions did not work well, resulting in a lower Rand index. In other games, a cluster identified by human clustering was found to be divided into two clusters by the computer clustering method. This lowered the Rand index slightly; however, from a practical perspective, this was not a serious problem.

apply under circumstances of an unknown number of clusters. In our experiment, these approaches gave similar clustering results. The computational cost of the group average clustering was lower than the cost associated with Ward's method; therefore, we used the group average clustering.

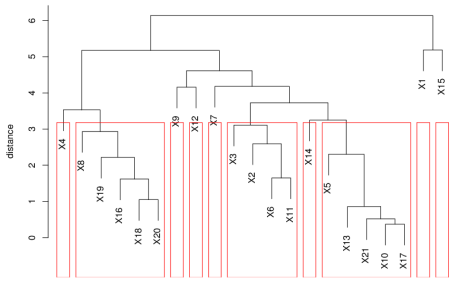

Fig. 7. Dendrogram for match No.1 (opponent).

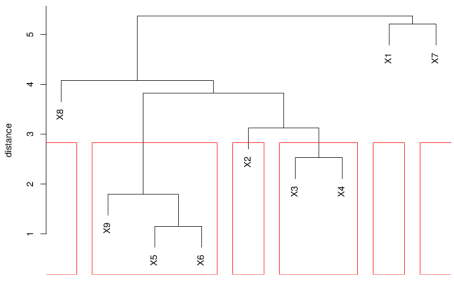

Fig. 8. Dendrogram for match No.2 (opponent).

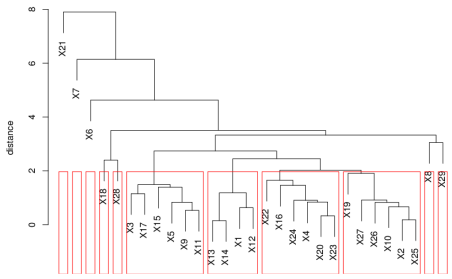

Fig. 9. Dendrogram for match No.3 (opponent).

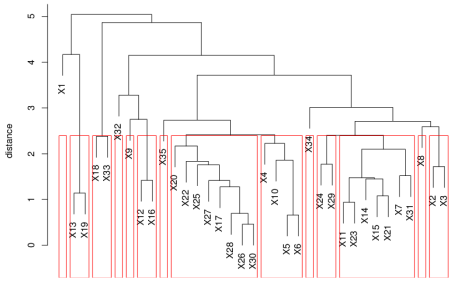

Fig. 10. Dendrogram for match No.4 (opponent).

8 Experiment: opponent team classification

The strategies of each opponent team in the RoboDragons' official games were classified. Figures 7−10 provide the classification results, and Table 2 lists the Rand indices. Table 2 reveals that the Rand indices assumed values between 0.840 and 0.901. For comparison, Erdogan et al. obtained values between 0.87 and 0.96 by using the trajectory data. Our experimental results revealed that high Rand index values similar to Erdogan's results were obtained from the action sequence data. This experiment revealed the cluster division problem discussed in previous sections. Future work to improve this problem is necessary.

9 Computational time

The total clustering analysis computational time was measured for game No. 4, in which 35 set plays were executed. This calculation included the computational time associated with the preprocessing of a time series of actions, the creation of a distance matrix, and the clustering using the group average method. We got an average computational time of 0.67 msec and a maximal computational

time of 1.82 msec, which show that the real-time computation of clustering is possible.

10 Concluding remarks

We have proposed a learning (classification) method based on an opponent's actions in this paper. A sequence of actions was derived from a time series of data logged from an SSL game. A sequence of actions includes less data than the logged data, permitting faster computation. An evaluation of this method using the Rand index revealed that clustering using the proposed method provided a good classification of an opponent team's behaviors (strategies, in most cases). The computational time was so small that real-time computation was possible using this method.

Future work will focus on refinements of the proposed method, extensions to any scene during play, the generation of a counter action using the logged data of past games, and implementation to our RoboDragons system.

References

1. K. Asano, K. Murakami and T. Naruse, "Detection of basic behaviors in logged data in RoboCup Small Size League", RoboCup 2008: Robot Soccer World Cup XII, LNCS 5399 pp439-450, Springer, 2009
2. C. Erdogan, and M. Veloso, "Action Selection via Learning Behavior Patterns in Multi-Robot Domains," In Proceedings of International Joint Conference on Artificial Intelligence 2011, pp.192-197, 2011
3. B.S.Everitt et al. "Cluster Analysis 5th Edition" , Wiley, 2011
4. David L. Davies and Donald W. Bouldin, "A Cluster Separation Measure" , IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI- 1(2), pp.224-227, 1979
5. W. M. Rand,"Objective criteria for the evaluation of clustering methods",Journal of the American Statistical Association (American Statistical Association) 66 (336): 846-850,1971
6. F.W. Trevizan, and M.M. Veloso, "Learning Opponent's Strategies In the RoboCup Small Size League," In Proceedings of AAMAS'10 Workshop on Agents in Real-time and Dynamic Environments, 2010
7. U. Visser, and H. Weland, "Using Online Learning to Analyze the Opponent's Behavior," RoboCup 2002: Robot Soccer World Cup VI, LNAI 2752, pp.78-93, Springer, 2003
8. K. Yasui et al. "A new detection method of kick actions from logged data of SSL games", JSAI Technical Report SIG-Challenge-B201-6, 2012 (In Japanese)
9. K. Yasui, K. Kobayashi, K. Murakami and T. Naruse, "Analyzing and Learning an Opponent's Strategies in the RoboCup Small Size League ", RoboCup 2013: Robot Soccer World Cup XVII, LNCS 8371 pp159-170, Springer, 2014
10. K. Yasui, M. Ito and T. Naruse, "Classifying an Opponent's Behaviors for Real-time Learning in the RoboCup Small Size League", IEICE Trans. on Info. and Systems, Vol. J97-D, No. 8, pp.1297 - 1306, 2014 (In Japanese)