20th RoboCup International Symposium, Leipzig, Germany, July 2016.

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

Yusuke Adachi, Masahide Ito and Tadashi Naruse

Faculty 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 to classify strategies of an opponent used in the RoboCup soccer small size league. It is based on a sequence of basic actions, where the basic action is a kick action, a mark action and so on. This method greatly improves our previous method[9] in the following two points: it frees a restriction that the previous method can mainly be applicable to set plays, and it reduces the computing time. Evaluating the classification result by the Rand Index, we got the value 0.877 or more for classification of our team's strategies, considering our team as the opponent team, in 3 out of 4 games and the value 0.840 or more for 4 opponent teams (1 game for each opponent team), which achieved a high performance.

1 Introduction

In recent years, the strategies used in the RoboCup Small Size League (SSL) have been highly developed in order for each team's robots to take action to predict an opponent's behavior. This means that it has become increasingly necessary to learn more about an opponent's behavior. There are some studies of learning an opponent's strategies in SSL[9,2]. However, these are applicable mainly to the set plays since they use the trajectory data of robots and need longer computation time.

In this paper, to overcome the above problems, we propose a new method to classify the opponent's strategies. We focus on a sequence of basic actions, or simply a sequence of actions, where the basic action is a 4-tuple <action name, a start position, an end position, duration>. Typical action is a kick action, a pass action, a shoot action and so on. Sequences of actions are clustered into several groups in which each group would have the sequences of actions derived from a strategy. The advantage of the method is an easiness of predicting the next action to come, so that it can be possible to take counter actions easily.

In the following sections, we describe an extraction method of robot actions, then clustering method by defining a dissimilarity measure of a sequence of actions. Finally, we show an experimental results and discuss the availability of the method.

2 Related work

Erdogan et al.[2] proposed a method to classify opponent's behaviors in SSL and classified attacking behaviors in set plays in real SSL games. Expressing the opponent's behavior as the trajectories of offensive robots, they make a cluster analysis by computing the similarity of behaviors. Yasui et al.[9] also proposed a classification method of opponent's behaviors in a similar way to Erdogan. Yasui et al. apply their method to the on-line real time learning of an opponent's behaviors in set plays, and experimentally shows that an opponent's behavior can be classifiable about 2 seconds before the ball actuation. These studies show the effectiveness of learning an opponent's behaviors, however, these methods are mainly applied to set plays and they need too much computation time.

Trevizan et al.[6] proposed a method to compare the strategies of two teams in SSL. They divide a time series representing a game into non-overlapping intervals which they call episodes. They use 23 variables, such as the distance between a robot and the ball and the distance between a robot and the defense goal, to characterize the episode, however, they use the mean and standard deviation of each variable over an episode to reduce data size. Therefore, n episodes with f variables can be represented by the matrix of the size $2f \times n$. They compute a matrix norm of two episode matrices for team A and B and evaluate the similarity of the strategies between team A and B. Their method compares how similar two teams' strategies are. Their study's objective is different from the issues of this paper.

Visser et al.[7] proposed a classification method of an opponent's behaviors based on the decision tree in the RoboCup simulation league. Using the time series data consisting of ball - keeper distance, ball speed, the number of defenders in the penalty area, etc., they made a decision tree to analyze the goalkeeper's movements — GK stays in goal, GK leaves goal, GK returned to goal — for several games. The learning based on the decision tree is a supervised learning. We would propose an unsupervised learning, as we are aimed at on-line learning.

3 Robots' action detection

In this paper, we use logged data which had logged in previous RoboCup competitions. The logged data is a time series data consisting of positions and orientations of robots, position of the ball, referee command and so on, which is logged every 1/60 seconds.

To classify the strategies, we define 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 is converted to a sequence of these actions, so that it is an input for our classifying process. (NA (not available) action is suitably inserted if a part of the time series cannot convert to any of 8 available actions.)

In this section, we describe how robots' actions are detected from the logged data. The basic method is the one we proposed in [1]. We extend the 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] a bit and describe some of them in the following subsections.

passer mark Asano doesn’t consider, in his passer mark algorithm, whether a passing robot surely exists. In his algorithm, the passer mark is detected even if a robot simply runs after the ball. We correct the 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 that has the shortest distance to the line connecting the ball $\overrightarrow{B_f}$ and the robot $\overrightarrow{T_{i,f}}$. We consider $T_{S,f}$ as the receiver robot.

