chizuru4rogue: Deep Learning for Deep Dungeon Crawling

Dylan G. Drescher

B.Sc. Computer Science - University of Bath

2022 - 2023

Abstract

In this article we introduce chizuru4rogue, which is a computer program designed to play the revered video game Rogue. It is designed to use all the information at its disposal to determine the next best move in order to push further and win the game.

Copyright

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my sincere gratitude to the following people:

Dr. Jie Zhang
My project supervisor, for providing me with invaluable guidance and feedback on my dissertation throughout the year.

Mr. Raphael and Mrs. Christine Drescher
My parents, for always encouraging me to strive for the best and never doubting my ability.

1 Introduction

2 Literature Review

3 Rogue

From here, a "run" refers to a single Rogue gameplay session from start to end.

4 Design

A player interacts with Rogue through the keyboard alone. The game is turn-based, so the agent interacts with the game every time they may perform an action on their turn. When it is the agent's turn, the agent receives an encapsulation of the data a human can observe, henceforth known as the "observation state". The agent then returns a keystroke that they wish to use as their action, as every action in Rogue is performed with a keystroke.

As the game is terminal based, parsing each pixel is unnecessary, what we do instead is take each cell and what letter they currently are as input.

As Rogue is a complex environment, chizuru4rogue will utilise a combination of reinforcement learning algorithms.

Rogue is a partially observable Markov Decision Process. To deal with this, we use a Long Short-term Memory system, an extension of a feedforward neural network, to process the sequence of observations. This is because LSTMs are capable of "remembering" information for longer periods of time. The LSTM algorithm was first defined by Hochreiter & Schmidhuber (1997) and popularised much later, one example of an agent implementing a LSTM including AlphaStar by Vinyals et al. (2019).

The goal of chizuru4rogue is to maximise the final score that the agent gets within one run. A run's final score is used as the reward for the reinforcement learning methods within the agent. A run's final score is determined by how much gold a player collects. The deeper a player ventures in the dungeon, the more gold they can collect. Additionally, the player gains a large score bonus if the game ends while the player possesses the Amulet of Yendor, an item found in dungeon level 26.

We use a combination of supervised learning and self-play. During the supervised learning portion of the learning process, we provide a combination of replays from humans and Rog-o-Matic (Mauldin et al. (1983)), an expert system that plays Rogue to a human level. During the self-play portion of the learning process, chizuru4rogue will play thousands of runs using Rogueinabox (Asperti et al. (2017)) as an interface to receive game state and send actions.

Using only reinforcement learning is challenging, mainly due to the large

action space that Rogue provides. Unlike most other video games where the actions you can perform is contextual, Rogue is a game where every single command is available to you at all times. This allows the player to develop a wide variety of strategies, but increases the overall complexity of the game. Additionally, some commands are combined with selecting an item from the player's inventory e.g. "wear chain mail armour", increasing the size of the action space in different contexts.

4.1 Policy Optimisation

Our goal was to find an optimal policy that maximises the chance that the agent can successfully reach the 26th dungeon level and get the Amulet of Yendor.

5 Requirements Specification

6 Implementation

7 Agent Training and Investigation

8 Reflection

9 Conclusion

References

Asperti, A., De Pieri, C. & Pedrini, G. (2017), 'Rogueinabox: An environment for roguelike learning', *International Journal of Computers* **2**, 146–154.

Hochreiter, S. & Schmidhuber, J. (1997), 'Long short-term memory', *Neural computation* **9**(8), 1735–1780.

Mauldin, M., Jacobson, G., Appel, A. & Hamey, L. (1983), 'ROG-O-MATIC: A belligerent expert system'.

Vinyals, O., Babuschkin, I., Czarnecki, W. M. et al. (2019), 'Grandmaster level in starcraft ii using multi-agent reinforcement learning', *Nature* **575**, 350–354.