Game Artificial Intelligence (CE811): Assignment 1

Rohit Venugopal VENUG87701

School of Computer Science and Electronic Engineering
University of Essex, UK
rv19514@essex.ac.uk

Abstract. During the course of this project, various types of agents were created to play the social deduction card game, "The Resistance". The main idea was to implement some learning mechanism so as to predict a competitor's behaviour. Learning techniques such as Multiple Regression and Decision Tree Classifier were employed to enable the agent to learn and analyze other players. Each of the agents are evaluated by checking their results against other bots, and their overall playability.

1 Introduction

The goal of this assignment was to create an intelligent agent that can play the card game "The Resistance". The Resistance is a deductive reasoning card based game, which is played by two groups; the resistance and the spies. Only the spies know who the other spies are, and the resistance do not. This means, for each mission in the game to succeed, each player has to try and identify another player's faction based on their behaviour. In such an environment, where other players can lie and give out false information, analysing behavior and deductive reasoning can be a challenging task for an intelligent agent.

This report presents three agents; the first agent follows a rule-based approach to playing the game, while the other two agents employ some kind of learning mechanism to analyse and try and predict the behaviour of other competitors based on previous data. In Section 2, various techniques for creating agents for both other games and similar games are discussed. Section 3 deals with the goal behind the project, and the framework of the game. Section 4 and 5 discuss the various techniques used in the by the agents, and how certain methods and techniques were implemented. In Section 6, the performance of the bots is checked against other bots like beginners and intermediates. Section 7 discusses the development process and also analyzes the results of the bots, their accuracy and learning capabilities. In the final section, the report reaches the conclusion, and ideas for further development or future work is discussed.

2 Literature Review

Creating and using Artificial Intelligence for games is nothing new, especially with Deep Blue [4] defeating Chess champion Garry Kasparov in 1997, and more recently Google's AlphaGo [12] defeating the world Go champion Ke Jie. [6] discusses the use

of Artificial Intelligence and Computational Intelligence in games, and how, while lot of intelligent agents do exist to play games, not all of them necessarily employ a learning mechanism to play.

[9] proposed a method to improve the use of AI in Real-time Strategy (RTS) games, so that it behaves more like a human. This is especially important because older techniques for Game AI involved exhaustive searching or brute-force to win,for example as that of [4] and [10].

However, while both Chess and Go have a fixed state space and definite rules for playing the game, social deduction games like The Resistance, depends more on social interaction among the players and more importantly deception and manipulation. Building an AI agent to play such a game is a much more challenging task.

Not much previous work could be found on building agents for The Resistance. But there do exist other similar social deduction games like Werewolf, Mafia and others, for which some work has been done on building an intelligent agent.

[5] tried and used Natural Language Processing (NLP) to build an agent to play the game of Werewolf.

[3] analyzes the use and effectiveness of Monte Carlo Tree Search (MCTS) in a variety of games including Connection Games (Hex), Real-Time Games (Tron), Real-Time Strategy Games (Warcraft) and even Non-Deterministic games (Magic: The Gathering and Scotland Yard). In 2012, AIGameDev held a competition to design bots for the Resistance game, and made them play against each other. [13] analyzes the bots presented at the competition and implemented an opponent modelling agent to try and assign suspicion scores.

[7] implemented a new architecture to play imperfect information games like Werewolf, which is very similar to The Resistance. In this approach, each player has a trustworthiness value and also a threat score, and they are updated based on the past actions of the competitors. The agent can then decide how to play based on these characteristics.

A more recent and interesting approach is that taken by [11], where they propose a new algorithm to play The Resistance known as DeepRole. The approach combines Counterfactual Regret Minimization (CFR) along with deep neural networks, to make deductions about players in the game. DeepRole outperforms almost all human players.

3 Benchmark

The purpose behind the project was to develop an intelligent agent that can play games of ”The Resistance”. This is considered a challenging task because ”The Resistance” is a social deduction card based game and also an imperfect information game. There are two main factions in the game; the spies and the resistance. None of the resistance know each other, but the spies do. In a series of five mission, each faction has to play so as to win the game.

