

Rule-Based Interactive Assisted Reinforcement Learning

Adam Bignold

A/Prof. Peter Vamplew
A/Prof. Richard Dazeley
Dr. Cameron Foale



Thesis is submitted in total fulfilment of the
requirement for the degree of Doctor of Philosophy

Centre for Informatics and Applied Optimization
School of Science, Engineering and Information Technology
Federation University Australia
PO Box 663
University Drive, Mount Helen
Ballarat, VIC 3353, Australia.

Keywords

Advice; Assisted; Closing the Loop; Evaluative; Human; Human-Agent; Incorrect Advice; Informative; Interactive; Persistent; Reinforcement Learning; RDR; Ripple-Down Rules; Rule-Based; Simulated User;

Abstract

Reinforcement Learning (RL) has seen increasing interest over the past few years, partially owing to breakthroughs in the digestion and application of external information. The use of external information results in improved learning speeds and solutions to more complex domains. This thesis, a collection of five key contributions, demonstrates that comparable performance gains to existing Interactive Reinforcement Learning methods can be achieved using less data, sourced during operation, and without prior verification and validation of the information’s integrity.

First, this thesis introduces Assisted Reinforcement Learning (ARL), a collective term referring to RL methods that utilise external information to leverage the learning process, and provides a non-exhaustive review of current ARL methods. Second, two advice delivery methods common in ARL, evaluative and informative, are compared through human trials. The comparison highlights how human engagement, accuracy of advice, agent performance, and advice utility differ between the two methods. Third, this thesis introduces simulated users as a methodology for testing and comparing ARL methods. Simulated users enable testing and comparing of ARL systems without costly and time-consuming human trials. While not a replacement for well-designed human trials, simulated users offer a cheap and robust approach to ARL design and comparison. Fourth, the concept of persistence is introduced to Interactive Reinforcement Learning. The retention and reuse of advice maximises utility and can lead to improved performance and reduced human demand.

Finally, this thesis presents rule-based interactive RL, an iterative method for providing advice to an agent. Existing interactive RL methods rely on constant human supervision and evaluation, requiring a substantial commitment from the advice-giver. Rule-based advice can be provided proactively and be generalised over the state-space while remaining flexible enough to handle potentially inaccurate or irrelevant information. Ultimately, the thesis contributions are validated empirically and clearly show that rule-based advice significantly reduces human guidance requirements while improving agent performance.

Statement of Authorship

This thesis contains no work extracted in whole or in part from a thesis, dissertation or research paper previously presented for another degree or diploma except where explicit reference is made. No other person's work has been relied upon or used without due acknowledgement in the main text and bibliography of the thesis.

The content of this thesis, which is presented as a thesis incorporating published papers, consists of some papers that have been published/accepted or submitted in draft form. The PhD candidate declares that he is the main author of these papers and that his contribution to each paper is fifty percent or more. Explicit references have been made in chapters that have been submitted as papers.

A handwritten signature in cursive script, reading "Adam Bignold".

Adam Bignold
Federation University Australia
February 17, 2019

Supervisory Team



Adam Bignold (PhD Candidate)

a.bignold@federation.edu.au



Peter Vamplew (Principal Supervisor)¹

p.vamplew@federation.edu.au



Richard Dazeley (Associate Supervisor)²

richard.dazeley@deakin.edu.au



Cameron Foale (Associate Supervisor)

c.foale@federation.edu.au

¹Peter Vamplew was associate supervisor from March 2015 to June 2018.

²Richard Dazeley was principal supervisor from March 2015 to June 2018.

Acknowledgements

I would like to thank my supervisors, A/Prof Peter Vamplew, A/Prof Richard Dazeley, and Dr Cameron Foale, for their continuous support of my Ph.D. study and research. Their guidance, patience, and availability helped me throughout the development and writing of this thesis. I could not have imagined having a better group of supervisors. My sincere thanks also goes to Dr Matt Taylor and Dr Tim Brys, for their contributions and review of my Ph.D.

I would also like to thank my family, my mother, and my Grandma. My mother has shown more endurance in her life than I hope I would ever need. She will always remain an inspiration to me. Thank you to my Grandma for making sure I don't forget my family and for reminding me that there is a life outside of the office.

Finally, I would like to thank Amelia. Thank you for your support, for putting up with my late nights and my distractions, and for keeping me motivated. Your own work ethic and persistence has inspired me throughout this entire endeavour.

Adam Bignold was supported by an Australian Government Research Training Program (RTP) Stipend and RTP Fee-Offset Scholarship through Federation University Australia.

Contents

Abstract	ii
Statement of Authorship	iii
Supervisory Team	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	xiii
List of Tables	xv
1 Introduction	1
1.1 Motivation	2
1.2 Objectives and Approach	3
1.2.1 Thesis Questions	3
1.3 Contributions	4
1.3.1 Approach	6
1.4 Readers Guide	8
1.4.1 Contribution/Chapter Correspondence	9
2 Reinforcement Learning	10
2.1 Introduction to Reinforcement Learning	10
2.2 The Challenges of Reinforcement Learning	13
2.2.1 Curse of Dimensionality	14
2.2.2 Time Requirements	14
2.2.3 Behaviour Transfer	15

2.2.4	Goal Misalignment and Concept Drift	15
2.3	Interactive Reinforcement Learning	16
2.3.1	Current Research	17
2.3.2	The Challenges of Interactive Reinforcement Learning	19
2.4	Similar Domains	21
3	Assisted Reinforcement Learning	23
3.1	Introduction	25
3.2	Assisted Reinforcement Learning	28
3.2.1	A Conceptual Framework for Assisted Reinforcement Learning	30
3.2.2	Information Source	33
3.2.3	Temporal	34
3.2.4	Interpretation	36
3.2.5	Structure	38
3.2.6	External Model	39
3.2.7	Modification	40
3.2.8	Agent	42
3.3	Forms of Assisted Reinforcement Learning	42
3.3.1	Heuristic Reinforcement Learning	42
3.3.2	Interactive Reinforcement Learning	44
3.3.3	Reinforcement Learning from Demonstration	46
3.3.4	Transfer Learning	47
3.3.5	Multiple Information Sources	50
3.4	Future Work	52
3.4.1	Incorrect Assistance	52
3.4.2	Multiple Information Sources	53
3.4.3	Interpretability	53
3.4.4	Two-Way Communication	54
3.5	Conclusion	55

4	Experimental Methodology	56
4.1	Environment Descriptions	56
4.1.1	State Space Representation	56
4.1.2	Mountain Car	58
4.1.3	Self-Driving Car	60
4.1.4	Mario	65
4.2	Experiment Toolkit	74
4.2.1	Community Sourced Toolkit	74
5	Human-Sourced Advice	77
5.1	Human-Sourced Advice	77
5.1.1	Transfer Learning	78
5.1.2	Reinforcement Learning from Demonstration	79
5.1.3	Interactive Reinforcement Learning	80
5.2	Advice Style	80
5.2.1	Evaluative Advice	81
5.2.2	Informative Advice	81
5.2.3	Evaluative versus Informative	82
5.2.4	Human Engagement	83
5.3	Experiment	85
5.3.1	Methodology	85
5.3.2	Results	88
5.3.3	Conclusion	94
6	Simulated Users	96
6.1	Role of the Human	96
6.1.1	Characteristics of Human Interactions	97
6.1.2	Problems with Human Testing	98
6.2	Simulated Users	99
6.2.1	Current Applications of Simulated Users	100
6.2.2	Evaluation Principles	101

