chizuru4rogue: Deep Reinforcement Learning for Dungeon Crawling

Literature, Technology and Data Survey

Dylan G. Drescher

December 2022

1 Introduction

Reinforcement learning is known to solve a wide variety of complex problems. One domain that is especially popular in applications of reinforcement learning is computer games. Reinforcement learning can be used to create agents that play games, or enhance the gameplay experience for the player. There are many papers out there that cover reinforcement learning and machine learning. This review will focus on reinforcement learning and other machine learning applications in playing computer games, in order to discover how best to address the problem of creating an agent to play Rogue.

2 Previous Work

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 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.

One recent example of applying machine learning techniques to Rogue

is Rogueinabox, by 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. Their agent did not take gold and combat within the game into account, the goal of the agent was to reach the exit of each dungeon level. 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 extraordinarily difficult if the player does not fight monsters and gets stronger.

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'.

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.

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.