In each mission, a certain team is chosen by the leader, and other members get to vote and decide whether or not, that particular team should go forward for the mission. If there are a majority of positive votes, the mission takes place. If there are a majority of negative votes, then, a new team will have to be selected.

Once a team embarks on a mission, the team members can decide whether to sabotage the mission or let it succeed. Even if a mission gets sabotaged, there is no definite way of knowing who or which member sabotaged it.

And also, since resistance players do not know who the other resistance players are, there is always uncertainty in choosing a team. Players may manipulate other players, give false or no information. Therefore, making an agent perform well in such an environment can be demanding.

The framework [1] to run "The Resistance" game is the one hosted by the AIGameDev Team for their competition in 2012, and it was built by Alex J Champanard.

The various agents built for the purpose of the project interact in a simple way with the framework. Upon running a Competition (a number of games together), the framework randomly chooses the spies and the resistance from the agents provided. The agents provided in the framework include those that have been submitted for the competition and also those from the previous year projects. Certain information is hidden automatically from some of the players if they are not spies. This includes information like who the other spies are. This is all taken care of by the framework.

4 Background

Three agents were created using the following techniques.

4.1 Rule-Based Approach

In a typical Rule-Based approach [8], a certain curated set of rules have been specified, typically by a domain expert, and an intelligent agent uses these rules to behave in different ways. It usually contains an inference engine and a working memory as well.

4.2 Multiple Regression

Multiple Regression [2] is an extension of Linear Regression, where we want to predict our dependent or target variable based on more than one independent variables. Multiple Regression is good for data where there is some linearity among the independent and dependent variables.

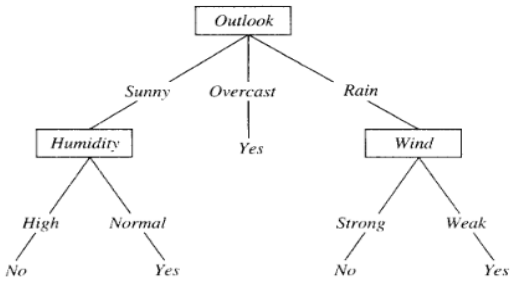
$$Y = b_0 + b_1x_1 + b_2x_2 + ... + b_nx_n \tag{1}$$

This is the line equation for Multiple Regression. Y represents the predicted output value. As we can see it is dependent on a lot of attributes or independent variables.

4.3 Decision Trees

A Decision Tree [8] [2] is a tree like structure that can be used to classify or predict a target class based on a given set of attributes. The internal nodes of the tree correspond to the attribute, while the leaf nodes gives us the final class. It is usually considered one of the best supervised learning methods, and is also preferred because of the ease of interpretation.

Fig. 1: Decision Tree Example



5 Techniques Implemented

5.1 Simple Agent

The Simple Agent uses a rule-based system to play the game. It contains a lot of if-else rules determining how it should behave in each turn of the game. The Agent makes use of a dictionary to store the probability value of someone being a spy.

The agent here makes use of a dictionary to store the probability value of someone being a spy, and uses that to make a decision on whether or not to include someone on their team (this decision takes place only if agent is resistance). The value is calculated in a simple way. When a game starts, every player in the game has a value of 20.

If in a two player mission, there is one sabotage, then the bot definitely knows that the other player was a spy. And the corresponding player's value is increased by a large value. Similarly if it was three player mission, and there was a sabotage, values of the other two players are increased slightly. In case a mission goes without any sabotages, the values of all members of the team is decremented by a small amount.

When it is time to choose the next team, the agent chooses the players with the least value so as to let the resistance win the game.

If the agent is spy, then the agent randomly picks one player from the list of spies, and the remaining players from the resistance members. This is to ensure that there is always one spy in a team.

