CE811: Assignment 1

Muhammad Saad, ms22045
School of Computer Science and Engineering
University of Essex, UK
ms22045@essex.ac.uk

The Resistance is a role-playing card-based party game. In this report we created our computer based player called MyBot that can play this game automatically. The idea is to create such an autonomous bot that can play this social role-based game by itself using any of the learned techniques studied in this course. For the purpose of this assignment we implemented different neural network models to achieve high accuracy of decision making while playing the game.

Introduction

The resistance is a card-based game, and python framework for The Resistance board and card game, along with various AI bots were submitted for the 2012 competition at the Game/AI Conference. Although the conditions are hard coded, each bot in this particular game has different attributes which has an effect in their decision taking abilities resulting in a different outcome after each game. In this report, our motive is to create such a bot that can play this game using any of the techniques studied in this course such as deep learning, decision trees, behavior trees, blackboards, reinforcement learning and evolution. We have used a neural network to solve this problem, along with parameters tuning for the model while training our objective is to achieve high accuracy without overfitting the model.

The Resistance

The resistance is a social role-playing card-based game which is based on a future where government and corporations are corrupted by money and power, affecting the life of common people; therefore, some people are planning to overthrow the government in secret. The players in the resistance are motivated individuals planning to overthrow the government but there are some government spies which have also infiltrated this movement. If a player is a spy, their motive is to prevent the missions of the resistance movement to be successful, else if a player is in resistance movement, his motive is to make every mission successful by selecting only trusted players for each mission. Different leader is selected at the start of each game, then the selected leader has to select a team of individuals; afterwards, everyone has to vote if the team should proceed or not, if the voting is failed then the team is selected again with some changes while selecting the players, otherwise the mission proceeds. If there is a spy in the team, he can sabotage that mission. On condition that the majority of missions were sabotaged in the game, the spies will win, else the Resistance will win. In this report, we will program our own bot to play this game automatically where it will try to win the maximum number of games either while playing as a spy or resistance.

Background

The game playing problem can be solved by many algorithms, such as decision trees, behavior trees, deep learning, reinforcement learning etc. After careful consideration of the data gathered from the games played by bots provided, we came to the conclusion that it can be solved by a simple neural network [11] that can learn a function that can map the inputs to the output target which in our case is the binary value of the player being a spy.

The Neural Network is inspired by the sophisticated functionality of human brains where hundreds of billions of interconnected neurons process information in parallel. Examples include language translation and pattern recognition software [12] An artificial neural network, often known as a "neural network," consists of an input layer, three or more hidden layers with varying number of neurons, and a layer of output at the end. Every neuron in the forward layer is forward connected to each neuron in the forward layer, and each connection has a weight assigned which is a numeric value. There is also an activation (or transfer) function in each neuron. In addition to introducing non linearity to the neural network, the activation function's other goal is to limit the value of each neuron so that the network is not paralysed by divergent neurons. Some of the common activation functions are Relu, sigmoid, and tanh. [12]. After the neural network output, the sum of loss is calculated by taking the difference of the neural network predicted output and the actual output. This calculated loss is used by the model to learn new weights in the backpropagation, with the help of gradient descent, where the weights are updated of each neuron [13]. Therefore, the updated weights improve the model to perform better in the next iteration and decrease the loss.

The submitted player MyBot in the assignment 1, named MyBot plays with the help of a trained neural network model, which predicts the probability of a player being a spy. This neural network is trained on 10000 games played by different bots, the data provided has some information about how many missions were failed, how many missions total played, and what was the pattern of voting of each player, apparently the historical data also contains whether the player was spy or not for the reason that after the game is completed players are able to know about who was the spy and who was not. The neural network model used in this assignment is a sequential model with 3 hidden layers with 512, 1024, and 512 neurons. During training the model updates the weights which are connecting each neuron present in the model using Adam as an optimiser to converge the model to optimal point [9].

Techniques Implemented

Firstly, the game was played 10000 times with different levels of bots and the data of each game is logged in a file, along with the boolean value if a player is a spy or not. The logged data has columns total missions, failed missions, number of upvotes with suspect counts and number of downvotes with suspect counts, and apparently the target boolean value of spy.

Once data is gathered, the next step is to preprocess the data. For normalization of the data to constraint the value in the range of 0 to 1, we used min-max normalization technique as it is proven to improve the prediction accuracy in most dataset [3].

