Reinforcement Learning - A Browser Based Visualisation Tool

Kevin Gleeson

B.Sc.(Hons) in Software Development

April 11, 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 (memory) as it interacts within its environment. The application is a browser based visual tool where a user can tweak parameters within a form before running the application. Once the form is submitted it will then make a request to run the application held on a 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. There are two different temporal difference algorithms for the user to choose from being Q learning, an off policy strategy and SARSA (State Action Reward Action), an on policy strategy. The performance of these two algorithms will be presented to the user for examination within a linear chart. This will aid the user in better understanding the concept of reinforcement learning.

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

Chapter 1

Introduction

Reinforcement learning

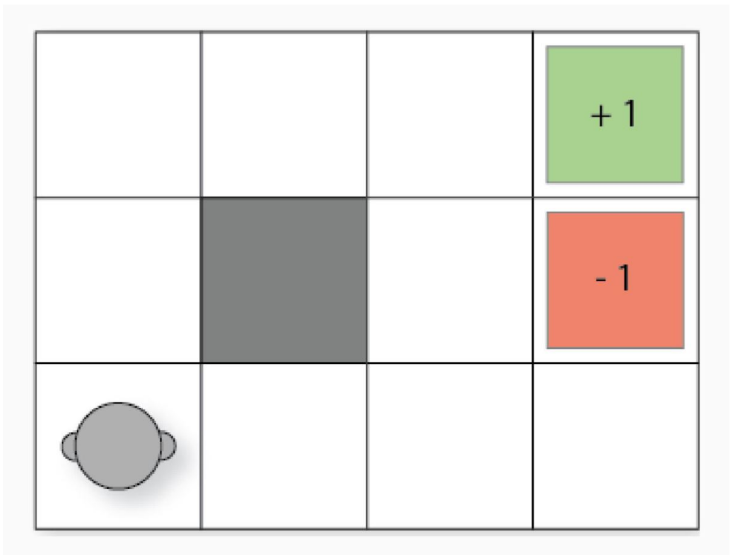


Figure 1.1

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. An agent transitions from one state to another by choosing an action with the highest reward value. The reward can be either positive or negative based on the decisions made by the agent as it transitions from it's current state to the next chosen state.
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. It will eventually learn the optimal path to a goal after many training episodes have completed.

1.1 Components

This application will have the following reinforcement learning components.

1.1.1 Agent

The agent is an object within the environment that makes decisions based on it's current state space and possible rewards gained by choosing an action. For this application the agent is placed in a two dimensional environment. The possible actions that can be taken by the agent are up, down, left or right.

1.1.2 Episode

An episode is the time from when an agent starts its training to when it has reached a positive or negative terminal state. Within each episode an agent takes time steps where each time step is the transition from its current position to the next within the environment.

1.1.3 Environment

The environment used for this project will be grid world. The grid world domain is a two dimensional grid with the agent's start position at the bottom left of the grid and the goal state 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. For the purpose of this application a 6 * 6 two dimensional grid environment will be used. As the script is running the agent will move from one square in the grid to the next adjacent square of up, below, to the left or the right of its current position. As the agent moves through the environment it gains knowledge via reward signals gathered by transitioning from one state to another based on the action taken from its current state.

1.1.4 Epsilon

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 value is decayed for every episode run to allow for the exploration of the environment.

1.1.5 gamma

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 an immediate reward with no concern for future rewards.

1.1.6 alpha

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.

1.1.7 per step cost

There is a negative reward cost for each step the agent makes in this case -0.04. This will help in getting the best path to the end state.

Q Values

Q values are a weighted score attached to an action of a particular state. The agents next movement is based on theses weighted scores.

Updating Q values

The agent chooses it's action decision based on what highest reward it can get from the next available states.
With each time step action taken by the agent the following q learning formulae is used to update the Q values within the Q Table. Q (current state, action) += alpha *[reward + gamma* max value of Q (next state, all possible actions) – Q (current state, action)]

This states to update the current states value for an action chosen using the maximum values from the next states for actions.

