CITS 3001 Algorithms, Agents and Artificial Intelligence

Miao Hui Su

22535578

Word Count: 2349

Literature Review

“The Resistance” is a modern, hidden identities board game for five to ten players where teams are elected to participate in up to five, highly simplified missions. Mission success is simply a matter of each player on the team choosing it, but a small number of players are in fact spies who seek to sabotage these missions.[1]

The Resistance is a game model based on incomplete information. Usually, we use machine learning, reinforcement learning and other means to solve this problem, in addition to monte Carlo trees and pruning algorithms to search The game's state space.[3] However, for The Resistance game, monte Carlo tree is difficult to estimate the possibilities without knowing the opponent information.[2] However, Bayes' theorem can update the probability when the information is constantly iterated, so I choose to use Bayes' theorem to solve the probability estimation problem of The Resistance.[4]

Bayes' theorem is more from the perspective of the observer. The randomness of the event is only caused by the incomplete information grasped by the observer. The amount of information grasped by the observer will affect the observer's cognition of the event. Bayes' theorem can be briefly described as the following formula:

$$P(A|B) = \frac{P(A) * P(B|A)}{P(B)}$$

P (A | B) happened in B occurs under the condition of A probability.[4] For The Resistance game, I want to use a simple example to explain how bayes' formula can be used to calculate the probability of a player being a spy. Suppose we start a game with five players and two of them are spies. There is a Resistance player whose index in the player array is 0. At the beginning of the game, he will set the probability of other players being spies to 0.5. In the first round of the game, Mission contains players 1 and 2, and a spy's Betray occurs. He will calculate the probability that each player is a spy in this situation. In the case of two spies among the remaining four players, the possible events for spies are as follows:

Spies	Resistance
1、2	3、4
1、3	2、4
1、4	2、3
2、3	1、4
2、4	1、3
3、4	1、2

In the table above, there are four options for what happens at this point that Spies were (1,3), (1,4), (2,3) and (2,4), so P(B) = $\frac{4}{6}$. If Player 1 was a spy, corresponding schemes were (1,2), (1,3) and (1,4). At this point, Only (1,3), (1,4) satisfy the betray of a spy. So P (B | player 1 is spy) = $\frac{2}{3}$, so we get the contain in the misson player 1 and 2, and produced a spy betray the probability of no. 1 player in this situation is a spy.

Rationale

For games with incomplete information like “The Resistance”, we cannot simply determine whether a player is a spy or not by preset parameters. As The rounds of the game increase, we will comprehensively determine whether a player is a spy based on the performance of players in different rounds. In the course of the game, we will constantly acquire new information, Bayes'

theorem can be a good solution to such a model that constantly measures the probability of events based on new information.

In “The Resistance”, players may be assigned to two roles: spy or Resistance, and models created using Bayes' theorem work well with both. For defenders, calculated according to the Bayes' theorem can choose mission is put forward for spies spy probability low probability of players so that the success of the mission possible, at the same time, the need to vote for a mission, the defenders can also according to the player's spy probability of mission to consider whether should agree to the mission. In this model, we assume that all players use the same Bayes' theorem to predict the spy probability strategy, thus allowing spies to choose missions and betray missions based on their own and their teammates' spy probability. Specific operational strategies for resisters and spies will be described later in the article.

Implementation

In this model, I create an agent called MyAgent that implements the Agent interface and defines some of its own private methods to make Bayes' theorem work. I will describe the details of the individual member variables and methods below.

First I'll introduce MyAgent's member variables, which store the data needed to run the game.

name	type	modifier	Introduction
isSpy	boolean	private	Stores whether an agent is a spy.
id	int	private	Stores the index of this agent in the player array.
SpyProbability	Double []	private	Stores the spy probability of the player at the corresponding index position.
name	String	private	The name of this agent
playersCnt	int	private	Store the number of players in the game.
spiesCnt	int	private	Save the number of spies in this game.
SpiesSet	HashSet<Integer>	private	Save the people identified as spies in the game.
spyNum	int []	private static final	How many spies are stored for a given number of players.
AgentNum	int	private static	Stores how many MyAgents there are currently.

Next, I'll cover MyAgent's main approach, which will show you how I built a bayes' theorem based game AI model.

- GetUniqueName:
  - Param: There is no parameter.
  - Return: The name of the Agent calling the method.
  - Modifier: private static
  - Introduction: Gets a unique name in the local game based on the current number of MyAgents.
- getName:
  - Param: There is no parameter.
  - Return: The name of the Agent calling the method.

- Modifier: public
- Introduction: Gets the current Agent name.

- newGame:
  - Param:
    - numPlayers : int. The number of players in this game.
    - playerIndex : int. The current MyAgent index among game players.
    - spies : int[]. An array that stores indexes for all spy players. Empty array if the player is a Resistance.
  - Return: void
  - Modifier: public
  - Introduction:Start a new game. In this method, member variables are initialized and each player's initial spy probability is assigned a value.
- MemeberInList
  - Param:
    - Member : int. Number to be retrieved.
    - memberList : int[]. The array to be retrieved
  - Return: Whether the input element is in the input array.
  - Modifier: private
  - Introduction: Determines whether an element is in an array.
- SelectMinElement:
  - Param:
    - Member : Double[]. An array that stores data to be retrieved.
    - N : int. How many elements to pick.
  - Return: Gets the index of the smallest n elements in the input array.
  - Modifier: private
  - Introduction:Gets the index of the smallest n elements of the input array and stores it in a Set.
- SelectMaxElement:
  - Param:
    - Member : Double[]. An array that stores data to be retrieved.
    - N : int. How many elements to pick.

- Return: Gets the index of the largest n elements in the input array.
- Modifier: private
- Introduction:Gets the index of the largest n elements of the input array and stores it in a Set.

- proposeMission:
  - Param:
    - Teamsize : int. The number of mission forces required.
    - failsRequired : int. Number of failed missions.
  - Return: Array of proposed mission members
  - Introduction:If the player calling this method is a spy, it will select the total number of players not more than spies and the number of spies equal to failsRequired, and then select the player from the list of spies with the least probability of being a spy to join the mission. Then the most likely players of Resistance were selected to join mission to confuse the crowd. If the player calling this method is not a spy, it will have only one strategy, which is to find out which players are least likely to be spies and join the mission.
- Vote:
  - Param:
    - Mission : int[]. Store the players in mission.
    - Leader : int. The player who made the mission.
  - Return: Is this mission approved.
  - Modifier: public
  - Introduction:If the player calling this method is a spy and the leader of the mission is also a spy, then the mission will pass as described in the proposeMission; if not, the method checks whether the number of players in the mission that are spies is less than the total number of players that are spies. If so, Then the mission will pass, otherwise not. If the player calling this method is not a spy, then this method will count the number of players whose probability value is greater than 0.6 and the number of players who are confirmed to be spies. If this number is greater than or equal to the number of players needed to fail, then the mission will not be passed, otherwise the mission will be passed.
- combineElement:
  - Param:
    - IndexList : int[]. Stores the data to be combined.
    - pairCnt : int. The number of elements in a combination.
    - has : int. How many elements are there currently.
    - cur : int. The current selected index.
  - Return: void, but all n combinations that get elements from the input array are stored in member variables.

- Modifier: private
- Introduction:Create combinations of n elements recursively.

- BayesianAnalysis:
  - Param:
    - Mission : int[]. All players on this mission.
    - failCnt : int. Select the number of spies at Betray in current mission.
  - Return: An array that stores the spy probability of each indexed player.
  - Modifier: private
  - Introduction:First, I will according to the players other than my own spiesCnt combination could calculate how many kinds of spy, and then respectively calculated according to the current mission each combination failCnt A failure probability P (T), in the case of A player's spy the current situation of P (T) | A), Finally, based on these two probability and the probability of A player's spy P (A) to calculate P (A | T), and then returns the result. In particular, if there is a spy in the mission and it chooses no betray, then the calculated P(T) may be equal to 0. In this case, I think the information of the game in this round is destroyed, which cannot reflect everyone's spy probability well, so I will choose to give up this round of probability update.

- Betray:
  - Param:
    - Mission : int[]. All players on this mission.
    - Leader : int. The proposer of a character.
  - Return: Whether to betray the mission under the input conditions.
  - Modifier: public
  - Introduction:Since the betray method is only called for spies, if all spies in mission are spies, choose not to betray this mission to reduce the Bayesian probability of your Resistance players. If not all of the spies in mission are spies, then we need to count whether the number of spies reaches the number that causes the mission to fail. If so, we will select the mission betray; otherwise, we will choose the mission not betray.

- missionOutcome
  - Param:
    - Mission : int[]. All players on this mission.
    - Leader : int. The proposer of a character.
    - numFails : int. Number of failed mission.
    - missionSuccess : boolean. Is mission successful
  - Return: void

- Modifier: public
- Introduction: Analyze the mission. If the number of failed missions equals the number of workers, then the mission is full of spies, so there is no need to perform Bayesian analysis, just add everyone in the mission to the SpiesSet member variable. If the number of failures is smaller than the total number of missions, a Bayesian analysis is performed based on the current mission, and the result is added to the player's original probability and divided by two to get the probability that the new player is a spy.

Validation

To verify the Bayes' theorem based game model I created, I replaced a RandomAgent in the game player with the Agent I created, and started the game. I will analyze what happened in a game in the following.

In this game, MyAgent is assigned index positions 2, spy players are assigned index positions 3 and 4, and I'll use player 3 as an example.

Mission	failsCnt	Bayes probability	Spy probability
0、2	0	1.0	0.75
1、3、4	1	0.75	0.75
2、3	0	0	0.375
0、1、4	0	0.75	0.5625
0、2、4	0	1	0.84375

Game over, Resistance wins. As we can see from this game, the Bayes-based probability model has a good effect for the spy player at positive Betray, but not for the player who does not betray. However, regardless of the algorithm, if a spy player chooses not betray from the beginning to the end, he is no different from the rest of the players who are not spies.

The probability model based on Bayes has a good effect on this simplified The Resistance. As for MyAgent, he acquiesces that the players he faces also adopt the same strategy as his own. Therefore, when playing games with RandomAgent, the spies using RandomAgent often fail to identify because they have not performed the betray operation from the beginning to the end. However, in general, MyAgent can solve the spy probability estimation problem in The Resistance games well by using the mathematical model.

References

[1] D. P. Taylor, “Investigating Approaches to AI for Trust-based, Multi-agent Board Games with Imperfect Information; With Don Eskridge's ``The Resistance'',” BscH. dissertation, Dept. Comp. Sci., Univ of Derby., Derby, United Kingdom, 2014.

[2] D. P. Taylor, "AI for The Resistance", *Confect's Codex*. 2014 [Online]. Available: https://dtconfect.wordpress.com/projects/year-4/resistance/. [Accessed: 31- Oct- 2016]

[3] Edward Powley, Daniel Whitehouse, Peter Cowling(2013), Reducing the burden of knowledge: Simulation-based methods in imperfect information games. http://www.aifactory.co.uk/newsletter/2013_01_reduce_burden.htm

[4] K. Cowles, R. Kass and T. O'Hagan, "What is Bayesian Analysis? | International Society for Bayesian Analysis", *International Society for Bayesian Analysis*, 2016. [Online]. Available: https://bayesian.org/Bayes-Explained. [Accessed: 31- Oct- 2016]