Behavior analysis of executed and attacked players in Werewolf game by ILP

Ema Nishizaki and Tomonobu Ozaki

Graduate School of Integrated Basic Sciences, Nihon University
3-25-40 Sakurajosui, Setagaya-ku, Tokyo 156-8550, Japan
m6116m13@educ.chs.nihon-u.ac.jp, tozaki@chs.nihon-u.ac.jp

Abstract. Recently, the Werewolf game, one of multiplayer incomplete information games, is recognized as a promising research testbed for artificial intelligence. In this work-in-progress paper, in order to obtain useful knowledge on the Werewolf games, log analyses of human players in the Werewolf BBS are conducted. By applying inductive logic programming to the log data of six games in the BBS, we attempt to extract classification rules on being attacked, executed and voted, whose bodies compose predicates representing players' past behaviors. In preliminary experiments, classification rules capturing characteristic behaviors among multiple agents were successfully obtained.

Keywords: Werewolf game, behavior analysis

1 Introduction

The Werewolf game is one of conversation-based role-playing party games. The game consists of two teams, werewolf team and villager team. Only werewolves can recognize their teammates, while villagers are given no information to which team each player belongs. Some villagers have a special role and they can obtain additional information by their own ability. Such information heterogeneity is one of main characteristics of the game. In the game, villagers try to find and execute all werewolves, while werewolves try to deceive and eliminate villagers by attacking them. The Werewolf game has two phases, day and night. These two phases are iterated during the game. Players discuss to find out werewolves or to deceive villagers in day phase. At the end of day phase, an executed player is decided by majority voting. In night phase, werewolves select a villager and attack him/her. Executed or attacked players get kicked out of the game.

The Werewolf game is recognized as a promising research testbed for artificial intelligence. Several research on the game have been reported recently. For example, a method of game logs analysis based on attunements and rebuttals is proposed to extract useful knowledge for the strategy construction[1]. In [2], multimodal analysis was applied to the Werewolf games to detect deceptive roles. A role estimation model is proposed based on the text analysis in [3].

In this work-in-progress paper, in order to obtain useful knowledge for building intelligent software agents for the Werewolf game, log analyses of human players are conducted by using inductive logic programming.

2 Werewolf BBS

The Werewolf BBS[1] is an online community website for the Werewolf game where a text-based chat system is provided for playing the games. The Werewolf BBS is a real time communication system. One day in the Werewolf game corresponds to one day in the real world. Players can communicate each other by text only. No voice and video messages can be exchanged.

All conversations by players are recorded during the game. There are four types of log data in the Werewolf BBS. We focus on the "white logs" which all players can browse as a target of our analysis.

3 Analysis of Werewolf BBS using ILP

3.1 Selection of Games

We prepare six standard and average games for the analysis. Three of them are the games werewolf teams won and the rests are the games villager teams won. Fifteen players having enough experience participate in each game. While werewolf teams consist of three werewolves and one lunatic, villagers teams have eleven players including three villagers having special roles. Three games the werewolf team won have eight days to the end. They contain 1234.0 records on average. The rest three games the villager team won have seven days and contain 1166.6 records on average.

3.2 Background Knowledge

We prepare twelve predicates to represent players' behaviors by considering the Werewolf protocol[4]. The predicates are listed: 'comingout' (a coming-out of the role), 'estimate' (an estimate of other player's role), 'divined' (a report of the divination), 'inquested' (a report of the inquest), 'guarded' (a report of the guard), 'question' (a player's question to other players), 'answer' (an answer of question by a player), 'agree' (a player's agreement to other players), 'disagree' (a player's disagreement to other players), 'line' (an estimation of that two players belong to the same team), 'unline' (an estimation of that two players belong to different team), and 'disrelation' (a backstabbing within werewolf teams). All predicates have at least three arguments *Game*, *Day*, *Player* for representing that a player *Player* takes the corresponding action on the *Day*th day in a game *Game*. The original log data which contains the contents of conversation among players written in a natural language is converted into a set of facts on the prepared predicates manually. Note that all players may tell a lie. For example, a werewolf may behave like a seer to deceive players in the villager team. The converted facts represent what each player says regardless of whether the utterance is a lie or not.