6.2.3	Representing Human-Sourced Information	102
6.3	Evaluative Methodology using Simulated Users in Interactive Reinforcement Learning	103
6.4	Experiment	105
6.5	Conclusion	108
7	Persistent Advice	109
7.1	Persistence	109
7.2	Experiment	112
7.3	Results	118
7.3.1	Probabilistic Policy Reuse	118
7.3.2	Persistent Advice on the Mountain Car Domain	120
7.4	Conclusion	126
8	Rule-Based Interactive Reinforcement Learning	128
8.1	Rules	129
8.1.1	Rules Trees	130
8.1.2	Ripple-Down Rules	132
8.2	Rule-Based Interaction Reinforcement Learning	135
8.2.1	RDR-RL Agent	136
8.3	Advice Gathering	136
8.4	Advice Modelling	137
8.5	Experiments and Methodology	140
8.5.1	Agents	140
8.5.2	Simulated Users	142
8.5.3	Environments and Experiments	147
8.6	Results	150
8.6.1	Mountain Car	150
8.6.2	Self-Driving Car	152
8.6.3	Super Mario Brothers	154
8.7	Conclusion	157

9	Conclusion and Future Work	158
9.1	Contributions	158
9.1.1	Assisted Reinforcement Learning	158
9.1.2	Evaluative versus Informative Advice	158
9.1.3	Simulated Users in Interactive Reinforcement Learning	159
9.1.4	Persistent Advice	159
9.1.5	Rules-Based Reinforcement Learning	159
9.2	Future Work	160
9.2.1	Effect of Latency on Accuracy	160
9.2.2	Simulated Users	160
9.2.3	Human/Agent Interfaces	161
9.2.4	Closing the Loop	161
9.2.5	Multiple Users	162
9.2.6	Incorrect Advice Identification and Mitigation	162
9.3	Final Words	162
	Bibliography	164
10	Appendix	177
A	Ethics Application	177
A.1	Plain English Statement	178
A.2	Ethics Approval	181
A.3	Participant Consent Form	183
A.4	Ethics Final Report	184
A.5	Questionnaire	189

List of Figures

2.1	The traditional Reinforcement Learning framework.	10
2.2	Illustration of self-driving car sensor positions and action space.	12
2.3	Interactive Reinforcement Learning.	17
3.1	Traditional Reinforcement Learning. Adapted from Sutton & Barto, 1998. . .	25
3.2	Assisted Reinforcement Learning.	28
3.3	Detailed view of the Assisted Reinforcement Learning Framework.	31
3.4	The Assisted Reinforcement Learning Taxonomy.	32
3.5	Example of audio interpretation in Assisted Reinforcement Learning.	37
3.6	Heuristic Reinforcement Learning.	42
3.7	Interactive Reinforcement Learning.	44
3.8	Transfer Learning Example	48
4.1	Descritization of the position space in the Mountain Car problem.	57
4.2	Labelled screen-shot of the RL-Glue Mountain Car environment (RL-Glue). .	58
4.3	Formulation of the Mountain Car policy	59
4.4	Representation of the agent in RL-Glue Mountain Car environment.	60
4.5	Representation of the agent in a Simulated Car environment.	61
4.6	Process flow of Simulated Car environment.	63
4.7	Optimal path for Simulated Car environment.	65
4.8	RL-Glue Infinite Mario environment.	66
4.9	Infinite Mario generalisation grid overlay example for Brys representation. .	70
4.10	Process flow of Infinite Mario environment.	73
4.11	Illustration of the RL-Glue protocol.	74
5.1	Definition of the evaluative assisted experimental agent using the Assisted Reinforcement Learning framework.	86

5.2	Definition of the informative assisted experimental agent using the Assisted Reinforcement Learning framework.	86
5.3	State-based accuracy of informative and evaluative participants for Mountain Car environment.	93
5.4	State-based availability of informative and evaluative participants for Mountain Car environment.	93
5.5	Reward bias of evaluative advice on Mountain Car. State-Based.	95
6.1	Definition of the evaluative assisted experimental agent using the Assisted Reinforcement Learning framework	106
6.2	Evaluation of an Assisted Reinforcement Learning agent using a simulated user using varying levels of availability and accuracy (Independent).	107
6.3	Evaluation of an Assisted Reinforcement Learning agent using a simulated user using varying levels of availability and accuracy (Dependent).	108
7.1	Definition of the non-persistent evaluative assisted experimental agent using the Assisted Reinforcement Learning framework.	113
7.2	Definition of the persistent evaluative assisted experimental agent using the Assisted Reinforcement Learning framework.	113
7.3	Definition of the non-persistent informative assisted experimental agent using the Assisted Reinforcement Learning framework.	114
7.4	Definition of the persistent informative assisted experimental agent using the Assisted Reinforcement Learning framework.	114
7.5	Probablistic Policy Reuse (PPR)	115
7.6	PPR versus direct-use action selection for Interactive Reinforcement Learning using retained informative advice.	119
7.7	Performance of non-persistent evaluative and informative agents on Mountain Car.	121
7.8	Steps per episode for non-persistent and persistent agents using evaluative advice.	122

7.9	Steps per episode for non-persistent and persistent agents using informative advice.	125
8.1	Binary Decision Tree.	131
8.2	Ripple-Down Rules (Debbie Richards, 2009).	133
8.3	Ripple-Down Rules.	134
8.4	Process flow of a Rule-Based Interactive Reinforcement Learning agent. . . .	139
8.5	Definition of the persistent informative assisted experimental agent using the Assisted Reinforcement Learning framework.	141
8.6	Definition of the persistent rule-based assisted experimental agent using the Assisted Reinforcement Learning framework.	141
8.7	Process flow for rule-based simulated user using RDR advice model.	145
8.8	Definition of the persistent rules-based assisted experimental agent with feature set interpretation using the Assisted Reinforcement Learning framework. . . .	149
8.9	Step performance for rule-based and state-based Interactive Reinforcement Learning agents for the mountain car environment.	151
8.10	Step performance for rule-based and state-based Interactive Reinforcement Learning agents for the self-driving car environment.	153
8.11	Reward performance for rule-based and state-based Interactive Reinforcement Learning agents for the self-driving car environment.	153
8.12	Unassisted Q-Learning benchmark performance on Super Mario using Littman and Brys+ feature spaces.	154
8.13	Rule-Assisted Interactive Reinforcement Learning on Super Mario using a Brys+ advice and state feature set.	155
8.14	Rule-Assisted Interactive Reinforcement Learning on Super Mario using a Brys+ advice feature set and a Littman state feature set.	156

