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.

Declaration of Access

Contents

List of Figures

List of Tables

Acknowledgements

To-do

1 Introduction

2 Literature Review

2.1 Fundamentals

The fundamentals of reinforcement learning and many fundamental algorithms is explained in detail by Sutton & Barto (2018). The core idea behind reinforcement learning algorithms is that an agent performs *actions* on an *environment* by deriving what it should do from its *policy*, which is a mapping from states to actions. Once the agent performs an action, it receives the new game state as well as a *reward* signal, telling the agent how good its choice was.

The purpose of rewards is for the agent to constantly estimate a *value function*. This function tells the agent either how profitable being in a certain state and following its policy is, or how profitable taking a certain action then following its policy is. The theory is that the agent should aim to maximise its reward over the long term.

One of the most well-known reinforcement learning algorithms is the Q-learning algorithm (Sutton & Barto 2018, Chapter 6.5). In this algorithm, the agent keeps track of a table, mapping state-action pairs to its value. When the agent reaches a certain state, it consults its Q-table to determine the most valuable action to take.

2.2 Deep Learning

While the Q-learning algorithm can solve simple problem domains sufficiently, when it comes to more complex domains that don't have fully observable states such as Atari games, the amount of resources it takes to run the algorithm can become extremely large. The way that Mnih et al. (2015) chose to solve this is to use a convolutional neural network to approximate the Q-learning action-value functions, through an algorithm the authors call "Deep Q-network".

The Deep Q-network in their writing was shown to play several Atari games to a superhuman level, most notably Video Pinball and Boxing. A similar algorithm involving neural networks was employed by Silver et al. (2016) in their development of the AlphaGo system, an agent that was found to beat human grandmaster players in the game of Go. The authors used a convolutional neural network alongside "policy gradient" reinforcement learning, where

While the DQN algorithm by itself is serviceable for simpler problem domains such as Atari, there are better methods to tackle more challenging domains. When trying to create an agent that plays the online game Dota 2, Berner et al. (2019) used a Long Short-term Memory network.

LSTMs were first defined by Hochreiter & Schmidhuber (1997) and improved upon in later works. An LSTM is an extension of the "recurrent" neural network, where nodes use feedback connections to allow the network to "remember" information in the long term. This solves the problem that traditional neural networks have, where they can't store information that can be useful to them in the long term.

2.3 Exploring Rogue

The first notable instance of a program being developed to play Rogue was by Mauldin et al. (1983), where they created "ROG-O-MATIC", an expert system that plays Rogue. An expert system, as stated by Jackson (1986) in their book's introduction, "is a computing system capable of representing and reasoning about some knowledge-rich domain". Essentially, these systems aim to emulate a human expert in a particular domain and their decision-making. While expert systems are artificial intelligence, they make no use of machine learning to learn and adapt to situations, they follow what instructions have been programmed within them. ROG-O-MATIC provides a good yardstick to measure the performance of the agent we will be creating.

An interface for machine learning agents to play Rogue has been created, called Rogueinabox (Asperti et al. (2017)). Rogueinabox is a framework that allow developers to create agents that interface with the game Rogue. In the Rogueinabox article, the authors ran a Deep Q-learning agent on the game for testing. They simplified the problem domain to have the agent only consider exploring the dungeon to find the stairs, without fighting or collecting items. Their agent performed reasonably well accounting dungeon exploration alone, however, the aim of our agent is to reach the Amulet of Yendor and clear the game, which is difficult if the player does not fight monsters and gets stronger.

A paper by Cortesi & Asperti (2018) improved upon the initial agent proposed in the original Rogueinabox paper. The problem domain was still simplified to only consider getting to the exit stairs alone. While the previous implementation employed a DQN, the agent in Cortesi's paper implemented an A3C algorithm as a base, rather than a DQN. The A3C algorithm is first defined by Mnih et al. (2015) and involves

2.4 Exploring Other Roguelikes

Rogue is not the only roguelike that has been explored with machine learning. NetHack is another roguelike that has been explored in SkillHack (Matthews et al. (2022)).

3 Rogue

Rogue is a 1980 role-playing computer game inspired by text-based adventure games and tabletop role-playing games like Dungeons and Dragons that led to the creation of "roguelikes" - games that are based off of the core gameplay of Rogue. Roguelike games are mainly characterised by challenging, turn based hack and slash gameplay, procedurally generated levels and permanent character death.

In Rogue, your objective is to descend the Dungeon of Doom to slay monsters, get lots of gold, collect the Amulet of Yendor and escape the dungeon with it alive. The game is turn based, which means the player can spend as long as they want thinking their next move before the game processes the environment.

In every run[1], the player starts from scratch in dungeon level 1, where they must explore the dungeon, defeat monsters to get stronger and find better equipment. The dungeon is randomly generated each run, providing a unique challenge to the player every time they play.