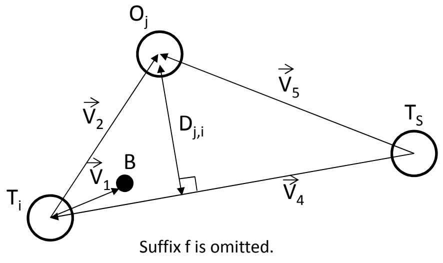

Fig. 1. Passer mark

We compute a 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 inner products $\overrightarrow{V_1} \cdot \overrightarrow{V_2}$ and $\overrightarrow{V_4} \cdot \overrightarrow{V_5}$ is/are negative, γ_p is added to $D_{j,i,f}$, where γ_p is a given constant, since we would exclude the non-mark case. (See Eq.(1).) Averaging $D_{j,i,f}$s over the interval $[f, f+n-1]$ gives the following equation, and with it we judge whether O_j marks a passer robot or not. (When $MarkPass_{j,i,f}$ variable is 1, O_j marks the passer T_i at time f.)

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

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 a bit far from a shooter. So we use, as an evaluation metric, a distance between a (mark) robot and a line connecting the shooter and the center of the goal mouth. Then, we compute $MarkShoot_{j,i,f}$ variable by using the similar equation [2] as Eq. (1). (When $MarkShoot_{j,i,f}$ variable is 1, O_j marks the shooter T_i at time f.)

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

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

Then, we compute $MarkBall_{j,i,f}$ variable by using the similar equation [3] as Eq. (1). (When $MarkBall_{j,i,f}$ variable is 1, O_j marks the ball keeping robot T_i at time f.)

3.2 Pass Waiting action

Pass waiting action is not discussed in [1]. We newly define it here.

Let O_b be a opponent robot that is nearest to the ball. It is reasonable to say that a candidate robot waiting to receive a pass is the one stood to the left side of O_b in Figure 2. O_j in Fig. 2 is one of the candidates. For such opponent robots, compute an angle θ_j that is a shootable angle. If there is an opponent robot which has a shootable angle greater than given threshold, we define the opponent is

[1] We used $TH_p = 400mm$ and $n = 3$ for our experiment.

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

[3] In the equation, threshold is denoted by TH_b and we used $TH_b = 800mm$ in our experiment. Also, for both α and β in Eq. (2), we used the value 0.5 .

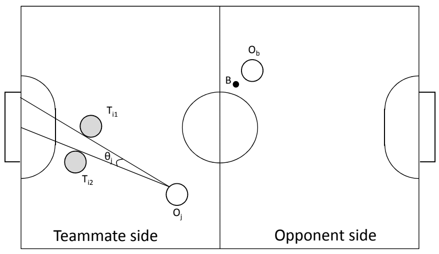

Fig. 2. Pass waiting action

in a pass waiting action. To reduce the influence of noise, the shootable angle is averaged over some interval. The variable $WaitPass_{j,f}$ is given by

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

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

3.3 Kick actions

In RoboCup Soccer, the kick actions are important as well as the mark actions. We proposed a detection algorithm of kick action from logged data in [1] and its modification in [8]. We use the algorithm basically for kick detection. However, for the purpose of this paper, we would classify kick actions according to the kick purpose, i.e. kick for shoot, for pass and for clear. The kick action not belonging to these three purposes is also considered. In this section, we describe kick actions detection algorithm.

[Definition of symbols]
Kick actions = {KickShoot, KickPass, KickClear, KickBall}. KickBall is a kick action other than first three actions.
L_b : a line segment that begins from the kick point P_s and ends at the last point P_e that the ball goes straight, and let $\overrightarrow{P_b}$ be its vector form.
$\overrightarrow{P_{oi}}$: a vector beginning from the kick point and ending in opponent robot O_i
P_{gl}, P_{gr} : edge points of teammate goal mouth