5.2 Data-Collection

A Data-Collection Agent was made to play a few thousand games against both beginner bots and intermediate bots. From this, data after each mission was collected. The data collected was stored in a CSV file so that it could be accessed later for the purpose of learning. The data included:

- Spy Probabilty: Normalized spy probability values as calculated by the Simple Agent
- Team: Whether or not they were part of the team for a particular mission
- Leader: If they were the leader of the mission

- Voting History: History of their votes for the mission
- Mission Outcome: Whether the mission succeeded or failed
- Spy: Whether the particular player was a spy or not

5.3 Agent using Multiple Regression

Data collected from a few thousand games by the Data-Collection Agent mentioned in the previous section is given to Multiple Regression model built using the Scikit-learn library.

The Multiple Regression model takes the current data of players as a list, as input, and then as output gives the class of whether or not a a competitor is a spy. Based on this, if our agent is part of the resistance, they can make a decision of who to include in their team.

5.4 Agent using Decision Trees

A Decision Tree Classifier is built using the Scikit-Learn Library, which uses the CART algorithm to build a tree. Data collected by the Data-Collection Agent is given to the CART Algorithm, so as to predict whether a given player is spy or not.

If our agent is playing as resistance, this information can be used to decide who to include in the team. This is a form of learning, because, we are classifying a player as spy or not, based on their past behaviors and actions.

5.5 Voting Mechanism

Once a team has been selected, voting takes place to decide whether or not the team should go forward with the mission. The voting mechanism is the same in all three agents.

If the agent is the leader, then it will always vote for its own team. If the agent is part of the resistance and someone else is the leader, and the current team is a three player mission and our agent is not part of, it will vote to disband the team, so that it can be a part of the mission.

If it is a spy, it will vote for the team to go forward as long as there is at least one spy on the team.

6 Experimental Study

Each of the agents were tested against the beginner and intermediate bots in 1000 game batches, and the results were recorded.

6.1 Simple Agent

The Simple Agent employs a rule-based mechanism to help it play. The rules basically encapsulated the common features or knowledge of playing the game and also kept track of each players previous behaviours and actions, so as to help it predict a competitor's faction. The simple agent performed mostly well against the beginners and moderately well against the intermediate bots. After this, the goal of the project shifted to a more learning based approach for our agent.

Fig. 2: Simple Agent against beginners

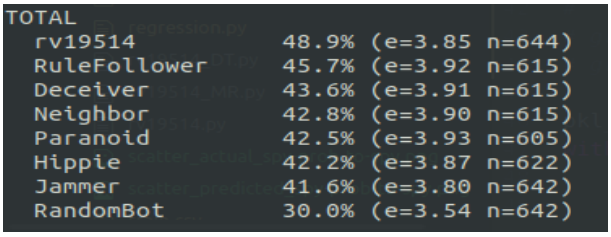
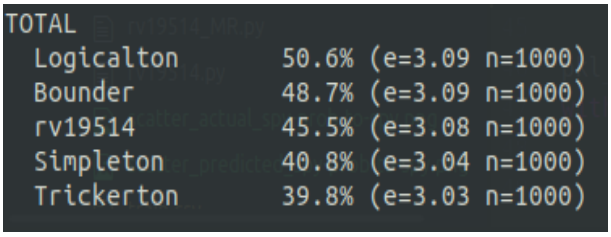


Fig. 3: Simple Agent against intermediates



6.2 Agent using Multiple Regression

This Agent makes use of Multiple Regression to help predict whether a player is a spy or not. It can perform reasonably well against beginner bots and also well against the intermediate bots.

Fig. 4: Multiple Regression Agent against beginners

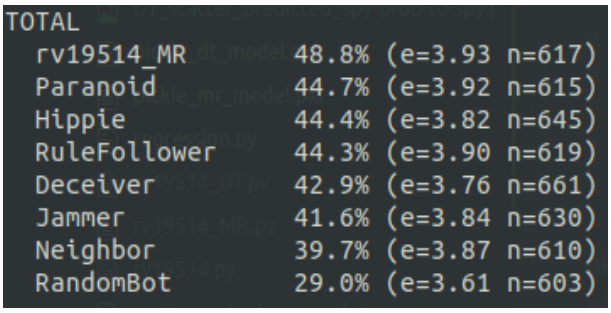
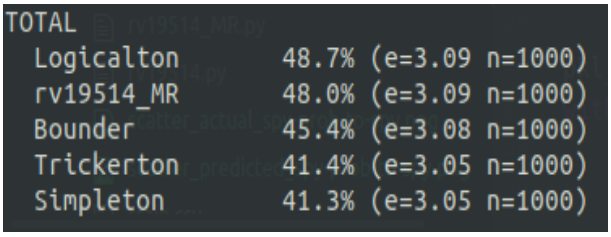


Fig. 5: Multiple Regression Agent against intermediates



6.3 Agent using Decision Tree

This Agent uses a Decision Tree Classifier on the data that has been collected to predict whether a given player is a spy or resistance. The agent usually fares well against the beginner bots, but at times can perform low due to the randomness of certain bots. The random nature of certain bots does not fit in with the general nature of learning task. The agent also performs moderately well against the intermediate bots, usually ranking second against them. It also outperforms the Multiple Regression Agent most of the time.

Fig. 6: Decision Tree Agent against beginners

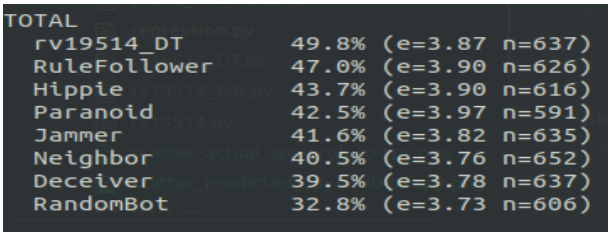
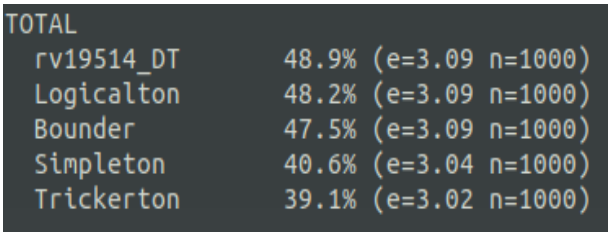


Fig. 7: Decision Tree Agent against intermediates



7 Analysis

7.1 Development Process

The simple agent performed well considering the simplicity of its rule base. Making use a dictionary and calculating the spy probability of each player made a huge change in its performance, because now there was some semblance of decision making to select a team and it was not just random. The same framework of the simple agent was used in both the learning agents, with minor modifications.

The Agent using Multiple Regression worked surprisingly well considering the task it was given. The data that was collected was definitely not linear. That is, we can plot the graph and see that there is no linear correlation between the data provided and the output (spy or resistance). Although we needed an output of 0 (resistance) or 1 (spy) from the learning agent, the multiple regression model can only give a continuous range of values. Since it draws a straight line that approximately matches the data provided, we will never get a 0 (resistance) or 1 (spy). But due to the continuous range of values being provided we know that the higher the value, the closer that player has a chance of being a spy. So we can once again take the players with the least output value and use them as members of the team. This led to an improvement in the agent's score.

The Regression model worked well, but there was scope for improvement, and that is where the Decision Tree Agent comes in. The decision tree was able to classify the test data set with a good deal of accuracy, and the agent using the decision tree saw an increased score while competing with other agents most of the time.

The agents always sabotage a mission if they are a spy. This is done because, not sabotaging the first mission lead to decrease in the agent's score. The main reason for this decrease is that, the other bots don't usually make much decisions about player behavior. So the agent sabotages all missions to increase their score.

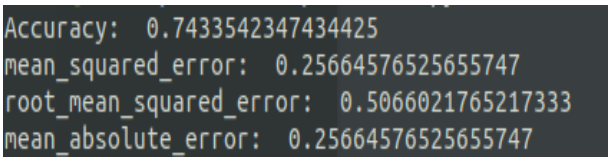
7.2 Result Analysis

7.2.1 Performance

Each of the machine learning models, were trained on 80% of the data collected by the Data-Collection Agent. Each of the model was then tested on the remaining unseen 20% data, and evaluated.

