Reinforcement Learning - A Browser Based Visualisation Tool

Kevin Gleeson

B.Sc.(Hons) in Software Development

April 2, 2019

Final Year Project

Advised by: Mr Martin Hynes

Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract This project will help to explain the temporal difference reinforcement learning process by displaying an agents behaviour, performance and Q-Table as it interacts within its environment. The application is a browser based visual tool where a user can interact by tweaking parameters within a form. Once the form is submitted it will then make a request to run a main python script held on a flask server. Once the script has completed the user will be presented with and animation of the agent moving through it's environment. In addition a graph of the agent performance and the q-table will presented to the user for examination. This will aid the user in better understanding the idea of reinforcement learning.

Authors Kevin Gleeson 4th year student studying Software Development at GMIT Galway.

Chapter 1

Introduction

Reinforcement Learning is an unsupervised machine-learning technique that allows an agent to explore and learn from its environment without any prior knowledge of the domain. As the agent moves through the environment it gains knowledge via reward signals gathered by transitioning from state to another based on the action taken from its current state. With each step the agent is only concerned with its current state and what rewards it can gain from transitioning to it's next state. [] The agent chooses it's action decision based on what highest reward it can get from the next available states.
The purpose of this application is to demonstrate and explain reinforcement learning through a browser based visualisation tool.
The application will have the following elements on the Browser:

- The agent moving within its environment when the simulation is run. This will be displayed using HTML 5 canvas.
- User input to tweak parameters before each run of the simulation. The parameters that will be available to the user are:
  - The end goal reward
  - The negative trap reward
  - The agent learning rate
  - The learning decay rate
  - The discount factor
  - The Exploration rate
  - The Exploration decay rate
  - The per step reward
  - The maximum number of episodes to be run

- The maximum number of agent steps per episode
- Choice of algorithm

What is reinforcement learning?

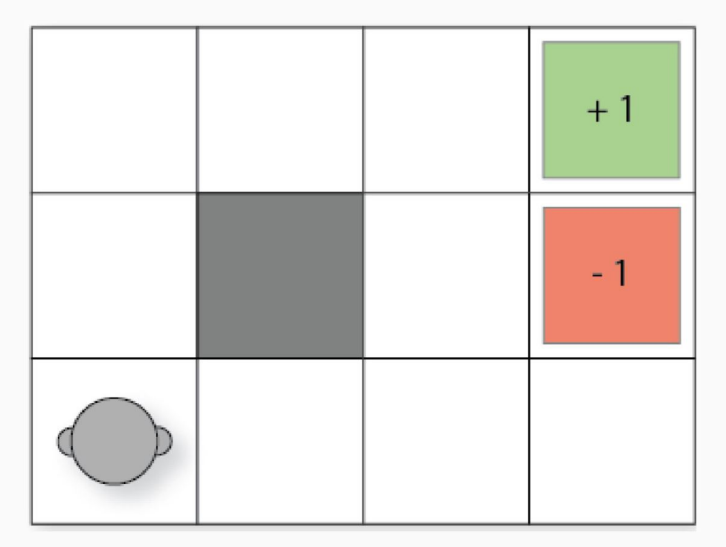


Figure 1.1

Reinforcement learning is the process of rewarding an agent for a decision made within its environment. The reward can be either positive or negative based on the decisions made by the agent as it transitions from one state to another.
For example, if a puppy has no knowledge of the sit command it will not perform the desired action on the first attempt. Each time the puppy sits when commanded its decision is reinforced with a positive treat/reward. If the puppy does not sit the reward is negative (no treat). Eventually after many iterations of training, the dog will associate a treat/reward with that specific command and eventually learn that sitting will get them a treat. The puppy in essence is taking actions to maximise rewards while exploring an unknown environment.
With reinforcement machine learning, this technique is used to train an agent to learn about its environment through trial and error. The environment used for this project will be grid world. The grid world domain is a two dimensional grid with the agent starting at the bottom left of the grid, the goal state is at the top right of the grid in addition there are traps that the agent needs to avoid while travelling from the start state to it's goal state.

- The agent's actions effect the environment by moving around and exploring.
- The state is what the agent can observe at a given time. In the grid above, the agent can occupy eleven possible squares. We can number theses states from 1 – 11 moving from left to right with the bottom left square being state number 1.
- In the agents initial state (State 1) it knows nothing about its environment and chooses an action of moving left, right, up or down.
- The Epsilon variable sets the probability of choosing a random action. When set to one it will always choose a random action. If set to .8 it will choose the a random action 80% of the time.

This will give the agent a chance to explore the environment depending on what the value is set to.

- Q values are a weighted score attached to an action of a particular state.
- There is a negative reward cost for each move the agent makes in this case -0.04. This will help in getting the best path to the end state.
- The learning rate (alpha) is a value between zero and one determines how much the Q value is updated for each action taken. It will be .5 for this example.
- The discount factor (Gamma) set to .9 is the immediate reward gained for an action taken. The higher the value the more the agent will take the immediate reward.
- The reward cost, gamma and alpha are hyper-parameters chosen by the user.
- There is a formula to follow to update the Q values of each action taken: Q (current state, action) += alpha *[reward + gamma* max value of Q (next state, all possible actions) – Q (current state, action)]
- The Q table is a record of all of the agent's actions taken in a given state. This is the agent's memory and is set to zero when first run. If all Q values are equal, it will Choose one at random.