[1] `http://www.wolfg.x0.com/`

In addition to the facts, we prepare twelve intensional rules for handling behaviors in previous days as background knowledge in ILP. All rules have the form of

```
Pred( Game:Day, N, Player, Args··· ) :-
    prev_days( N ), PDay is Day - N,
    Pred( Game:PDay, Player, Args···).
```

and represent that a player `Player` took an action *Pred* `N` days ago from `Day`th day in a game `Game`. The predicate `prev_days/1` is an auxiliary predicate defined as `prev_days(N):- member(N, [0,1,···,X])` where `X` is a parameter to control the maximum time difference. We set `X` to 3 in the experiments. The predicate *Pred* is instantiated by twelve predicates for the facts. For example, by using 'comingout', we obtain a rule for the past (and current) behavior on comingout as

```
comingout( Game:Day, N, Player, Role) :-
    prev_days( N ), PDay is Day - N,
    comingout( Game:PDay, Player, Role ).
```

where an argument `Role` shows the role a player `Player` claims. The rule which captures past behaviors on 'line' is shown below.

```
line( Game:Day, N, Player, A, B ) :-
    prev_days( N ), PDay is Day - N,
    line( Game:PDay, Player, A, B ).
```

This rule states that a player `Player` estimated that two players `A` and `B` belong to the same team `N` days ago from `Day`th day in a game `Game`.

3.3 Positive and Negative Examples

In this work-in-progress paper, three classification tasks are treated.

The first task is to derive classification rules which characterize the behaviors of executed players. A predicate `executed( Game:Day, Player )` is used for this task, which represents a player `Player` was executed on the `Day`th day in a game `Game`. From the log data, 39 positive examples and 528 negative ones are extracted.

As a second task, we try to obtain classification rules for the behaviors of attacked players. We prepare a predicate `attacked( Game:Day, Player )` which indicates that the werewolf team succeeded in attacking a player `Player` on the `Day`th day in a game `Game`. Note that, while executed players are selected from all participants by voting, attacked players must be members of villager team or a lunatic. In addition, hunters sometimes guard the villagers from the attack by werewolves. Thus, positive examples for this task are less than those in the first task. Only 27 positive examples and 505 negative ones are used.

The third task is to obtain rules on a voting behavior. A predicate `voted( Game:Day, Player )` states that a player `Player` got a vote on the `Day`th day in a game `Game`. In total, the log data contains 89 positive and 478 negative examples, respectively.

4 Results

An inductive logic programming engine Aleph[2] is employed to solve the classification problems shown in the previous section. We used three commands for theory construction (induce/0, induce_cover/0 and induce_max/0) in order to extract classification rules as many as possible. In the following subsections, the results of each classification task are described.

4.1 Execution

The Aleph system extracts 28 classification rules on the execution. An example of derived rule is shown below. The corresponding graphical representations is shown in Fig.1(a). In the figure, each node corresponds to a player. Directed edges represent predicates in the rule body and they show relationships among agents. A number n in the parenthesis means that the corresponding behavior to the edge is observed n days before from when the action in the rule head occurred.

```
executed( Game:Day, X ) :-
    estimate( Game:Day, 0, Y, X, wolf ),
    estimate( Game:Day, 1, X, Y, not(wolf) ),
    agree( Game:Day, 2, X, Z ).
```

This rule says that: (1)a player `Y` believes that a player `X` is a werewolf on the execution day, and (2)the executed player `X` estimated that a player `Y` is not a werewolf one day ago, and (3)he/she agreed with a player `Z` two days ago. We believe that this rule captures an appropriate situation in selecting players to be executed because the executed player `X` is suspected as a werewolf by other player `Y`.

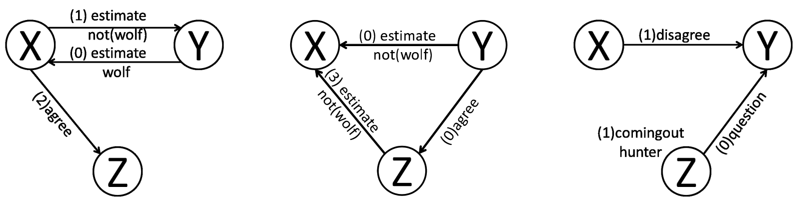
This rule can be instantiated by substituting two villagers for `X` and `Z` and a lunatic or a werewolf for `Y` in plural games irrespective of the winning team. In addition, `X` is never instantiated by a player in the werewolf team. We can infer from the above situation that this rule shows the behavior of a villager targeted by the werewolf team.

