Description of Team TOT

OTSUKI Takashi

August 14th, 2021

1 Overview

TOT is basically a C# port of my Java agent[1] which won the second place last year. The changes from last year are some bug fixes and tuned hyperparameters of DNNs for role estimation. This agent is built using AIWolfLib[2] , AIWolfServer[3] and OpenCvSharp[4].

2 Features for role estimation

2.1 Basic features[Otsuki 21]

Basic feature vector concerning agent a is represented by $\boldsymbol{f_e}(a)$, and consists of following features.

- Life: 1 if the agent is dead, otherwise 0.
- The number of human judgements by professed seers/mediums on the agent.
- The number of werewolf judgements by professed seers/mediums on the agent.
- The number of human judgements by the agent.
- The number of werewolf judgements by the agent.
- The number of times the vote is different from the vote declaration.
- IsEnemy: 1 if the agent seems to be an enemy, otherwise 0.
- IsAlly: 1 if the agent seems to be an ally, otherwise 0.
- IsHuman: 1 if the agent is a human, otherwise 0.
- IsWerewolf: 1 if the agent is a werewolf, otherwise 0.
- IsExecuted: 1 if the agent was executed, otherwise 0.
- IsKilled: 1 if the agent was killed by werewolves, otherwise 0.
- A 6-dimentional one-hot vector representing the role declared by the agent.

[1]https://github.com/AIWolfSharp/otsuki2020
[2]https://github.com/AIWolfSharp/AIWolfLib
[3]https://github.com/AIWolfSharp/AIWolfServer
[4]https://github.com/shimat/opencvsharp

Table 1: Types of utterance considered

Name	Description
Skip	Watch the situation without utterance
Vote	Declaration of vote
Estimate	Role estimation
Co	Comingout
Divined	Report of divination
Identified	Report of identification of the executed agent
Operator	Operator sentence (e.g. compound sentence)

2.2 Utterance pattern features

In addition to agent a's basic feature vector $\boldsymbol{f_e}(a)$, TOT uses a's utterance pattern feature vector $\boldsymbol{f_u}(a, role)$ to take into account the characteristics of a's utterances when a is in $role$. $\boldsymbol{f_u}(a, role)$ is composed as follows.

- Agent a's relative utterance frequency vector when a is in $role$ is represented by $\boldsymbol{r}(a, role) = (r_{0S}, r_{0V}, \cdots, r_{0O}, r_{1S}, \cdots, r_{9O})$, where letting r_{tu} $(0 \leq t \leq 9, u \in \{Skip, \cdots, Operator\})$ be the relative frequency of a's utterance u in turn t on the first day.
- Agent a's utterance observation vector is represented by $\boldsymbol{o}(a) = (o_{0S}, o_{0V}, \cdots, o_{9O})$, where $o_{tu} \in \{0, 1\}$ represents the existence of a's u in t on the first day.
- Agent a's utterance pattern feature vector related to $role$ is represented by 70 dimensioal vector $\boldsymbol{f_u}(a, role) = (r_{0S}o_{0S}, \cdots, r_{9O}o_{9O})$.

3 Role estimation using DNN

3.1 Input vector of DNN

The input vector of DNN for estimating agent he's role is composed of the following features.

(1) date.

(2) 6-dimentional one-hot vector representing the role of the estimator me.

(3) $\boldsymbol{f_e}(me)$.

(4) $\boldsymbol{f_e}(he)$.

(5) Basic feature vector for all agents except me and he.

(6) he's utterance pattern feature vectors for all roles.

The number of dimensions of the input vector is 517 in case of 5-people village and 697 in case of 15-people village.

3.2 Structure of DNN

The structure of DNN used is 4-layers, fully-connected. It has two hidden layers having hidden units with ReLU activation function, and the output layer has one output unit having logistic sigmoid activation function. The tuning of hyperparameters is carried out based on SMBO (Sequential Model-based Global Optimization)[Bergstra 11].

3.3 Data

The training and the test of DNN above is carried out using data extracted from the game logs of test competition 2020 final. The downloaded game logs concering team “otsuki” contain 61,300 logs for 5-people village, and 25,600 logs for 15-people village.