State	Action left	Action right	Action up	Action down
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0

If the agent decides to move up one square to state 5 the Q table is updated using the formula above which looks like .5 * -.04 + .9 *0 – 0 = -0.02

State	Action left	Action right	Action up	Action down
1	0	0	-0.02	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0

If the Agent decides to move down to state one again the value of moving down from state 5 to state 1 is updated to -0.02 also.

State	Action left	Action right	Action up	Action down
1	0	0	-0.02	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	-0.02
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

Then when back in state one the agent's best choice (highest value) is down, left or right as they are all 0 and higher the -0.02. Eventually all of the actions of a given state will have a value added. The agent will chose the highest value as the optimal path to take to the end goal. Once the agent gets to either end state, the episode is terminated and re-run. When episodes are re-run, the Q-Table will continually update until the optimal path is found and minimal updates will be performed.

references [1, 2, 3, 4]

Chapter 2

Context

2.1 Overview

The aim of this project is to provide a visual aid that further explains the concept of reinforcement learning. The basic fundamentals of the Q-Learning and SARSA temporal difference algorithms are reasonably straight forward but can seem overly complex and verbose when attempting to verbally explain the topic. This application will help to show the user where and how the Q-values are stored and how the decision making process is made for the two above algorithms.

2.2 Objectives

The Main objectives of this project are:

- Implement two different temporal difference algorithms SARSA and Q-learning written in python.
- Allow for user interaction via a web page form
- Using Flask server to handle request from the user
- Present the user with data generated by the main python script on the server
- Parsing the Json, text and csv files generated via Ajax
- Use the parsed data to animate the agent in HTML canvas
- Google chart for graphing the agent performance

- Generate an dynamic table that updates from the csv file
- Add a heat map to the values of the table as it updates
- Deploying the application to Google Cloud Platform

2.3 Topics Covered

The chapters listed below will have the following elements examined.

- Methodology
  This chapter will explain what development process I used along with reasoning the technologies, algorithms and languages chosen.
- Technology Review
  This chapter will review each technological element of the application and provide justification for each technology discussed.
- System Design
  The overall architecture will be explained with diagrams of each component of the system supplied.
- System Evaluation
  The performance of the overall application will be evaluated here. In addition the limitations of the application discovered while in development will be discussed in detail.
- Conclusion
  In this chapter the results of the system evaluation will be discussed along with any new findings that may have occurred.

2.4 Github Repository

The below link is the url to my github repository holding my dissertation and software files.
https://github.com/kevgleeson78/Reinforcement-Learning.

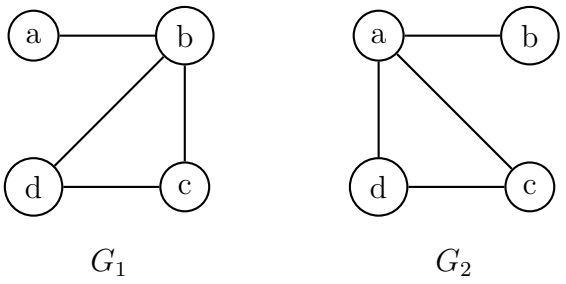
Chapter 3

Methodology

About one to two pages. Describe the way you went about your project:

- Agile / incremental and iterative approach to development. Planning, meetings.
- What about validation and testing? Junit or some other framework.
- If team based, did you use GitHub during the development process.
- Selection criteria for algorithms, languages, platforms and technologies.

Check out the nice graphs in Figure 3.2, and the nice diagram in Figure **??**.



G_1 G_2

Figure 3.1: Nice pictures

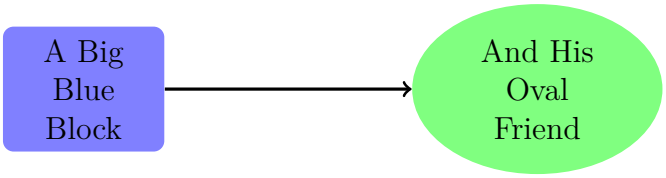


Figure 3.2: Nice pictures

Chapter 4

Technology Review

About seven to ten pages.

- Describe each of the technologies you used at a conceptual level. Standards, Database Model (e.g. MongoDB, CouchDB), XMl, WSDL, JSON, JAXP.
- Use references (IEEE format, e.g. [1]), Books, Papers, URLs (timestamp) – sources should be authoritative.

4.1 XML

Here's some nicely formatted XML:

```
<this>
  <looks lookswhat="good">
    Good
  </looks>
</this>
```

Chapter 5

System Design

As many pages as needed.

- Architecture, UML etc. An overview of the different components of the system. Diagrams etc... Screen shots etc.

Column 1	Column 2
Rows 2.1	Row 2.2

Table 5.1: A table.

Chapter 6

System Evaluation

As many pages as needed.

- Prove that your software is robust. How? Testing etc.
- Use performance benchmarks (space and time) if algorithmic.
- Measure the outcomes / outputs of your system / software against the objectives from the Introduction.
- Highlight any limitations or opportuni-ties in your approach or technologies used.

Chapter 7

Conclusion

About three pages.

- Briefly summarise your context and ob-jectives (a few lines).
- Highlight your findings from the evalua-tion section / chapter and any opportuni-ties identified.

Bibliography

[1] A. Einstein, "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]," *Annalen der Physik*, vol. 322, no. 10, pp. 891–921, 1905.

[2] D. Knuth, "Knuth: Computers and typesetting."

[3] M. Goossens, F. Mittelbach, and A. Samarin, *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.

[4] "1 intro up to rl/td.key." `https://login.cs.utexas.edu/sites/default/files/legacy_files/research/documents/1%20intro%20up%20to%20RL%3ATD.pdf`. (Accessed on 03/28/2019).