List of Tables

1.1	Correspondence between the thesis contributions and the chapters	9
2.1	Example policy of a Reinforcement Learning agent controlling a car.	13
3.1	Correspondence between the authors section contributions.	23
4.1	Feature space of Infinite Mario. Littman Representation.	68
4.2	Feature space of Infinite Mario. Brys+ Representation.	69
4.3	Reward conditions for Infinite Mario.	70
4.4	Action space for Infinite Mario.	72
5.1	Participants self-reported understanding of the solution and dynamics of the Mountain Car environment.	89
5.2	Participants self-reported level of engagement with the agent for the Mountain Car problem.	89
5.3	Participants self-reported level of accuracy for the Mountain Car problem. . .	90
5.4	Average of participants self-reported sentiment of how well the agent followed their advice provided for the Mountain Car problem.	90
5.5	The average number of episodes that participants provided advice for on the Mountain Car environment.	90
5.6	Percentage of steps that participants provided advice for the Mountain Car problem.	91
5.7	The percentage of interactions in which the advice provided was optimal for the state/action in the Mountain Car problem.	91
5.8	Reward bias of evaluative advice.	94
7.1	Observed accuracy and availability for human advisors in the Mountain Car environment.	116

7.2	Simulated user settings for Mountain Car environment.	117
7.3	Agent/User combinations for persistent agent testing.	118
7.4	Number of interactions and percentage of interactions for non-persistent evaluative and informative agents in the Mountain Car environment.	120
7.5	Number of interactions and percentage of interactions for non-persistent and persistent evaluative and informative agents in the Mountain Car environment.	126
8.1	State-based simulated user knowledge bases for the Mountain Car and Self-Driving Car environments.	143
8.2	Rule-based simulated user knowledge bases for the mountain car and self-driving car environments.	146
8.3	Rule-based simulated user knowledge base for the Super Mario environment.	147
8.4	Number of Interactions and Percentage of Interactions for State-Based and Rule-Based Agents in the Mountain Car Environment	152
8.5	Number of Interactions and Percentage of Interactions for State-Based and Rule-Based Agents in the Self-Driving Car Environment	153

Chapter 1

Introduction

A long-standing goal of Machine Learning is the creation of agents and systems that are capable of functioning in real-world environments (Sutton & Barto, 1998). Common types of problems are decision/control tasks, where given some information about the current state of the problem or environment, the best action to take now in order to maximise long-term success must be determined. Reinforcement Learning (RL) has positioned itself as a solid candidate for such tasks. It owes its success to its ability to improve while operating, to learn without supervision, and adapt to changing circumstances (Kaelbling, Littman, & Moore, 1996). Classical Reinforcement Learning (Sutton & Barto, 1998) utilises an agent that interacts with an environment, learning by trial and error. The agent explores the environment and learns solely from the rewards it receives. Early work using this foundational approach has shown success in domains such as inventory management (Giannoccaro & Pontrandolfo, 2002), robot soccer (Kitano et al., 1997), and game-playing (Tesauro, 1994).

As with most Machine Learning techniques, Reinforcement Learning struggles to learn in large state spaces. As environments become larger the agent’s training time increases and finding a solution can become impractical (Cassandra & Kaelbling, 2016). In the mid-nineties, as RL was transitioning from toy problems to real-world tasks, Kaelbling et al. (Kaelbling et al., 1996) argued that if the field is to succeed in scaling then the use of information external to the environment would be needed.

Interactive Reinforcement Learning (IntRL) is a field of RL research in which a human interacts with an RL agent in real-time (Thomaz, Hoffman, & Breazeal, 2005). The human can provide extra information to the agent regarding its behaviour, the environment, or future actions it should perform. Humans are leaders when it comes to problem solving, forward planning, and teaching, and also have a large collection of knowledge and experiences to draw

upon when encountering new environments and problems(Thimmesh, 2006). The premise of Interactive Reinforcement Learning is to utilise these skills of humans to assist the agent with its own learning and problem solving. This approach has shown to considerably improve the agent learning speed and can allow Reinforcement Learning to scale to larger or more complex problems(Subramanian, Isbell Jr, & Thomaz, 2016).

1.1 Motivation

This research is motivated by recent advances in Interactive Reinforcement Learning in transferring human experience and expertise to Reinforcement Learning agents(Griffith, Subramanian, Scholz, Isbell, & Thomaz, 2013). Reinforcement Learning agents have shown over many decades the ability to learn complex behaviours through trial-and-error alone. However, it is the premise of this research and the field of Interactive Reinforcement Learning, that the tutelage of humans can be used to improve agent learning. The use of human knowledge, experience, and expertise can be utilised to assist Reinforcement learning agents to learn significantly faster, scale to more complex problems, and become more flexible(Knox & Stone, 2009; Thomaz & Breazeal, 2007).

Current Interactive Reinforcement Learning research has been limited to interactions that offer relevant advice to the current state only (Knox & Stone, 2010; Griffith et al., 2013; Krening et al., 2017). Additionally, the information provided by each interaction is not retained, and instead discarded by the agent after a single use. A motivation of this thesis is to provide the agent with a method for retaining and reusing provided knowledge, allowing humans to give general advice relevant to more than just the current state.

There are two of major barriers to humans providing information to RL agents. The first is the time required by the human. Allowing humans to provide information to the agent, and allowing the agent to know when that information is relevant, serves to reduce the number of interactions required. Additionally, the ability to provide generalized advice - advice that is relevant over a number of states - also reduces the number of interactions.

The second barrier is the skill needed by the human to provide the information. Humans have usually needed both programming skills and intimate knowledge of the problem dynamics to encode information relevant to the agent’s learning(Randløv & Alstrøm, 1998). A

principle of Interactive Reinforcement Learning is that the method for providing information to the agent should be understandable and usable by people without programming or domain expertise(Thomaz et al., 2005; Amershi, Cakmak, Knox, & Kulesza, 2014).

With these barriers in mind, and the previous motivations listed, these are the motivations of this thesis:

- **Retention and Reuse of Advice:** An agent should be able to remember advice given to it and reuse it in future generalised decision making.
- **Lower Interactions:** The time required by a supervising human should remain as low as possible to reduce the burden on the human.
- **Accessibility:** Methods for providing information to an agent should be accessible to users without programming or machine learning expertise.

1.2 Objectives and Approach

1.2.1 Thesis Questions

The principal question addressed in the thesis is:

To what extent can an iteratively built model of human-sourced advice be used by a Reinforcement Learning agent to accelerate learning and reduce human interactions in a real-time environment?

The thesis also addresses the following sub-questions:

- (i) How effective is evaluative¹ and informative² advice when used by the agent?**
- (ii) To what extent do the results of trials based on simulated users accurately predict the behaviour of an Interactive Reinforcement Learning system compared with real human users?**

¹The user evaluates the agent's past choice in actions. See Chapter 5.

²The user informs the agent's future choice in actions. See Chapter 5.

- (iii) **To what extent can a rule-based model be built iteratively that can handle noisy or incorrect advice?**

To answer these questions, this thesis contributes a novel approach to real-time advice delivery and retention to improve agent performance and decrease human-agent interactions with the following contributions:

- A framework which unifies prior research for externally-influenced Reinforcement Learning agents.
- A comparison of evaluative versus informative advice.
- A suitability study of simulated users for agent testing and benchmarking.
- A methodology for the evaluation of simulated users.
- A list of characteristics of human interactions.
- A rule-based model for advice retention
- Exception-driven iterative model development

1.3 Contributions

This thesis makes the following contributions to the fields of Reinforcement Learning and Interactive Reinforcement Learning.

- **Assisted Reinforcement Learning.** This thesis establishes Assisted Reinforcement Learning, a definition, framework, and taxonomy. Assisted Reinforcement Learning encapsulates Reinforcement Learning methods that use external information to modify/leverage the learning process. The Assisted Reinforcement Learning framework and taxonomy are used to describe and detail the relationship between external information and the agent, highlighting the process of information decomposition, structure, retention and how it can be utilised to influence agent learning. This framework is designed to foster collaboration by simplifying classification and comparability of methods that

leverage external information in the learning process. It is applied in this thesis to describe and illustrate the Interactive Reinforcement Learning agents to be used in this research.

- **Evaluative v.s. Informative Advice.** Interactive Reinforcement Learning uses information sourced from humans to influence the learning of an agent. Identified by this thesis are two methods for human advice delivery in Interactive Reinforcement Learning; evaluative and informative. Advice critiquing an agent’s performed actions is evaluative, and advice assisting the agent in future decision making is informative. Analysis of human trials performed in this research include a comparison of these two methods, providing insights into how human engagement, accuracy, and availability differ.
- **Simulated Users in Interactive Reinforcement Learning.** A simulated user is an automated entity designed to replicate the functions and behaviour of a human user. The purpose is to allow rapid and replicable training and testing. Instead of relying on human assistance, the agent relies on a simulated user whose source of expertise is defined ahead of time. They offer a quantitative method for representing and simulating humans for the evaluation and training of machine learning methods. Simulated users have so far seen only limited application in Interactive Reinforcement Learning research, and prior approaches have not clearly considered the full range of characteristics required for a simulated users. This thesis investigates the use of simulated users applied to Interactive Reinforcement Learning, and contributes methods for replicating some characteristics of human advice delivery. While the full range of human characteristics can’t be completely and accurately replicated, simulated users are suitable to evaluate Interactive Reinforcement Learning agents to get indicative results.
- **Persistent Advice.** Interactive Reinforcement Learning enables non-experts to interact with agents and provide feedback and advice in an online environment. Current methods discard feedback shortly after each interaction. This thesis introduces *advice persistence* to Interactive Reinforcement Learning - methods for the retention and reuse

of advice for future decision making. With the use of persistence an agent can maximise the utility of each interaction, resulting in increased learning performance and a considerable reduction in the time required from the advice giver.

- **Rule-Based Interactive Reinforcement Learning.** Current Interactive Reinforcement Learning methods have limited advice delivery to the simplest forms of interaction, either boolean responses of evaluation or simple informative commands such as ‘Go Left’. The interactions also limit the advice to the current or previous state, to be forgotten soon after. This thesis contributes rule-based advice delivery. A rule-based delivery method that allows a human to provide advice, either evaluative or informative, that can generalise over many states and is available to the agent indefinitely. Using a structured approach to modelling the incoming advice, such as Ripple-Down Rules (Compton et al., 1991; Richards, 2009; B. Kang, Compton, & Preston, 1995), can allow the addition of new rules as exceptions to previously supplied advice, refining the generalisations made by the human in the past. Additionally, the use of rules as an advice delivery mechanism allows a significant reduction in the number of interactions between the user and the agent, and may facilitate a method in which non-experts can deliver advice quickly and easily.

1.3.1 Approach

This section briefly details the approach to be used to answer the principal question, and sub-questions, identified in Section 1.2.1. The general approach to answering the principal question presented in this thesis is to create a *constructive proof*: a functioning Interactive Reinforcement Learning agent that uses a rule-based advice delivery mechanism capable of managing incorrect information and advice from an advising user. This agent will be tested in multiple domains to gain an understanding of the extent to which human-sourced advice can accelerate its learning compared to standard Q-Learning Reinforcement Learning and existing Interactive Reinforcement Learning methods.

The approach to addressing sub-question (i) is to perform human trials. In these trials, ten participants will interact with an evaluative Interactive Reinforcement Learning agent, and a separate group of ten participants will interact with an informative Interactive Reinforcement

Learning agent. The users are required to assist the agent to a point where they believe the agent has learnt enough about the objective or until they feel like the agent is no longer learning. Additionally, they may continue assisting until they feel like they have had enough of attempting to train the agent. The results of these trials will show the accuracy of the users advice, how much advice they are willing to provide, how the properties of the advice vary over time, and how the agent’s performance is affected.

The approach to sub-question (ii) requires comparing the change in agent performance between agents that have humans advising them versus agents that have simulated users advising. Using the human trials performed as part of the approach to sub-question (i), an estimate of advice accuracy, consistency, and human commitment can be measured. Using these estimates, simulated users can be created that can replicate similar behaviours. These simulated users will be used to train the same type of agents from the human trials, and the performance between the two agents are compared.

The approach to determining the extent to which a rule-based model can be built iteratively that can handle incorrect advice, sub-question (iii), is to test such an agent with advice that is incomplete or does not align with the optimal policy. Multiple simulated users will be created with either low, medium, or high advice accuracy. Using different combinations of these simulated users, a rule-based agent will be advised on multiple domains and the performance changes of the agent analysed.

This thesis will review current best practice in provided external advice to a Reinforcement Learning. This will involve the development of a taxonomy and framework for Assisted Reinforcement Learning. Assisted Reinforcement Learning is an approach to Reinforcement Learning that utilises external information, either before, during, or after training, to improve performance. Interactive Reinforcement Learning is an example of Assisted Reinforcement Learning. The approach to the creation of the framework and taxonomy has been to identify fields of Reinforcement Learning that utilise external information to assist an agent in learning, identify how the information is sourced and utilised, and to build a framework that can accommodate the different fields. The result is a framework and taxonomy that can be used to define and describe agents that utilise external information with an aim to facilitate collaboration and communication between the different fields.

More details about the investigative approaches can be found in each questions relevant chapters. To find/determine which chapters pertain to each contribution refer to Section 1.4.1.

1.4 Readers Guide

This section gives a general description of each chapter, as well as a table showing the correspondence between the contributions and each chapter.

- **Chapter 2 - Reinforcement Learning** introduces some background information about Reinforcement Learning and Interactive Reinforcement Learning, and briefly describes some of the current challenges and research in each of the fields. This chapter also expands on the motivation for this thesis.
- **Chapter 3 - Assisted Reinforcement Learning** introduces the Assisted Reinforcement Learning framework and taxonomy. This chapter consists of an exegesis and a submitted paper. The paper also supplements the literature review provided in Chapter 2, providing information on other related, but less relevant, fields of externally-influenced Reinforcement Learning.
- **Chapter 4 - Experimental Methodology** provides information about the environments in which agents are evaluated in. Each environment is described generally in this chapter, and the specifics are provided in the chapters in which they are used. This chapter also details the testing methodologies and architectures used.
- **Chapter 5 - Human-Sourced Advice** presents evaluative and informative advice, two methods of human-advice giving. The chapter also details the methodology and results of a human trial comparing the two advice types and shows how each affects human engagement, accuracy, and availability.
- **Chapter 6 - Simulated Users** describes the problems faced when performing human trials, and evaluates simulated users as an alternative to human testing. The chapter investigates the extent to which simulated users can adequately evaluate the behaviour of an agent.

- **Chapter 7 - Persistent Advice** investigates the use of persistent advice, a method of retaining and reusing advice provided by the human, rather than the human having to continually provide transient advice.
- **Chapter 8 - Rule-Based Interactive Reinforcement Learning** presents a rule-based advice method for Interactive Reinforcement Learning. The chapter details the rule-based method for advice delivery, retention, and utility, and benchmarks performance against other Reinforcement Learning agents.
- **Chapter 9 - Conclusion** summarises the contributions of this thesis, details some of the limitations of the research, and outlines some areas for future work.
- **Appendices** provides additional information regarding the ethics application and trial questionnaire.

1.4.1 Contribution/Chapter Correspondence

CONTRIBUTION	CHAPTER					
	2	3	5	6	7	8
ASSISTED REINFORCEMENT LEARNING	*	*	+	—	+	+
EVALUATIVE / INFORMATIVE ADVICE	*	*	*	—	*	+
SIMULATED USERS	*	*	—	*	*	*
PERSISTENT ADVICE	*	*	—	—	*	*
RULE-BASED LEARNING	*	*	*	*	*	*

* : ESSENTIAL; + : RELEVANT; — : IRRELEVANT

Table 1.1: Correspondence between the thesis contributions and the chapters.

Chapter 2

Reinforcement Learning

2.1 Introduction to Reinforcement Learning

Reinforcement Learning is an area of Machine Learning whereby artificially-intelligent agents learn behaviours through interaction with their environments (Sutton & Barto, 1998; Kaelbling et al., 1996). Reinforcement Learning agents learn by trial-and-error, repeatedly interacting with the environment and learning which actions lead to desired outcomes and which do not. Reinforcement Learning was first identified in behavioural psychology (Skinner, 1990) where animals, or agents, were observed to repeat actions that lead to positive outcomes and avoid actions that don't. Decades later, behavioural Reinforcement Learning has been extended to machines, where agents are given numerical rewards rather than biological stimuli, to indicate an action's value. High numerical rewards are given for performing desired behaviours, and low for undesirable behaviours. The goal of the agent is to maximise the numerical reward it receives while interacting with an environment.

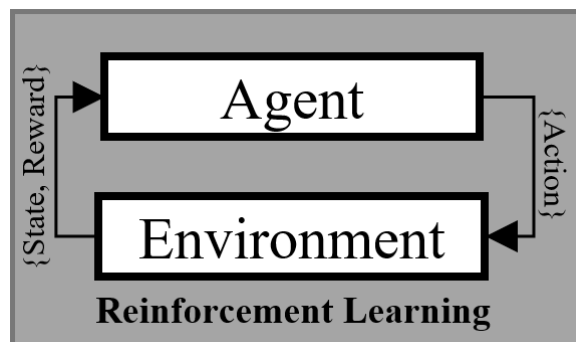


Figure 2.1: The traditional Reinforcement Learning framework.

An agent operates within an environment. At any a given point in time the agent will have an observation of the environment and will use this information to decide on an action

to take. After executing this action, the agent is given a new snapshot of the environment, and may also be given a reward as a measure of the previous action's benefit to the objective. The goal of the RL agent is to maximise this reward over the long term.

A distinguishing strength of Reinforcement Learning is its ability to learn without prior information. Reinforcement Learning's blank-slate approach allows full and accurate behaviours to be learnt without the agent having any prior knowledge about the environment's dynamics or what the desired behaviour is (Bellman, 1957; Howard, 1964). It is distinct from supervised learning methods, in that clear and correct examples are not required to learn a behaviour, rather, a reward function is used to influence the behaviours learnt. The reward function is the mathematical formulation of the agent's objective.

The fundamental elements of Reinforcement Learning are states, actions, and the reward. A state is a collection of available information that is observable at a given point in time. For example, at any point in time the state information that is available to the driver of a car may include its speed, position, and nearby obstacles. The state space, also referred to as the environment, is the collection of all possible states the agent, i.e. car driver, may encounter.

An action is any function that the agent may perform that may have an impact on the environment or itself. An action is chosen by the agent based on the information given in the current state. For example, the driver of a car has the actions of changing the car's velocity (by accelerating or braking), or changing the car's position (by steering left or right). The set of possible actions that an agent may execute is the action space. The action space may differ between states.

The aim of the agent is to learn which is the best action to take for each state. The mapping of actions to states is known as the agent's policy or behaviour. The best action to take will be the action that returns the best reward over time. In chess it may be necessary to sacrifice some pieces early on to win the game overall. The same applies to Reinforcement Learning agents; the agent will attempt to maximise the reward over its lifetime, even if it must incur temporary penalties.

Reinforcement Learning is suitable for learning tasks that can be modelled as Markov Decision Processes (MDP) (Bellman, 1957; Sutton & Barto, 1998; Kaelbling et al., 1996). A MDP is specified by the tuple (S, A, T, R, γ) , which represents the set of states in the

environment, S , the set of actions available in each state, A , the transition function $T : S_n \times A \rightarrow S_{n+1}$, a reward function $R : S \times A \rightarrow R$, and a discount factor $0 \leq \gamma \leq 1$. The goal of a Reinforcement Learning agent is to learn a mapping of states known as the policy, $\pi : S \rightarrow A$, which maximises the expected reward received from the environment. The discount factor determines the agent's preference for future versus immediate rewards, with the higher the discount factor the more priority future rewards have.

A Reinforcement Learning agent also has a learning rate, $0 \leq \alpha \leq 1$, that determines the step size for the agent's optimization. The larger the learning rate the faster the agent will learn but the resulting behaviour may not be optimal. Small learning rates will result in an optimal behaviour but learning may take much longer. A balanced constant, or decaying learning rate is sufficient for most agents.

Example: A Reinforcement Learning Car

Figure 2.2 depicts a Reinforcement Learning agent in control of a car. The car has two sensors that each activate when an object is nearby, one on the left and on the right of the car. The number of possible states for the environment is 4 (2 state variables, each with 2 possible values). The possible states are shown in Table 2.1

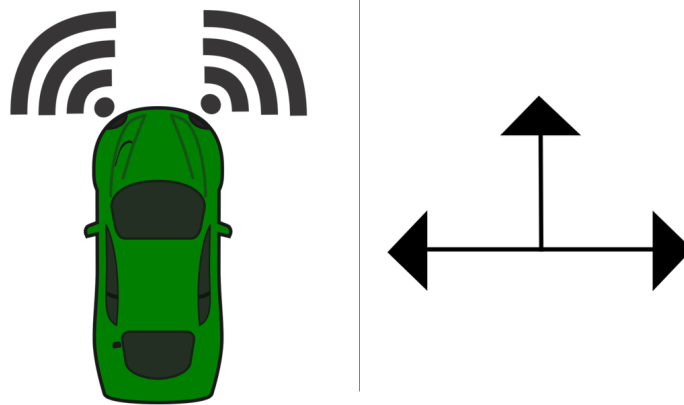


Figure 2.2: A simplified example of a self-driving Reinforcement Learning agent/car. The car has a sensor on each side that turn on when there is an obstacle on the respective side. The car can choose to turn left, right, or move forward.

A reward function for the car is designed to influence the behaviour the agent will learn. The reward function might reward the agent a -1 for each time step that a sensor is detecting

an obstacle, for each sensor. The reward function might also reward the agent a -100 each time it collides with an obstacle. The large negative reward will discourage the agent from entering circumstances in which a collision occurs, while the small ongoing negative rewards encourage the agent to keep its distance from the obstacles.

After sufficient learning, the car will learn a desired behaviour given the information and actions it has access to. For this scenario, the agent will learn that if the left sensor is on then it should turn right, and if the right sensor is on it should turn left. If both sensors are off then all actions are equally likely to be safe, and the agent can take any action with confidence. If both sensors are on it is likely there is an obstacle directly in front of the agent, and the agent should turn left or right with equal confidence about its safety. The possible state space, and the optimal actions for each state are shown in Table 2.1. If the agent had more possible actions, such as reversing, or if it had more information about the environment, such as more sensors or sensors looking further away from the car, then it could improve its behaviour and be less likely to be in a collision.

STATE.	RIGHT SENSOR.	LEFT SENSOR.	ACTION.
1	OFF	ON	LEFT
2	ON	OFF	RIGHT
3	ON	ON	LEFT OR RIGHT
4	OFF	OFF	FORWARD, LEFT, RIGHT

Table 2.1: Example policy of a Reinforcement Learning agent controlling a car.

2.2 The Challenges of Reinforcement Learning

There are a few concerns with Reinforcement Learning that may hinder its ability to learn. Some of these concerns stem from Reinforcement Learning’s trial-and-error approach to learning and its lack of labelled examples, and others are issues common to Machine Learning. While there are proposed conditional solutions to some of these issues, they still remain active research areas due to their importance to the performance of Reinforcement Learning. The contributions of this research aim to address, wholly or partially, the issues currently facing Reinforcement Learning.

2.2.1 Curse of Dimensionality

Coined by Richard Bellman in his work on dynamic programming, a precursor to computational Reinforcement Learning, the ‘Curse of Dimensionality’ refers to concerns arising from data with high-dimensional spaces (Bellman, 2013). The major concern that affects Reinforcement Learning is the exponential increase in the state space size as the number of dimensions increases. State space explosion is particularly large in environments with continuous observations. Reinforcement Learning agents learn by interacting with the state space; the larger the state space, the more time required by the agent to learn the optimal behaviour.

To free an agent from the curse of dimensionality the number of dimensions or the number of states need to be reduced (Sutton, 1996). This is typically done by collapsing dimensions down upon each other or reducing the granularity of the dimensions, potentially ignoring information that is of little to no importance to the agent’s learning (Sutton, McAllester, Singh, & Mansour, 2000; Stone, Sutton, & Singh, 2000). The main concern with these approaches is the identification of states and dimensions that offer little information to the agent, often requiring specialized knowledge and function approximation.

2.2.2 Time Requirements

The time required by a Reinforcement Learning agent is affected by two factors, the size of the environment and the agent’s information gain. The agent will continue exploring the environment until every state/action pair has been visited multiple times, or a suitable behaviour has been learnt. It often requires multiple visits to the same state, performing the same action, to accurately estimate the potential future reward for each state-action pair ¹. Reducing the number of state-action pairs, as described in Section 2.2.1, or by increasing the agent’s information gain will reduce the time required by the agent. The agent’s information gain is the rate in which the agent can learn the correct expected future rewards for a given state-action pair.

¹Formal convergence proofs rely on each state-action pair being experienced an infinite number of times.

Methods for improving an agent’s information gain is a large research focus. Some methods include:

- Initializing the expected future reward to a value closer to the actual value. This requires intimate knowledge about the problem, either from a human or a knowledgeable agent (Taylor & Stone, 2009).
- Provide the agent with additional information about the environment through supervision or additional reward signals (Knox & Stone, 2010; Suay & Chernova, 2011; Randsjøv & Alstrøm, 1998).
- Improving the agent’s exploration policies, resulting in a better exploration-exploitation trade-off (Sutton & Barto, 1998; Fernández & Veloso, 2006).
- The use of options (Sutton, Precup, & Singh, 1999); a series of actions taken over time.

2.2.3 Behaviour Transfer

The cost of training a Reinforcement Learning agent can be significant, both in terms of agent training time and, in the case of interactive learning methods (Knox & Stone, 2010; Suay & Chernova, 2011; Thomaz et al., 2005), human supervision time. For this reason, it would be ideal if agents did not need to relearn complete behaviours from scratch when moving to new environments. Transfer learning is a large field of Reinforcement Learning whose focus is on creating methods for transferring information learnt from one domain to a new, but similar, domain (Taylor & Stone, 2009; Sharma et al., 2007). Current methods require a domain expert to connect the related areas between the new and the old domains.

2.2.4 Goal Misalignment and Concept Drift

Goal misalignment refers to circumstances when the agent’s goal and the goal of the designer differ. This is commonly caused by an incorrect reward function, where the agent and human understandings of the function do not match. For example, the human may design a reward function for a self-driving car/agent that gives a large negative reward when the car crashes, expecting that the car will learn to drive without crashing. The agent may learn from this, however, that it should not drive the car at all as that is the safest method for not

crashing, a behaviour the human did not intend or desire. One solution to a misunderstood reward function is to redesign it with more conditions, however, this can be time-consuming and may require the agent to begin learning all over again.

Concept drift refers to the process in which the objective of the agent and the humans intention's begin to misalign over time (Schatzmann, Weilhammer, Stuttle, & Young, 2006). This may require the reward function to be redesigned.

2.3 Interactive Reinforcement Learning

As with most machine learning techniques, Reinforcement Learning struggles to learn in large state spaces. As environments become larger the agent's training time increases and finding a solution can become impractical (Cassandra & Kaelbling, 2016). In the mid-nineties, as Reinforcement Learning was transitioning from toy problems to real-world tasks, Kaelbling et al. (Kaelbling et al., 1996) argued that if the field is to succeed in scaling then the use of information external to the environment would be needed.

In the decades since the Kaelbling et al. (Kaelbling et al., 1996) survey, different streams of Reinforcement Learning have emerged, many using external information to assist either the process of generalising the environment, reinforcement agent decision making, or to provide focused exploration. Reinforcement Learning fields that utilise external information now make-up a sizeable portion of the Reinforcement Learning literature. One such field is Interactive Reinforcement Learning.

Interactive Reinforcement Learning (IntRL) is an extension to Reinforcement Learning in which the agent is assisted by real-time interaction with a human or coach (Thomaz et al., 2005). IntRL methods allow a human to supervise the agent and change the reward an agent receives or the action it performs. However, the human also retains the option to not provide any assistance, during which time the agent will default to its normal Reinforcement Learning behaviour. The motivation of the field is to improve agent learning speed and scale agents up to more complex problems by transferring knowledge from humans to agents by easy-to-use methods, while retaining the benefits Reinforcement Learning has over fully supervised learning.

Interactive Reinforcement Learning attempts to address the previously raised issues of

Reinforcement Learning with the use of advice. Users providing advice can assist the agent by providing information to guide exploration efforts. This helps to reduce the search space and the time required to learn the optimal policy. Advice, whether it is evaluating past performance or informing future decision making, allows the user to transfer their own expertise to the agent.

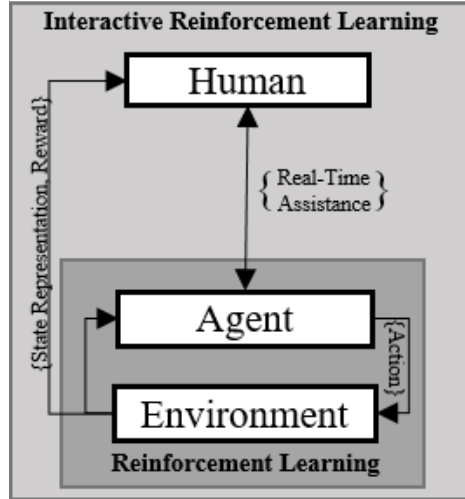


Figure 2.3: Interactive Reinforcement Learning.

One premise that underpins Interactive Reinforcement Learning is that humans have the ability to learn and store large volumes of information, and that this information can be useful to the learning of a Reinforcement Learning agent. Information that may seem simple to humans, when provided to the agent, can provide significant increases in an agent's learning speed. For example, the simple knowledge of which side of the field the goal is at in the soccer domain, or which direction the objective is (Randløv & Alstrøm, 1998), can greatly improve the agent's performance.

2.3.1 Current Research

An early and highly recognised piece of research in Interactive Reinforcement Learning, extending from social learning, is *Sophie's Kitchen*, an environment in which an interactive agent attempts to bake a cake with the assistance of a human (Thomaz et al., 2005). The aim of this work was to understand the role real-time human interaction can play in the development of an agent's abilities. In the experiment, the agent's goal is to prepare a cake

by performing the correct actions with the correct items in the correct order. At any point during the experiment an observing human can reward the agent with a value between -1 and 1. This reward acts as an additional reinforcement, and with the use of reward shaping (Ng, Harada, & Russell, 1999), guiding the agent’s exploration of the state space. This work found that the simple evaluative reward signal did improve the agent’s learning but also raised other issues. It was observed that humans tended to provide anticipatory reward, with the intention of persuading the agent to/from doing an action. However, the design of the agent meant that the anticipatory reward would instead encourage/discourage the previously chosen actions rather than the action to be chosen next.

Later work attempted to address this quirk of human teaching by introducing delayed rewards. TAMER-RL (Knox & Stone, 2010, 2009, 2008), is an Interactive Reinforcement Learning agent that operates much the same as the agent from Sophie’s World, with the exception that the additional reward signal from the human is spread across multiple states, both forward and back in time (Ng et al., 1999). The aim of this research was to address two concerns of Interactive Reinforcement Learning, that humans may provide anticipatory advice, and that humans may be slow in providing the reward resulting in the wrong state being addressed. The TAMER-RL agent has been benchmarked on a wide variety of environments and has consistently out-performed standard Q-Learning agents.

An issue with using an additional reward shaping signal such as those used in Sophie’s World and TAMER-RL is that the expected future rewards that the agents learns may not align with the rewards given by the environment. This results in the estimates for the expected rewards never converging, as the human supplied reward directly affects the estimates and the additional reward signal can be highly volatile.

More recent research has focused on policy shaping as the delivery mechanism for user-supplied advice (Griffith et al., 2013). Unlike reward shaping, policy shaping does not directly alter the estimated expected future rewards that the agent learns; instead, the policy the agent follows is changed. Using this method, the agent can continue to learn the correct reward estimates. When it comes time to decide on the next action to take the human can step in and suggest an action, rather than letting the agent decide. Policy shaping has also allowed for more detailed advice to be given to the agent (Krening et al., 2017), as the advice can be

encoded as action recommendations and future options, rather than being constrained to a scalar reward signal as used in Sophie’s Kitchen and TAMER-RL.

2.3.2 The Challenges of Interactive Reinforcement Learning

Incorrect Advice

The mitigation of the risks that incorrect advice has on agent performance has not been an area of focus for Interaction Reinforcement Learning. Existing research has not investigated the impact of incorrect advice on the learning performance on an agent, instead, it is usually assumed that all user-supplied advice is always correct and relevant to the current situation. Existing methods do have some fault-tolerance inherent to their design, as the agent still retains control of action selection and learning when the human is not supplying advice. However, if these agents are to be deployed to real world environments then the levels of fault tolerance and the impact of the incorrect advice should be known beforehand. Assuming that incorrect advice negatively impacts agent learning, methods for the identification and handling of the advice should remain an active area of research.

The Burden of Assisting

When assisting an agent there is an active burden on the human resulting from the cost in time and skill. Research into how humans interact with machine learning agents show that the time humans are willing to spend assisting diminishes (Amershi et al., 2014), therefore, to improve Interactive Reinforcement Learning either the amount of time the human can assist for or the advice utility needs to increase.

The amount of time humans are willing to provide can be extended through methods such as incentives, making the process enjoyable (Khatib et al., 2011; Horowitz et al., 2016) or by making the task competitive with other people (Li, Hung, Whiteson, & Knox, 2013). Such methods have shown, sometimes considerable, increases in the engagement of the advising user. However, these methods often rely on groups of users or expensive and creative solutions. Alternatively, by making the process of providing information easier for the human, such as natural language processing (Krening et al., 2017), it allows for less stress on the user and may increase the time they are willing to provide. Additionally, the easier the information

is to provide, the more people may be available to provide it, reducing the amount of time required by any one person.

While the amount of time people can commit to training an agent is theoretically infinite², the better solution would be to reduce the need for the human over time. One such method is to improve the utility of the advice the agent has access to.

Low Advice Utility

Advice utility refers to the benefit an agent can gain out of a single interaction with a user. A shared trait of the existing research in Interactive Reinforcement Learning is that the agent has limited memory for each interaction(Griffith et al., 2013). The information gained by an interaction between the human and the agent is only used once, perhaps spread across a few states, and then discarded(Knox & Stone, 2010, 2008, 2009). The benefit of the interaction is retained, relevant to the state in which the interaction occurred. However, the interaction itself and the information it contained are quickly forgotten. While this does reduce the computational resources required by the agent, it does mean that ‘stubborn’ agents, agents that do not gain the full benefit of the interaction in the time provided, may need to be continually assisted regarding the issues.

For example, in the earlier example of a simple self-driving car. The human may observe the car’s right sensor has activated, indicating that there is an obstacle on the right. In response, the human advises the car to turn left. While this small piece of advice can have a large positive impact on the agent’s learning, the full benefit of the advice is lost. The agent does not retain the advice, so next time it is in a similar situation, the human may need to assist the agent again or have the agent forego assistance altogether.