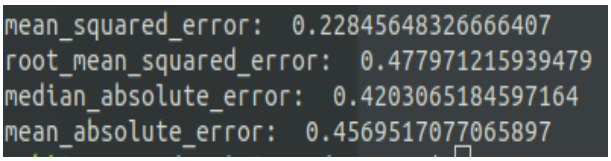
The decision tree model achieved an Accuracy of 74%, and a Mean Squared Error of 0.256.

Fig. 8: Decision Tree Evaluation



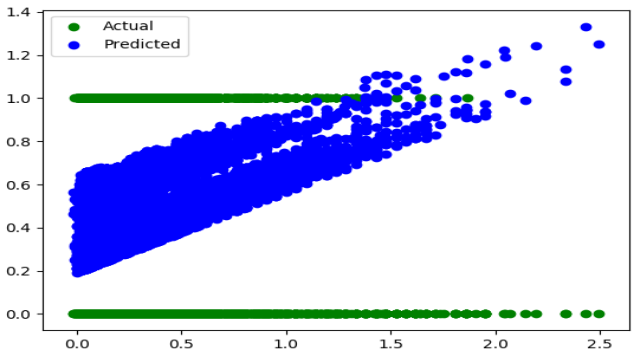
The multiple regression model has a slightly lesser Mean Squared Error value of 0.228.

Fig. 9: Multiple Regression Evaluation



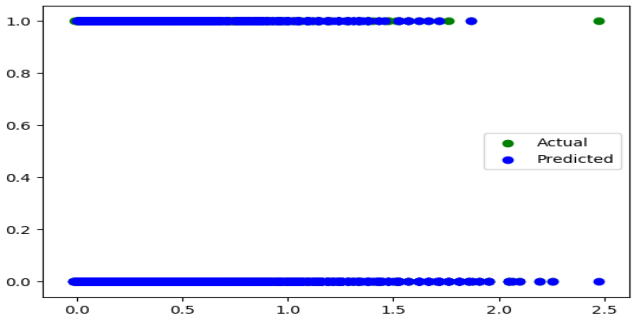
The actual and predicted results of the multiple regression model were plotted with respect to the spy probability values calculated by our agent, which is also one of the attributes given as input to the model.

Fig. 10: Actual vs Predicted spy probabilities using Regression



A similar scatter plot was plotted for the Decision Tree model. As we can see, most of the green points in the Decision Tree scatter plot, which represent the Actual values of players being spy or resistance, are not noticeable because there are overlapping blue points (predicted values). Therefore we can observe that the Decision Tree does a much better job at classifying from the available data, the faction of other players.

Fig. 11: Actual vs Predicted spy probabilities using Decision Tree



7.2.2 Overall Result

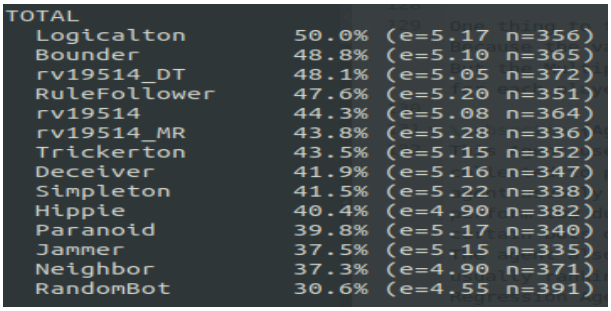
If a competition is run with beginners, intermediate bots and the three different agents, we can see that in most of the time, the Decision Tree Agent outperforms the other two agent, but at times, the other two agent may outperform the decision tree. This is because even though we have used a machine learning approach to teach the agent how to act based on behaviors, it is still a difficult concept to learn. There are more dependent variables that haven't been considered, and whether or not a player is a spy depends on many more hidden features and attributes.