The dungeon configuration is initially obscured to the player, revealing itself as the player moves around. As the player explores the dungeon, they will encounter enemies of increasing difficulty. Enemies in the game will aim to harm the player by attacking and reducing the player's HP[2]. If the player's HP reaches 0, the player dies and loses the game. When the player loses or quits the game, the player must start a new run.

Character permanent death provides a very interesting situation in the game that isn't present in many other games where you can save your progress and load previous save games. *Michael Toy*, Rogue's co-creator, touched on the topic of permadeath in Roguelike Celebration 2016 in Gamasutra (2016) by saying 'We were trying to make it more immersive by making things matter ... "this thing matters, so I'm going to think about this."'. Every decision you make has weight to it, as the player is unable to undo any mistakes they make or adverse situations they end up in, so the player is inclined to think their actions through, providing a sense of tension in the game that would otherwise be absent.

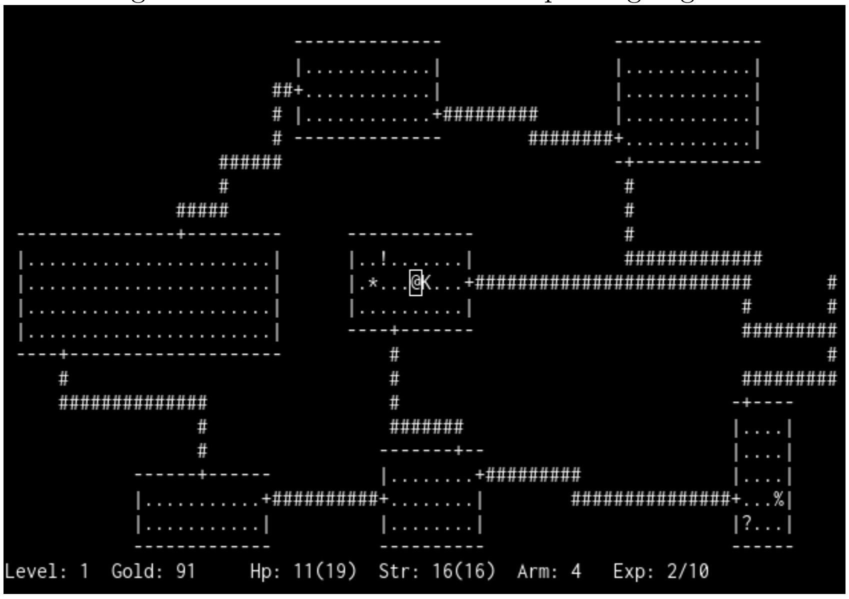
4 Design

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

[1]A single Rogue gameplay session from start to finish.

[2]Hit Points; representing how much more damage the player can take before dying.

Figure 1: A screenshot of an example Rogue game.



```
                           --------------                --------------
                           |............|                |............|
                         ##+............|                |............|
                         # |............+#########       |............|
                         # --------------        ########+............|
                    ######                               -+------------
                    #                                     #
                #####                                     #
 ---------------+---------      ------------              #
 |.......................|      |..!.......|              #############
 |.......................|      |.*...@K...+##########################       #
 |.......................|      |..........|                         #       #
 |.......................|      ----+-------                         #########
 ----+--------------------          #                                        #
     #                              #                                #########
     ##############                 #                               -+----
                  #                 #######                         |....|
                  #                -------+--                       |....|
            ------+------          |........+#########              |....|
            |...........+##########+........|        ###############+...%|
            |...........|          |........|                       |?...|
            -------------          ----------                       ------
Level: 1  Gold: 91     Hp: 11(19)  Str: 16(16)  Arm: 4   Exp: 2/10
```

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.

Berner, C., Brockman, G., Chan, B., Cheung, V. et al. (2019), 'Dota 2 with large scale deep reinforcement learning'.

Cortesi, D. & Asperti, A. (2018), 'Reinforcement learning in rogue'.

Gamasutra (2016), 'Rogue co-creator: permadeath was never supposed to be 'about pain'', https://www.gamedeveloper.com/design/-i-rogue-i-co-creator-permadeath-was-never-supposed-to-be-about-pain-.

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

Jackson, P. (1986), *Introduction to expert systems*, Addison-Wesley Pub. Co., Reading, MA.

Matthews, M., Samvelyan, M., Parker-Holder, J., Grefenstette, E. & Rocktäschel (2022), 'Hierarchical kickstarting for skill transfer in reinforcement learning'.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A. et al. (2015), 'Human-level control through deep reinforcement learning', *Nature* **518**, 529–533.

Silver, D., Huang, A., Maddison, C. et al. (2016), 'Mastering the game of go with deep neural networks and tree search', *Nature* **529**, 484–489.

Sutton, R. S. & Barto, A. G. (2018), *Reinforcement learning: an introduction*, MIT Press.

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