4.2 Attack

We obtain 23 classification rules on "being attacked" by the werewolf team. Within the 23 rules, 15 rules have predicates question, answer, agree or disagree, which show communications with other players. An example of obtained rules, depicted in Fig.1(b), is explained below.

```
attacked( Game:Day, X ) :-
    estimate( Game:Day, 3, Z, X, not(wolf) ),
    estimate( Game:Day, 0, Y, X, not(wolf) ),
    agree( Game:Day, 0, Y, Z ).
```

[2] `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph`



(a) Execution (b) Attack (c) Vote

Fig. 1. Graphical representations of extracted rules

This rule shows the following situation : (1)a player `Z` estimated that an attacked player `X` is not a werewolf three days before, (2)a player `Y` also believes that `X` is not a werewolf and agree with `Z` on the attacked day. We can easily imagine a basic strategy of the werewolf team from the rule. The team selects a player `X` to be attacked, who is less likely to be executed by villagers because `X` is not suspected by other two players `Y` and `Z`.

4.3 Vote

In total, we obtain 53 classification rules showing the behavior of players who get a vote. An example is shown graphically in Fig.1(c) and explained below.

```
voted( Game:Day, X ) :-
    disagree( Game:Day, 1, X, Y ),
    question( Game:Day, 0, Z, Y ),
    comingout( Game:Day, 1, Z, hunter ).
```

This rule says that : (1)a player `Z`, who claimed that he/she is a hunter one day ago, asks a player `Y` some questions, and (2)a player `X`, who disagreed with the player `Y` one day ago, got a vote. By applying this rule to the log data, we confirm that this rule fits well into the players getting a few votes.

5 Conclusion and Future Direction

In this work-in-progress paper, by applying inductive logic programming to three classification problems, we succeeded in obtaining rules which capture characteristic behaviors among multiple players in the Werewolf BBS.

As one of future works, we plan to incorporate certain predicates representing each player's view and intention. In the Werewolf game, players having different roles have different information. For example, wolves can recognize teammates, but others cannot. Seer players can perceive spurious seers easily because other player must not be a seer. By utilizing these heterogeneities effectively and by preparing certain mechanisms to estimate other players intention, we can expect to develop sophisticated and accurate classification rules for each role. For this purpose, we plan to employ the framework of answer set programming and its induction algorithm[5] as well as meta-interpretive learning[6] in higher order logic. Since players behaviors are not deterministic, another promising research direction is uncertainty handling. We also investigate to utilize probabilistic logic programs such as PRISM[7] to build probabilistic models for players behaviors.

Acknowledgements We heartily thank Mr. Ninjin for allowing us to use the log data in the Werewolf BBS. We have deep regards to Professor Fujio Toriumi at the University of Tokyo for providing us the Werewolf databases. A part of this work was supported by JSPS KAKENHI Grant Number JP26330262.

References

1. Michimasa Inaba, Fujio Toriumi, Hirotaka Osawa, Daisuke Katagami, Kosuke Shinoda and junji Nishino: Werewolf Game Analysis based on Attunements and Rebuttals, *The 28th Annual Conference of the Japanese Society for Artificial Intelligence*, 2014 (in Japanese)
2. Gokul Chittaranjan and Hayley Hung: Are You A Werewolf? Detecting Deceptive Roles and Outcomes in a Conversational Role-Playing Game, *Proc. of the 2010 IEEE Internation Conference on Acoustics, Speech and Signal Processing*, pp.53345337, 2010
3. Masaki Sakamoto, Atsushi Ueno and Tomohito Takubo: A Method for Estimating Roles in the Werewolf Game Based on Dialogue Data from a Game BBS, *IPSJ SIG Technical Report*, Vol.2016-GI-35, No.12, 2016 (in Japanese)
4. Hirotaka Osawa: Communication Protocol for the "Werewolf" game, *Human-Agent Interaction Symposium*, 2013 (in Japanese)
5. Mark Law, Alessandra Russo and Krysia Broda: Inductive Learning of Answer Set Programs, *Proc. of the 14th European Conference on Logics in Artificial Intelligence*, pp.311-325, 2014.
6. Stephen H. Muggleton, Dianhuan Lin and Alireza Tamaddoni-Nezhad: Meta-interpretive Learning of Higher-order Dyadic Datalog: Predicate Invention Revisited, *Journal of Machine Learning*,Volume 100, Issue 1, pp.49-73, 2015.
7. Taisuke Sato and Yoshitaka Kameya: New advances in logic-based probabilistic modeling by PRISM, *In Probabilistic Inductive Logic Programming*, LNCS 4911, Springer, pp.118155, 2008.