4 Algorithm of agent

4.1 15-people village

VILLAGER agent’s voting algorithm is based on team “takeda”, which won the first place two years ago. If more than half of the agents have declared their votes, VILLAGER votes for the agent having most the score calculated taking into account the number of votes and the likelihood of WEREWOLF on the agent. Otherwise, it votes for the agent having most the score calculated taking into account the likelihood of WEREWOLF and POSSESSED on the agent, wheather the agent is WEREWOLF or HUMAN, and whether the agent tells a lie.

In addition to the same algorithm as VILLAGER, BODYGUARD agent guards the agent having most the score calculated taking into account the likelihood of each role on the agent, the agent’s claimed role, whether the agent tells a lie and the win rate of the agent.

In addition to the same algorithm as VILLAGER, MEDIUM agent reports the result of its identification every morning after finding a WEREWOLF or a fake MEDIUM, or after the third day.

SEER agent divines the agent having the highest WEREWOLF probability among the agents whose species is unidentified. It reports the result of its divination every morning after finding a WEREWOLF or a fake SEER, or after the third day. In case that the number of alive agents is three and a POSSESSED is found, it says ”I’m a WEREWOLF” to counter PP. It votes for the agent having most the score calculated taking into account the likelihood of WEREWOLF and POSSESSED on the agent, wheather the agent is WEREWOLF or HUMAN, and whether the agent tells a lie.

POSSESSED agent has an inner SEER agent which has game information reflecting the fake divination, and basically acts as this inner SEER. In fake divination, the least WEREWOLF likely agent is judged as a WEREWOLF on the first and the third days, the most WEREWOLF likely agent is judged as a HUMAN on other days. In case that the number of alive agents is three, it carries out PP saying ”I’m a WEREWOLF”.

WEREWOLF agent attacks the agent having most the score calculated taking into account the likelihood of each role on the agent, and the win rate of the agent. It has an inner player acting as the fake role decided at the beginning of the game, in coordination with other werewolves, and basically talks as this inner player. If more than half of the agents have declared their votes, it votes for the agent having most the score calculated taking into account the likelihood of WEREWOLF on the agent from the view of the inner player. Otherwise, it votes for the same agent as the inner player's vote target. The inner SEER and the inner MEDIUM randomly judge on the target's species. In case that the number of alive agents is three and POSSESSED is found, it carries out PP saying ”I'm a WEREWOLF”.

4.2 5-people village

Agent for 5-people village evaluates all possible correspondence between agent and role based on the role probability given by DNN to find the candidate agent for the role specified.

VILLAGER agent votes for one agent from the WEREWOLF candidates. When PP occurs on the second day, it says ”I'm a WEREWOLF” to counter PP.

SEER agent divines one agent randomly selected from other agents every day. It says ”I'm a SEER” immediately after the game starts, then honestly reports the result of its divination every morning. It votes for one agent from the WEREWOLF candidates. When PP occurs on the second day, it says ”I'm a WEREWOLF” to counter PP.

POSSESSED agent has an inner SEER agent which has game information reflecting the fake divination, and basically acts as this inner SEER. It says ”I'm a SEER” immediately after the game starts. It regards one agent from WEREWOLF candidates as a WEREWOLF, then reports that this WEREWOLF is HUMAN.

WEREWOLF agent has an inner VILLAGER agent and basically acts as this inner VILLAGER on the first day. It votes for the least POSSESSED likely agent on the second day. It attacks the least POSSESSED likely agent every day. When the other agent says ”I'm a WEREWOLF” or ”I'm a POSSESSED” on the second day, it says ”I'm a WEREWOLF” to carry out PP.

References

[Otsuki 21] Otsuki: Effect of meta-inference using utterance patterns in role estimation of AI-Wolf, Proceedings of the Annual Conference of JSAI (2021)

[Bergstra 11] Bergstra, Bardenet, Bengio, Kegl: Algorithms for hyper-parameter optimization, Advances in Neural Information Processing Systems (2011).