Following algorithm decides the kick action based on the location of the end point of 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 $\triangle P_s P_{gl} P_{gr}$ then kick is KickShoot
5        otherwise kick is KikBall
```

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 which is not any of above 8 actions is expediently classified as NA action.

4 Action decision algorithm

In the previous section, we described the action detection algorithm. Next, for each opponent robot, a sequence of actions is calculated. Generally, multiple actions may be detected at the same time for the robot. In that case we select one action according to the priority of action, where the priority of kick action is highest, then pass waiting action, and mark action is lowest. As a result, a time series of logged data is converted to a time series of actions[4] and finally is converted to 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 a start and an end time of $action_{ni}$, respectively, and $frame_{ni}$ is a duration of $action_{ni}$. The ith element of a sequence of actions is denoted by

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

A note on a time series of actions

A time series of actions usually contain false actions. To remove such actions, a preprocessing is necessary.

[4] This is a series of actions given every time frame.

– An action which only continues for a couple of frames should deal with false action, and as a result will be replaced by the succeeding action. However, a kick action is an exception since a short kick action may happen at the edge of the field.
– If an action breaks into two actions by a false action, they should be unified into one action.
– If false action cannot be replaced by any of 8 actions, NA action is padded.

5 Dissimilarity of action sequence

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

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

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

The frame_diff is given by following equation,

$$\text{frame\_diff} = \left| \frac{frame_{n_1 t_1}}{frame\_play_{n_1 t_1}} - \frac{frame_{n_2 t_2}}{frame\_play_{n_2 t_2}} \right| \tag{7}$$

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

The p_distance is given by following equation,

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

where $FieldLength$ is the length of the side line of the field. P_distance takes a value between 0 and 2.

The diff_size_cost is given by 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 define 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.

Since action sequences are not always have the same length, we define the dissimilarity as how much the shorter sequence of actions coincides with the longer sequence of actions. Computation algorithm is given below.

Step1 Let *short* be shorter sequence, and *long* be 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 the *long*. Let i and j be counter variables with initial value 1. Let *start_j* and *limit_j* be the start of search pointer and the end of search pointer with initial value 1. Initialize d_1 to 0.

Step2 For 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 ith action in *short* sequence, search coincident action in the rage *start_j* and *limit_j* of *long* sequence.

Step3 If 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 action isn't 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 Step2, otherwise go to Step4.

Step4 Out of *kick_num* kick actions in *long* sequence, remove actions that matched with 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 define a dissimilarity d_2 between plays. A play has six robots' action sequences, so we must consider correspondence between any 2 sequences. The dissimilarity d_2 is 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 a trace of matrix A.

To classify team's behavior, we use the group average method[3] to cluster the sequences of actions under the dissimilarity metric d_2.

6 Decision of the number of clusters

To decide the number of clusters is important. If the range of the number of clusters is given in advance, we can use Davies-Bouldin index[4]. On the contrary, Yasui et al. proposed a method to decide the number of clusters not depending on the range[10]. It 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 adds up the sum of the distance of two elements in a cluster for 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 given in advance. The number of clusters is decided by Eq. (16).