1.1.8 Q Table

The Q table is a historic record of the agent's actions taken in a given state. The values held within the Q table are used by the agent to choose the next best state transition. During the first episode all values within the Q table are zero (No knowledge of the environment). With every time step taken the Q table is updated with q values from the above formulae. For brevity the below examples will use a smaller q table rather than the one used for this application.

At the beginning of the learning process the Q Table has all zero values

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

meaning it has no prior knowledge of its domain.

From its initial state a random action is chosen. If the action chosen is to move up one square to state 5 the Q table is updated using the q learning formulae above which looks like .5 * -.04 + .9 *0 – 0 = -0.02. It then populates the Q table's row of State one's action of moving up to the q value of -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
11	0	0	0	0

When back in state one the agent's best choices (highest value) is down, left or right as they are all 0 and higher the -0.02. This happens for every time step the agent takes. When enough episodes have run, all of the actions of a given state will have a value added including the goal state value. From this the agent will chose the highest value each state as the optimal action 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.

State	Action left	Action right	Action up	Action down
1	-0.02	-0.02	0.02	-0.02
2	-0.02	-0.05	-0.02	-0.02
3	-0.02	-23	-23	-0.02
4	-0.02	-23	-100	-0.02
5	-0.02	-0.02	10	-0.02
6	-75	-100	1.2	-15
7	0	0	0	0
8	15	75	-0.02	-0.02
9	-0.02	85	-0.02	-0.02
10	-0.02	100	-0.02	-0.02
11	0	0	0	0

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

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 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 the github repository holding my dissertation and software files.
https://github.com/kevgleeson78/Reinforcement-Learning. The contents of this repository are:

- Dissertation folder
  This folder holds the latex files for my dissertation developed using Tex Studio

- FlaskApp Folder
  This is the main application folder stored on a Flask server when deployed.

- FlaskApp / flaskTest.py
  This file is used to serve the main static html page and handle http form requests

- FlaskkApp / Environment.py
  This file is the main file holding the logic and environment space for the application.
  All of the data files are generated from here once run.

- FlaskApp / app.yaml
  This file is used to deploy the application to a Google Cloud App engine instance.

- FlaskkApp / requirements.txt
  This file is used to declare what resources are needed fro the application to run on Goolge Cloud

- FlaskApp / Static / JavaScript
  Each of the files contained within this folder are the main JavaScript files controlling the HTML canvas environment, Google Chart and Q-Table data.

- FlaskApp / Static / Css
  The folder holding the styling script for the html pages

- FlaskApp / Static / gif The folder holding the gif animation for the loading page

- FlaskApp / Static / Data
  The folder holding the agents position coordinates as a .txt file
  A csv file for the agents Q-Table values A json file for the agent rewards gained for each algorithm

- FlaskApp / Templates
  This folder contains the the initial html page, the waiting page and result page. These pages are serves to the view when http requests are made by the user.

Chapter 3

Methodology

3.1 Initial Planning

At the beginning of this project the over all problem set was broken down into the following areas to allow for a more manageable modular development process:

3.1.1 Initial Meetings

After an initial meeting with Dr. Patrick Mannion a high level view of the project was explored. This gave me a grasp of what components would be needed for the high level structure of the project. The different components identified were:

- The Environment should be a two dimensional grid
- Environment constraints are in the form of a grid world game where the agent will attempt to navigate to an end goal state and avoid any traps present in the gird.
- Front end technologies used should be JavaScript based.
- User interactivity via a from
- Server side functionality to handle form requests

