Teaching a neural network to play Minesweeper

Frederick Schultz
Mentored by Dr. William Mattson

Introduction

Machine learning is the process of training a neural network through experience and the use of data sets. With data sets, machine learning is accomplished by feeding a neural network training sample data and confirming the results from the sample data. A neural network is a system of connected nodes, also called neurons, that attempts to recognize underlying relationship in a set of data, similar to how the human brain learns. After being given sets of training data, the network can be put to work, accomplishing whatever task it has been trained to do. In most applications of neural networks with machine learning, the task that must be accomplished has all information available. In gaming, these scenarios are known as perfect games, where all players know all the information all the time, like chess. One example of an imperfect game is Battleship, in which the opponent's board is hidden. Past machine learning tests have been completed using guessing systems, which could be very helpful in Minesweeper, where many tile positions require guessing for the game to advance (Castillo & Wrobel, 2003). Minesweeper was chosen for this project, because it is also an imperfect game. In this study, the software PyCharm was used to create the neural network that played Minesweeper. This system was extremely helpful because of its basic structure and simple features. Furthermore, Scikit, an addon for PyCharm, was chosen to create the network, because it does not require much manual written code due to its interface, making it easier for those with less experience (Pedregosa et al., 2011). The network was compared to an algorithm to determine its effectiveness. The null hypothesis was that there would not be a difference between algorithm and network success rates. The alternate hypothesis was that that there would be a difference in success rates.

Materials and Methods

The neural network was trained over the course of two weeks to play Minesweeper before testing began. A sample of 200 games was taken from each play group. One group was the algorithmic method, and the other group was the neural network method. A move counter was coded into both systems, so the number of moves required to complete a single game could be retrieved and documented. The games were played on a 16 by 32 cell board with

Materials and Methods (cont.)

99 randomly dispersed mines. In order to test the actual skill of the system, and not its luck, the mines were not distributed to the board until after the first move had been made. Processing, a coding software, was used to create the algorithm system, and PyCharm was used to create the neural network system. TensorFlow and Keras, libraries of functions, were installed into PyCharm in order to train the network. The algorithm method is shown in Figure 1. Each game was played to completion and manually recorded as a win or loss into the data table (Table 1). If the game was recorded as a win, then the number of moves that were played in that game was recorded as well. A move was determined as any time that the player clicked on a tile to reveal it.

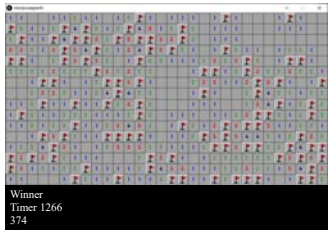


Figure 1 (above): Minesweeper board as played by the algorithm. The output box displays the time to play (in milliseconds), and number of moves taken during that game (374).

	Algorithm		Neural Network	
Game	Win(1)/Loss(0)	Moves	Win(1)/Loss(0)	Moves
24	1	324	1	352
25	0		0	
26	0		0	
27	0		0	
28	0		0	
29	0		1	400
30	1	385	0	
31	0		0	
32	0		0	

Table 1 (above): A sample of the data gathered from the 200 games played for each method.

After all of the games had been played to completion, the number of games won by each system was divided by the total number of games played to calculate the win rate. The win rates and total move counts for the successful games were statistically tested to determine if any differences were significant.

Results

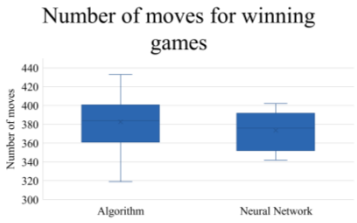
A two-proportion *z*-test was carried out to compare the win rates of the algorithm and neural network (Table 2). The *z*-test showed there was a significant difference in the win rates of the two systems ($z = 4.15$, $p < 0.001$) confirming that the algorithm produced more successes. The number of moves for winning games by the algorithm and neural network were compared with

Results (cont.)

a two-sample *t*-test (Graph 1). The algorithm with 38 wins (M = 382.68, SD = 29.66) and the neural network with 10 wins (M = 373.5, SD = 20.60), $t(46)=1.13$, $p = 0.27$. Using an alpha level of 0.05 the null hypothesis is not rejected, suggesting that there was not a statically significant difference between the number of moves in a winning game between the two systems.

Algorithm		Neural Network	
Win rate	Avg. Moves	Win rate	Avg. Moves
0.19	382.7	0.05	373.5

Table 2: These are the compiled results of the data gathered on all games played and average moves from successful games.



Graph 1: This graph shows the number of moves taken to complete games of both methods.

Conclusions

The test results confirmed that there was a significant difference in the success rates of the two Minesweeper playing methods, with the algorithm being the more winning method. Additionally, it was determined there was not a significant difference in the number of moves played in a winning game for each method. Further investigation could be done into why the neural network has an equivalent efficiency but lower win rate than the algorithm. To increase the effectiveness of the neural network additional training sets, beyond the 200 used in this project, could be used. Additionally, different algorhims could be analyzed or used as inspiration for the neural network. The methods of this research could be applied to analyzing other imperfect games such as poker.

References

Castillo, L. P., & Wrobel, S. (2003). Learning minesweeper with multirelational learning. https://www.ijcai.org/Proceedings/03/Papers/079.pdf

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., & Cournapeau, D. (2011). Scikit-learn: machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825–2830. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf