Reinforcement Learning in Pokémon Red to explore complex multi-reward environments

by

LIAM O'DRISCOLL
URN: 6640106

November 4, 2023

Department of Computer Science
University of Surrey
Guildford, Surrey
England, United Kingdom
GU2 7XH

Project Supervisor: Sotiris Moschoyiannis

I declare that this dissertation is my own work and that the work of others is acknowledged and indicated by explicit references.

Liam O'Driscoll
November 4, 2023

0.1 Definitions

- ♠ Agent: The decision making mechanism recieving state information and performing chosen actions.
- ♠ Environemnt: The world in which the agent interacts with.
- ♠ State: A representation of the environment at the current timestep.
- ♠ Timestep: A value that increments after each action has passed since the start of the episode.
- ♠ Episode: An instance of the environment that the agent is interacting with.
- ♠ Action: The choice made by the agent in response to the state.
- ♠ Reward: The return value when an action is applied to a state.
- ♠ Reward Function: The mechanism in the environment that indicates how well the selected action is to achieving the goal of the environment.
- ♠ Policy: The decision making mechanism within the agent that decides the best action to perform given the state.

0.2 Abbreviation

- ♠ RL - Reinforcement Learning
- ♠ AI - Artifical Intelligence
- ♠ ML - Machine Learning
- ♠ RPG - Role Playing Game
- ♠ A2C - Ascynchronous Actor Critic
- ♠ DQN - Deep Q Network

1 Introduction

The aim of this project is to develop a reinforcement learning agent to play Pokémon Red to compare the effectiveness of different styles of RL algorithms. RL is an area of machine learning (ML) where agents make decisions and perform actions on states to achieve a goal.

1.1 Aims

- ✖ The aim of this project is to develop a RL agent to play Pokémon Red to compare the effectiveness of different styles of RL algorithms and their effectiveness to learn complex reward functions.

1.2 Objectives

- ♦ Research applications of RL to Pokemon and conduct a literature review on them
- ♦ Implement Pokémon Red game to be a suitable for training of different RL algorithms.
- ♦ Evaluate the performance of different RL algorithms used to train agents within the environment.
- ♦ Evaluate performance of agents to different forms of rewards functions.
- ♦ Recommend further developments to the project and applications to real world projects.

2 Literature Review

In RL, the decision making agent learns through experiences and 'trial and error'. Initially, it has a lack in understanding of the environment. However, through random action selection and the reward that it recieves, it is able to learn an understanding of the environemnt. The agent is incentivized to maximise its reward and will aim to find actions that will yield more reward. This constant state, action and reward loop is what helps the agent improve by altering the policy after every cycle.

RL is different to common traditional machine learning techiniques as it learns "how to map situations to actions-so as to maximize a numerical reward signal [1]." Agent's requirement to learn through experience and actions performed on current states not only affect the present, but also affect future states and actions are two characteristics which distinguishes itself from other forms of ML. It is also what makes Pokémon Red a suitable environment to apply this style of ML to.

RL has only been applied to every Atari game, where it surpassed the human benchmark for every game [2]. However, these games lack long term randomness and present actions influencing future states. The game 'Pokémon Red' is an RPG game filled with various puzzles, non-linear world and a large amount of variance making each play through of the game unique while also require achieving the same goals. In addition, the game has 2 states the players is constantly in, the player is either in an overworld where they control their movement on a map or they are in a battle with another individual, where they control the actions of their monster.

Other similar projects which applies RL to find the optimal battling strategy by Kalose et al [3]. Their work focuses on one aspect of the game and does not have a large enough search space to justify application of RL techniques. Another similar work by Flaherty, Jimenez and Abbasi [4] applies RL algorithms A2C and DQN to play Pokémon Red. However, this piece of work does not go into enough detail about the comparison of different RL techniques to find the best method to train an agent to complete large complex environments with a large search space. I aim to extend their research in applying RL to Pokémon Red by applying more algorithms and various techniques RL that I will go into more detail in section 3.