The normalized data is then divided into training and validation sets. Afterwards, the sequential neural network model is created using keras library initially with two hidden layers, 1 input and 1 output layer. The hidden layers have 64 and 128 neurons, each with activation function Relu and

batch size of 16. For optimizing the model weights, we used Adam Optimizer to converge the model to global minimum. The Adam algorithm is a gradient descent method that is based on adaptive estimation of first-order and second-order moments. In the area of neural networks, the ADAM-Optimizer is one of the most popular adaptive step size methods. It was invented by Kingma and Ba [1].

Unfortunately, the accuracy and loss of this model was not up to the expectations. It gave an accuracy of 61.09% and binary cross entropy loss of 0.65. Following this result, some parameters had to be tweaked to make the model train better.

To make our model perform better accuracy, we first tried to change the number of hidden layers and the number of neurons in those layers. Therefore, we created a new model with the same settings but now a total of five hidden layers with 64, 128, 256, 128, and 64 neurons, the batch size is now set to 128. After training is concluded the model has not performed very well than our last model, the accuracy and loss both were same as our last model.

Moreover, we tried to experiment with the new settings in our model. We trained the model which now consists of 2 hidden layers, with 512, and 1024 neurons in the layers each with Relu activation function along with batch size now decreased to 32. In conclusion, the results were not improved that much with this model either. Accuracy is now 65.22% and loss is 0.4229, which is still undesirable.

Finally, we created a new model with three hidden layers, having 512, 1024 and 512 neurons in the layers, along with the batch size of 32. This model, to our surprise, gave an accuracy of 78.86% and a loss of 0.41. We have now integrated this model to play the games in the resistance against bots.

Experimental Study

Preprocessing of data, selection of techniques to use for playing the game, and training the model with selected parameters, our goal is now to create an agent that can play the game against other bots and perform better than them by using our trained model. We will now refer to our bot as MyBot in this report.

Firstly, MyBot played the games without using the trained model and changing any default logic for voting and selecting. To our expectations, MyBot was performing below average, scoring around 40%. Afterwards, we tried to tweak the selection and voting logic to see if it works better with hard coded conditions; hence MyBot performance was increased to mediocre after some hard coded changes, to be precise, the average score for 100 games were now around 45%. Even though that is not our end goal, we did some hard coded changes to compare bot’s performance before and after integrating the trained neural network model.

Secondly, we integrated a basic neural network model which had an accuracy of 60% to guide MyBot to make decisions based on a model trained by gathered data of 10000 games. But after integration of this basic model, the bot's performance has not changed a bit, average score for 100 games played remains to be around 45% for MyBot. It was apparent that our basic model failed to learn the function that determines the underlying mapping of input and output.

Lastly, we decided to utilize our best performing neural network model in MyBot predict function. After running 100 games the results were evidently improved, securing an average score of 62% against other bots. The neural network was used along with some enhancement in the logic of selecting and voting for missions functions. Running multiple iterations of the game, MyBot always proved to be a top ranked bot, which is evidence that our neural network model was able to learn the hidden mapping, while making our agent able to correctly determine whether a player is a spy or not for most of the time.

Analysis

We tried to run the game with no rules, the bot performed the least scores among the players, afterwards we tried to run the game with few hard coded rules, to which our player is now performing a little bit better than two worst playing bots.

Then we tried using a basic neural network model with low accuracy and trained with data of 1000 games played. But after integration of this model, MyBot was not performing well, to our expectation the performance was still the same.

After the upset from basic model, we moved to neural network to train the model with the same data, but this time data was also normalized, during training we noticed that accuracy and loss was not improving, on inspecting the data, we found out that to train the model data of 1000 games is too less, therefore, we prepared the data of 10000 games and train the model on this new large data. Apparently loss and accuracy now were improving on each epoch, but not up to our expectation, the model has to be tuned to perform better. For that, different activation functions, hidden layers, number of layers, gradient descent functions were used to determine the best possible combination for our model. Apparently neural network is a black box, and for most of the parameters one can not distinguish how a parameter is changing the accuracy and how it is affecting the result positively or negatively, though some techniques are better than others because it is backed by previous researches, for instance, we chose Adam optimizer over stochastic gradient descent to optimize the model, because this algorithm takes into consideration the 'exponentially weighted average' of the gradients [6]. Using averages makes the algorithm converge towards the minima in a faster pace. For increasing the hidden layer, there is a probability that the more hidden layers and neurons there are, the more complex mapping a model can learn.