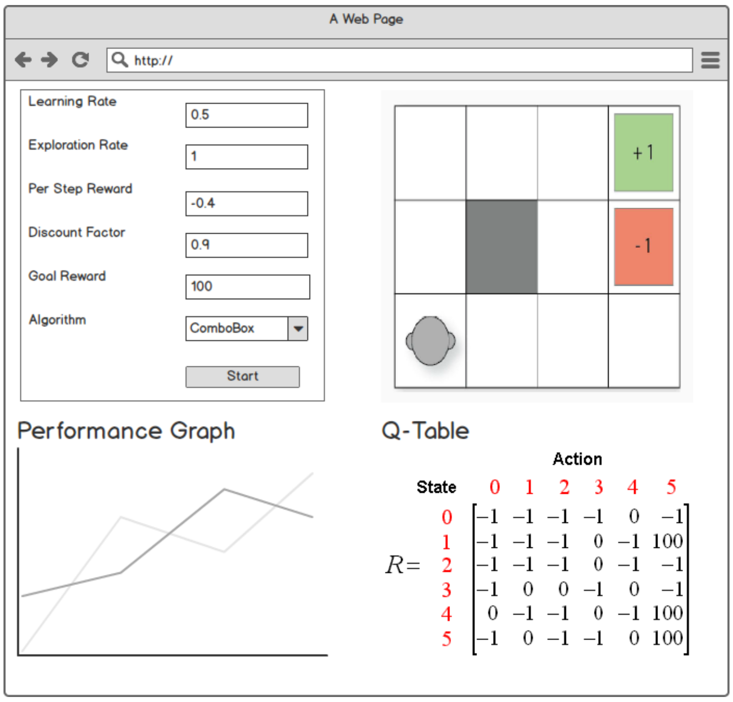
On further meetings with my project supervisor Mr. Martin Hynes the concepts were further broken down into fine grain units of work with initial milestones set for each phase of development.

3.1.2 Which area of reinforcement learning?

After viewing lectures for Temporal Difference Reinforcement Learning I decided that the Q-Learning algorithm would be best suited to the grid world environment. If time permitted I would then implement the SARSA algorithm for a comparison between the performance of the two algorithms. The reason I chose these two algorithms for comparison was they are very similar in design but have vastly different outcomes depending on the environment the agent is in.

3.1.3 Mockup of application

Reinforcement learning



Learning Rate	0.5
Exploration Rate	1
Per Step Reward	-0.4
Discount Factor	0.9
Goal Reward	100
Algorithm	ComboBox

Action

$$R = \begin{array}{c|cccccc} \text{State} & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & -1 & -1 & -1 & -1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & 100 \\ 2 & -1 & -1 & -1 & 0 & -1 & -1 \\ 3 & -1 & 0 & 0 & -1 & 0 & -1 \\ 4 & 0 & -1 & -1 & 0 & -1 & 100 \\ 5 & -1 & 0 & -1 & -1 & 0 & 100 \end{array}$$

Figure 3.1

3.1.4 User Requirements gathering

On meeting with Mr. Hynes the following user requirements were gathered from the mock up created and the high level components identified from the initial meeting with Dr. Patrick Mannion.

- When the application is launched the user should be presented with a form.
- The form should have input parameters the user can alter
- The user should have the ability to submit the form
- Once the form has been submitted the result page should be displayed
- Once the result page is displayed the user should be presented with:
  - The grid world environment
  - The agent moving within the environment
  - The Q-Table should be presented with updates as the agent moves from one state to another
  - A performance chart of the agents rewards for each episode
  - The type of Algorithm chosen above the form
  - The form on the result page should have the same parameters chosen when the form was originally submitted from the home page.
  - The user should have the ability to change the saved parameters and resubmit the from for a new view of the simulation
  - The layout should be responsive to all devices
- The user should have the ability to access the application deployed to the cloud

3.2 Mile Stones

Once the user requirements were identified the following mile stones were then set for the initial development of the application.