I chose to apply RL to this environment, Pokémon Red, because of the benefits it holds during trianing and applications of this research. This version of the game has the ability to speed-up the environment which allows for more timesteps to be completed so the agent can experience more states. Another reason is because its complexity. The end goal of Pokémon Red is to defeat all the gym leaders and become the champion. However, to reach this goal the player must complete a series of smaller tasks which are not explicitly specified in the reward function. An example of this would be navigation a 2-dimensional

plane, solving puzzles and performing pokemon battles along the way. Getting the agent to learn smaller tasks while completing the main goal of the environment can be applied and extended to the real world. Compared to other forms of AI, RL never stops learning even when deployed, which makes it a very effective method to adapt to new environments outside of the simulation and constantly learn to improve itself.

3 Technical Overview

For the environment, I plan on using a copy of the original game and having RL manually make inputs into the emulation of the game on my computer. This saves on a large amount of time manually recreating the game. Python has a gameboy emulator package called 'PyBoy' which I can run the game on and speed up emulation [5]. In addition, there is built-in support for RL training.

The implementations of the algorithms that will be used will come from stablebaselines. Stablebaselines' implementation of the algorithms have been well documented and have the ability to change the weightings of parameters used.

I will be using tensorboard to view the performance of the algorithms after training. Another reason why I am using stablebaselines' implementation of the algorithms is because of the integration with tensorboard to view the performance of the trained agent in a graph form.

When comparing algorithms, I will ensure that each algorithm being compared has an equal chance of performing at its best. I will hyperparameter tune each algorithm to find the best weightings for each algorithm's hyperparameter to obtain its best performance for comparison. Hyperparameter tuning is an additional step but has been found to greatly increase the algorithm's performance and efficiency, while also making comparison between algorithms more fair [6].

The first comparison of algorithms I will perform is value based methods to gradient descent. To compare the two different types of algorithms, I will use the algorithm A2C for gradient descent methods and PPO as value based methods. I will be aiming to compare each algorithm's learning rate, average reward level, losses and when their learning stabilizes.

Another algorithm I plan on testing is the performance of DQN to changes in how much the agent values future rewards. By changing the weighting associated with reward from future states, the agent would perform differently because it would try to move towards states with long term value versus short term high reward.

4 Workplan

Risks	**Solutions**
Issues with gettign the environment to work	Give ample time to researching working environments
Algorithms not supported with environment action space (discrete/continious)	Research a wide range of algorithms to be implemented
Agent unable to learn anything within the environment	Change the weightings for the algorithms or change the reward function of the environment
Reward function not specific enough	Research how other individuals implement their reward function and have multiple reward criterias to meet
All perform too smilarly making it difficult to compare	Change the weightings of each algorithm to ensure differences or train for longer

Month	Goals
October	• Rough structure of the report has been made. • Papers surrounding the project have been read (e.g., similar projects, algorithms that will be explored and technologies to be implemented) • Coding for the project is at its early stages. • Project Synopsis completed and submitted.
November	• Research and test which algorithms are applicable for comparison and applicable to project. • Draft introduction completed with a basic explanation of RL and how it is suitable for my environment. • Implementation of the Environment is complete
December	• Minimum viable product of code is achieved • Alter reward functions to give different incentives • Problem Analysis has been written • Design documentation and choice has been started
January	• Hyperparameter train sets of agents per algorithm • Train agents on different algorithms • Complete Design choice • Start evaluation of agents
February	• Any necessary extra agent training to be compelted • First version of Report is at a Submittable state
March	• Debugging time for any potential issues • Review of draft report submission
April	• Consider completing Extension Objectives • Final report completed
May	• Extra time for debugging • Last final checks on final version of report

5 References

[1] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement learning: An introduction*, Second. MIT Press Ltd, 2018.

[2] G. Brockman, V. Cheung, L. Pettersson, *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[3] A. Kalose, K. Kaya, and A. Kim, "Optimal battle strategy in pokemon using reinforcement learning," *Web: https://web. stanford. edu/class/aa228/reports/2018/final151. pdf*, 2018.

[4] J. Flaherty, A. Jimenez, B. Abbasi, B. Abbasi, J. Flahery, and A. Jimenez, "Playing pokemon red with reinforcement learning," 2021.

[5] Baekalfen, *Github: Pyboy*, 2021.

[6] B. Zhang, R. Rajan, L. Pineda, *et al.*, "On the importance of hyperparameter optimization for model-based reinforcement learning," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2021, pp. 4015–4023.