Once the training for our main model is completed, MyBot uses that trained model to predict the probabilities of in-game players being a spy. To improve the decision making for voting of missions and selecting players to go on missions with our trained model. We are firstly predicting the probabilities of each player being a spy and if our bot is playing as a resistance, we are only selecting the players which have lowest probabilities of being a spy, so that mission can be successful. Otherwise we try to select atleast one spy in the mission. For voting, if our bot is in resistance then we down vote the missions if any of the players has high probability of being a spy, else we upvote the mission because there is a very low probability that there is still a spy in the mission. If we are playing as a spy, then we only upvote missions who have at least one spy to decrease the chance of the mission to be successful.

There is a reason that performance of MyBot is limited, because for one agent, playing as resistance, correctly predicting the probabilities of other players being spies or not have little effect in the game as other players might not know who the spies are and will try to vote for the mission to go ahead, which spies can sabotage later. Given that our agent is the leader, now MyBot can correctly determine the probabilities of players being spy during selection; therefore, it makes sure to select the trusted and honest players so that the mission will be successful. Hence, if not being a leader limits our ability to score higher as a player in resistance for the reason that the winning also depends on other players contributing equally by voting for missions which do not have any spies.

Overall Conclusions

In conclusion, after carefully selecting the neural network parameters, the scores for MyBot improved dramatically; it can be improved more after research in selecting different techniques to play this game, carefully selecting the data for training, dropping unrelated columns which have no effect on output, and parameter tuning. This is also a finding that the score for the game, if played as Resistance member, will be correctly determined when our agent is the leader of the game because while predicting spies correctly, we can select only those who have little to no chance of being a spy, even after that, if other players don't vote for the mission to progress then our agent can do nothing but select different combination of players where a spy can also mix up in the team and can sabotage the mission later. This limiting element of the game prevents our single intelligent agent from performing better.. Moreover, for future work, as playing as a spy we can intelligently not sabotage the mission sometimes to not get noticed.

References

1. D. Eskridge, The Resistance Game, 2009.
2. Daehyon Kim (1999) Normalization methods for input and output vectors in backpropagation neural networks, International Journal of Computer Mathematics, 71:2, 161-171,
3. Soufiane Hayou, Arnaud Doucet, Judith Rousseau Proceedings of the 36th International Conference on Machine Learning, PMLR 97:2672-2680, 2019.
4. Aksu, G. , Güzeller, C. O. & Eser, M. T. (2019). The Effect of the Normalization Method Used in Different Sample Sizes on the Success of Artificial Neural Network Model . International Journal of Assessment Tools in Education , 6 (2) , 170-192 .
5. Nazri Mohd Nawi, Walid Hasen Atomi, M.Z.Rehman. Faculty of Computer Science and Information Technology (FSKTM), Universiti Tun Hussein Onn Malaysia (UTHM), 86400 Parit Raja, Batu Pahat, Johor, Malaysia.
6. Sebastian Bock, Josef Goppold, Martin Weiß, An improvement of the convergence proof of the ADAM-Optimizer, 2018 .

7. Siddharth Sharma, Simone Sharma UG Scholar, ACTIVATION FUNCTIONS IN NEURAL NETWORKS, Dept. of Computer Science and Engineering, Global Institute of Technology, Jaipur, 2020

8. Kiwook Bae, Heechang Ryu, Hayong Shin, Does Adam optimizer keep close to the optimal point, Cornell University, 2019

9. Daehyon Kim a a Transportation Research Division , Highway Research Center , Songnam-Shi, Kyonggi-Do, Korea, 2007 (for normalization min max)

10. Michulke, D., Thielscher, M. (2009). Neural Networks for State Evaluation in General Game Playing. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2009. Lecture Notes in Computer Science(), vol 5782. Springer, Berlin, Heidelberg.

11. Wang, SC. (2003). Artificial Neural Network. In: Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, vol 743. Springer, Boston, MA.

12. III.3 - Theory of the Backpropagation Neural Network**Based on “non indent” by Robert Hecht-Nielsen, which appeared in Proceedings of the International Joint Conference on Neural Networks 1, 593–611, June 1989. © 1989 IEEE.