Fig. 12: Competition 1 with every bot

```
TOTAL
  rv19514_DT        49.3% (e=5.23 n=347)
  Logicalton        48.7% (e=5.04 n=374)
  Bounder           48.2% (e=5.10 n=365)
  rv19514_MR        48.1% (e=5.16 n=356)
  RuleFollower      46.2% (e=5.07 n=368)
  rv19514           43.6% (e=5.24 n=340)
  Simpleton         43.5% (e=5.12 n=357)
  Trickerton        42.1% (e=5.25 n=336)
  Paranoid          41.4% (e=4.99 n=370)
  Deceiver          41.0% (e=5.04 n=362)
  Hippie            40.8% (e=4.84 n=393)
  Jammer            37.4% (e=5.07 n=346)
  Neighbor          35.0% (e=4.89 n=362)
  RandomBot         29.9% (e=4.95 n=324)
```

As such, even though a machine learning approach can do well in most cases here, it won't always perform well. A deep learning approach with neural networks or Deep Q-Learning, while complicated, may have the potential for learning and classifying behavior much more accurately. Even a Decision Tree Classifier but with more training data may perform better.

Fig. 13: Competition 2 with every bot



8 Conclusions and Future Work

Three different techniques were tried, tested and analyzed in this report. The simplest way to build an agent is definitely using a rule-base. The rule-based system is able to capture the general sense of how to play the game and use it effectively in most games, even outperforming the other two learned agents at times. But a learning agent, if implemented properly can have an immense advantage in any situation, because it can learn how its opponents think, and this can be a valuable asset in an incomplete information game like ”The Resistance”. The Multiple Regression model while didn't clearly specify whether a player was spy or resistance, still gave a range of probable values, from which such information could be deduced. While not completely accurate, it still improved the score of the agent. The Decision Tree model seems to work the best, and achieves the best accuracy in predicting the player's faction, given their history of behaviors and actions.

The multiple regression model, though it performed well, was not inherently suited for a task such as this, and this was why the decision tree classifier was built. While there are competitions where the decision tree agent scores lower than the other two agents, it is still one of the best performing agents overall. It has the potential to perform even better, if more training data could be collected.

One thing that can be observed is that even with a good decision tree, we will never be able to achieve a 100% accuracy in predicting the faction of the player. This is because, there is more information or attributes that determines how a player behaves, that has not been mapped to in our dataset. These hidden features and attributes could be found under careful analysis and scrutiny.

Therefore, even the best machine learning models may hit a bottleneck in performing amazingly well without knowing these extra pieces of information. An approach such as Recurrent Neural Networks (RNN) may work well in predicting the faction because of its ability to learn and memorise temporal data. Deep Q-Learning is another possible avenue to explore in the future to build a better agent.

References

1. A. J. C. AIGameDev, "Resistance framework," https://github.com/aigamedev/resistance, 2012.
2. E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.
3. C. Browne, E. J. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4:1, pp. 1–43, 2012.
4. M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.
5. C. Girlea, R. Girju, and E. Amir, "Psycholinguistic features for deceptive role detection in werewolf," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 417–422.
6. S. M. Lucas, "Computational intelligence and ai in games: A new ieee transactions," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 1–3, March 2009.
7. J. Marinheiro and H. L. Cardoso, "A generic agent architecture for cooperative multi-agent games." in *ICAART (1)*, 2017, pp. 107–118.
8. I. Millington, *AI for Games*. CRC Press, 2019.
9. D. Novak and D. Verber, "Real-time strategy games bot based on a nonsimultaneous human-like movement characteristic," *The GSTF journal on computing*, vol. 3, no. 2, pp. 43–48, 2013.
10. R. L. Rivest, "Game tree searching by min/max approximation," *Artificial Intelligence*, vol. 34, no. 1, pp. 77–96, 1987.
11. J. Serrino, M. Kleiman-Weiner, D. C. Parkes, and J. B. Tenenbaum, "Finding friend and foe in multi-agent games," *arXiv preprint arXiv:1906.02330*, 2019.
12. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016. [Online]. Available: http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html
13. D. P. Taylor, "Investigating approaches to ai for trust-based, multi-agent board games with imperfect information; with don eskridge's "the resistance"," 2014.