- Front end mock-up
- Requirements gathering
- Set up development environment
- Small scale command line prototype
- Write agent positional coordinates to text file

- Stream agent positional coordinates as raw data to web page
- Create HTML canvas grid environment

  Create Agent object within canvas grid

  Animate agent object in canvas from coordinate text file
- Write the total reward gained for each episode out to a Json file
- Display the reward Json file data in a linear chart
- Write out the Q-table to a csv file for each step the agent takes
- Read the csv file via an ajax request
- Display the csv file data to a table
- Clear the table after each step of the simulation
- Create a front end form for user input
- Data binding of form post request to variables held in Environment.py(main script)
- Data persistence of user input between requests
- Performance testing
- Cloud deployment

3.3 Selection criteria of Technologies

After the initial user requirements gathering the following technologies were then researched and chosen to begin the development of the application.

3.3.1 Programming Languages

Java had been considered as the main language but after consideration python was chosen for the ability to rapidly prototype small versions of the system allowing for an agile iterative approach to development.

The possibility of a purely front end JavaScript application was also explored but cross browser compatibility of the scripts needed was a major issue that indicated a different approach was needed. However JavaScript would be used for the front end of the application.

3.3.2 Integrated Development Environment

The IDE chosen for the development of the application is IntelliJ with the pyCharm plug-in. This plug-in is compatible with all of the frameworks needed for setting up the development environment. Git capabilities are also within the IDE allowing for tracking changes, committing and managing merge conflicts.

3.3.3 Server Side

For server side programming and scripting the Flask python framework was chosen. Flask is a light weight package that allows for the rapid deployment and development of web applications.

The Django python framework was also considered but it was decided that it was not necessary to have a fully blown MVC model seeing as the application has in essence only three pages of Home page, Training Page and result Page.

When testing a simple hello world application Django generated a project that was almost twice the size of a flask application with little or no difference in performance. For this reason Flask was chosen.

3.3.4 Front End

For the front end of the application all of the data will be presented within a web browser. The frameworks used for this application are HTML canvas for drawing the grid world environment and animating the agent around this space.

JQuery will be used to make ajax requests to retrieve the files from the server containing the relevant data needed.

D3.js was investigated for charting the agent performance but problems with the data rendering incorrectly were identified as a potential sign for an alternative method of displaying the data.

Google charts has been successfully tested and chosen for displaying the agent performance within the environment.

A table will be used to display the Q-Table values from the generated csv file. The JavaScript library ”Hottie” will be used to display the data within the table cells as a heat map based on the range of values from the highest to the lowest value.

3.3.5 Cloud Deployment

Initially Heroku was chosen for the cloud platform for the application to run on. However there was a problem with the server timing out after 20 seconds once a request has been made. Since the main script can take up to 30 seconds to complete this was a major issue.

The Google cloud application platform was investigated as an alternative solution and while there is a time out limit in place this can be altered if needed. For this reason the Google Cloud application platform was chosen fro deploying the application.

3.4 Scheduled meetings

Scheduled meetings were held with Mr. Martin Hynes every week to evaluate the progress of assigned tasks along with any problems encountered. These meetings were in the form of the scrum methodology where a quick overview of my progress was presented. In addition any problems I had were discussed along with what tasks needed to be done for the next scrum meeting. These new tasks identified were then assigned for the following week to complete.

3.5 Development Approach

An iterative development approach was used throughout the construction of this application. The first task was to develop a basic prototype that demonstrated the basic concepts of the algorithm Q-Learning. Once implemented this prototype will be the foundation for each additional feature added with the start of a new iteration.

3.6 Testing

No test suites were used to test this application however there was considerable manual testing done with each iteration completed. Any major bugs found that would stop the development of the project while testing were fixed right away. Minor bugs found such as layout issues could be addressed at a later date as they did not have a detrimental impact on the progress of the project. This testing strategy would be used for every small new feature added while developing.

3.7 Use of GitHub

Github was used to track my progress of the project. While this is a solo project github would still need to be used for version control allowing for the roll back of the system in the event of a new feature causing unwanted behaviour.

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:

```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).