7 Experiment: our team's strategies classification

In RoboCup 2015, we competed 4 official games and recorded logged data for the games. Using them, we did the classification experiment assuming our team as 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 classify our team's strategies by experiment. In the experiment, we used set play data. A set play begins at ball re-placement and ends at ball interception or ball out of field. For the 4 games, the results of clustering[6] are shown in Figures 3 $\sim$ 6.

Rand index[5] is used to evaluate the classification results. We give the correct classification for each game which is made by the first author of this paper. Rand index for each game is shown in Table 1. Rand index shows high value for each game except No. 2 game. For No. 2 game, by the opponent team's malfunction, the detection of mark action didn't work well so that it lowered the Rand index. For other games, it happened that a cluster given by human clustering was divided into two clusters in computer clustering. This lowered Rand index a bit. However, in practical use, it would not be a serious problem.

[5] For the parameter h, we ran the program in the range of 0.03 to 0.07 and $h = 0.06$ gave the best result.

[6] The number of clusters is not known in advance in this experiment, so that k-means method cannot use. Ward's method and the group average clustering work under the unknown number of clusters. In our experiment, they gave similar clustering results. The computational cost of the group average clustering is lower than the one of Ward's method. Therefore, we used the group average clustering.

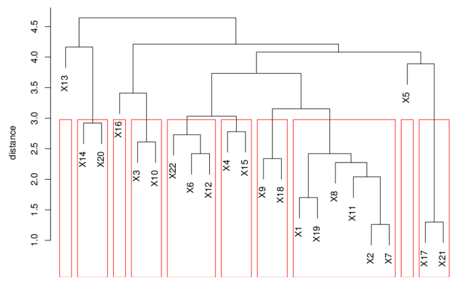

Fig. 3. Dendrogram for match No.1

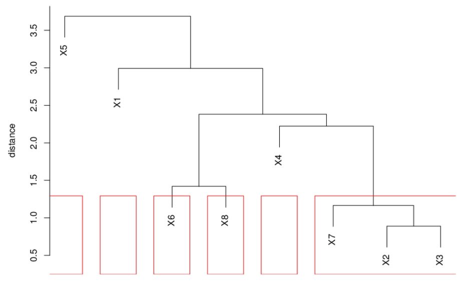

Fig. 4. Dendrogram for match No.2

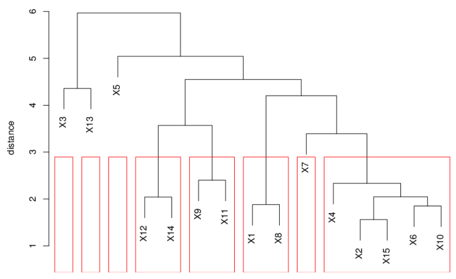

Fig. 5. Dendrogram for match No.3

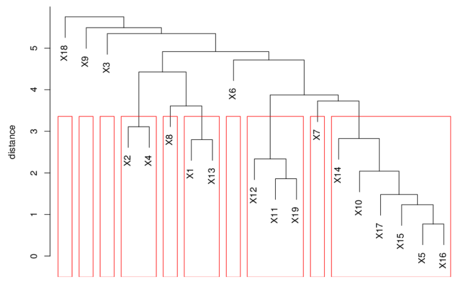

Fig. 6. Dendrogram for match No.4

Table 1. Rand Index (RoboDragons) (computer clustering vs. human clustering)

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

8 Experiment: opponent team's classification

For each opponent team in RoboDragons' official games, we classified the strategies of the team. Figures 7 $\sim$ 10 show the classification results and Table 2 shows Rand index. Table 2 shows that the Rand index takes the value between 0.840 and 0.901. Though Erdogan et al. got the value between 0.87 and 0.96, they used the trajectory data. Our experimental results show that high Rand index value close to Erdogan's results is gotten from the action sequences data. In the experiment, cluster division problem discussed in previous section occurred again. Future work on improving this problem is necessary.

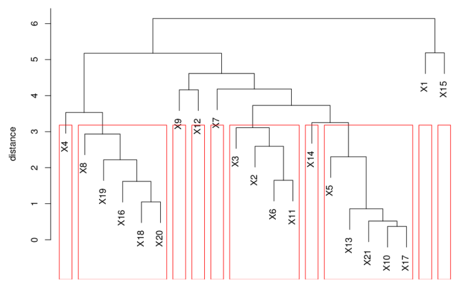

Fig. 7. Dendrogram for match No.1 (opponent)

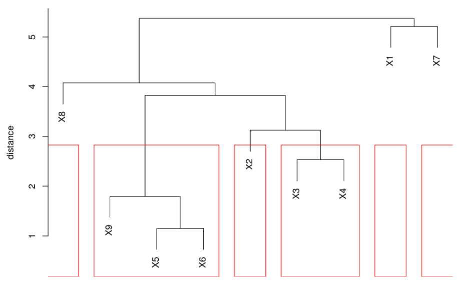

Fig. 8. Dendrogram for match No.2 (opponent)

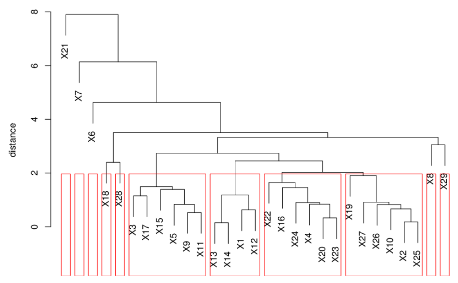

Fig. 9. Dendrogram for match No.3 (opponent)

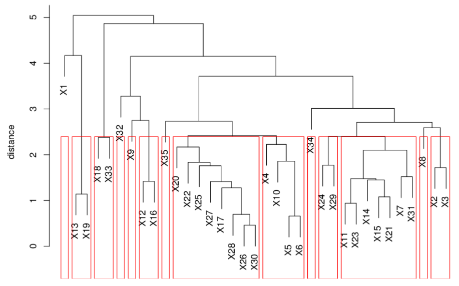

Fig. 10. Dendrogram for match No.4 (opponent)

Table 2. Rand Index (opponents)

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

9 Computation time

For the game No. 4 in which 35 set plays are executed, we measured the total computation time of the clustering[7]. It includes the computation time of the preprocessing of a time series of actions, the creation of distance matrix and the clustering by group average method. Table 3 shows the result. From Table 3, it is clear that the real time computation of clustering is possible.

[7] Since the clustering is done incrementally, when i'th set play is executed, i set plays are clustered.

Table 3. clustering time(AMD A10 7800)

	average time(ms)	max time(ms)
No.4(35 setplays)	0.67	1.82

10 Concluding remarks

We have proposed a learning (classification) method based on opponent's actions in this paper. A sequence of actions is derived from a time series of logged data of an SSL game. A sequence of actions has less data than logged data so that faster computation can be expected. Evaluation by Rand index shows that clustering by the proposed method gives good classification result of opponent team's behaviors (strategies in most cases). Computation time is so small that real time computation is possible enough.

Future work is refinement of the proposed method, extension to any scene in in-play, generation of 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)