A method for building and reusing a model of the user-supplied advice would be of value to the agent so that it can reference the model when uncertain about the current state. Such models exist in fields such as Transfer Learning, however, these models are not generated in real-time from user supplied advice but rather from agent experiences in previous domains. Such a model would allow the previously mentioned car to remember that when the right sensor activates, it should steer left, maximising the benefit of that single piece of information,

²Or limited to the time of their death, whichever comes first.

and reducing future demand of the user.

The delivery mechanism and structure of advice may also limit the utility of the information being provided. Structures and delivery mechanisms that allow generalised information, or information whose use is conditional and persistent, may improve the agent’s ability to process and apply it to the relevant situations.

2.4 Similar Domains

Other Reinforcement Learning fields worth mentioning are *Reinforcement Learning from Demonstration* and *Transfer Learning*. These fields both use information from external sources but do not have a strong focus on interactivity. Rather, these fields focus on the use of existing knowledge when designing and preparing the agents. While these fields may move to more interactive approaches in the future, the current research is not as pertinent to this project and did not warrant an in depth review, instead, summaries of the fields have been provided for the readers benefit.

Reinforcement Learning from Demonstration

Reinforcement Learning from Demonstration (RLfD) is similar to Interactive Reinforcement Learning, differing primarily on information structure and timing. RLfD allows a teacher to provide examples of how to perform a task before the agent begins training (Suay, Brys, Taylor, & Chernova, 2016; B. D. Argall, Chernova, Veloso, & Browning, 2009; Brys, 2016; Schaal, 1997; B. Argall, Browning, & Veloso, 2007) . The agent uses these examples to learn a behaviour that generalises or mimics the provided demonstrations. The teacher acts as a supplemental information source, offering advice to the agent that assists it in making decisions and guiding exploration. RLfD utilises the demonstrations in two ways, by altering the reward the environment gives the agent, or by modifying the agent’s decision making to take the demonstrated actions. Both of these alterations guide the agent to better mimic the demonstrations provided by the teacher.

Transfer Learning

Transfer Learning in Reinforcement Learning allows for agents to generalise information across tasks (Taylor & Stone, 2009; Taylor, Stone, & Liu, 2007; Sharma et al., 2007). Transfer learning assumes that training and testing environments may not have the same state space or distributions, but that behaviours learnt from one can be useful to the other. By identifying the relationships between domains, an agent can use known skills and behaviour to accelerate learning in new areas. The agent’s known skills and behaviour may be information it has gained itself from previous tasks or information given to it by other agents or sources about similar domains. The externally-sourced information that a transfer learning agent receives allows it to generalise known skills to new skills, often resulting in accelerated learning.

Assisted Reinforcement Learning

This chapter provided a brief introduction to the fields and challenges of Reinforcement Learning and Interactive Reinforcement Learning. More specific and relevant information is available in the respective chapters that follow. The next chapter, Assisted Reinforcement Learning, supplements the content of this chapter. However, the next chapter is intended for publication, irrespective of this thesis. As such, there may be sections of the following chapter that appear repetitive when read alongside this chapter.

Chapter 3

Assisted Reinforcement Learning

This chapter is written as a journal paper and may duplicate information described in earlier or later chapters. This paper, titled ‘*Assisted Reinforcement Learning: A Framework for Externally-Influenced Agents*’, will be submitted around the same time as this dissertation. This chapter supplements the information presented in Chapter 2, with some overlap in information as it is written as a standalone paper for publication.

During the literature review for this dissertation, many fields that use externally-sourced information were identified. The terminology used to describe the agents and techniques differed between the identified areas, complicating the literature review and making a comparison of the agents difficult. This chapter defines externally-influenced Reinforcement Learning and presents a framework aiming to improve comparability and collaboration in this research area.

There are multiple authors for this chapter. I would like to acknowledge the contributions of Associate Professor Peter Vamplew, Assistant Professor Matthew Taylor¹, and Dr. Tim Brys². The contributions are listed below.

CONTRIBUTIONS	
AUTHOR	SECTION CONTRIBUTED
ASSOCIATE PROFESSOR PETER VAMPLEW	3.3.5
ASSISTANT PROFESSOR MATTHEW TAYLOR	3.3.4
DR. TIM BRYs	3.2.7; 3.3.3

Table 3.1: Correspondence between authors and section contributions.

¹M. Taylor is with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164, USA, taylorm@eecs.wsu.edu

²T. Brys is with the VUB Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium timbrys@vub.ac.be

Assisted Reinforcement Learning: A Framework for Externally-Influenced Agents

Abstract

A long term goal of machine learning is to create agents that are capable of performing tasks in real-world situations. The use of external information is a method for scaling agents to more complex problems. This article identifies current streams of Reinforcement Learning that utilise external information to supplement agent learning and decision making. Some current methods include heuristic Reinforcement Learning, Interactive Reinforcement Learning, learning from demonstration, transfer learning, and multi-agent learning. These streams of Reinforcement Learning operate with a shared objective, however, a lack of collaboration exists between the streams. This article proposes an Assisted Reinforcement Learning Framework, aimed at fostering collaboration by simplifying classification and comparability of methods that leverage external information in the learning process. The framework details the relationship between external information and the agent, highlighting the process of information decomposition, structure, retention and how it can be utilised to influence agent learning. A brief analysis of current Assisted Reinforcement Learning methods and open questions are also presented, showing how each aligns to the framework.

3.1 Introduction

A long-term goal of machine learning is the creation of agents that are capable of functioning in real-world environments (Sutton & Barto, 1998). Reinforcement Learning has positioned itself as a solid candidate for such tasks. It owes its success to its ability to improve while operating, to learn without supervision, and adapt to changing circumstances (Kaelbling et al., 1996). The seminal Reinforcement Learning approach (Sutton & Barto, 1998) utilises an agent that interacts with an environment, learning by trial-and-error. The agent explores the environment and learns solely from the signals it receives (Figure 3.1). Early work using this foundational approach has shown success in domains such as inventory management (Giannoccaro & Pontrandolfo, 2002), robot soccer (Kitano et al., 1997), and game-playing (Tesauro, 1994).

As with most machine learning techniques, Reinforcement Learning struggles to learn in large state spaces. As environments become larger the agent’s training time increases and finding a solution can become impractical (Cassandra & Kaelbling, 2016). In the mid-nineties, as Reinforcement Learning was transitioning from toy problems to real-world tasks, Kaelbling et al. (Kaelbling et al., 1996) argued that if the field is to succeed in scaling then the use of information external to the environment would be needed.



Figure 3.1: Traditional Reinforcement Learning. Adapted from Sutton & Barto, 1998.

In the decades since the Kaelbling et al. (Kaelbling et al., 1996) survey, different streams of Reinforcement Learning have emerged, many using external information to assist either the process of generalising the environment representation, reinforcement agent decision making, or provide focused exploration. Reinforcement Learning techniques that utilise external

information now make-up a sizable portion of the Reinforcement Learning literature. Below are a few streams that focus on the use of external information in Reinforcement Learning.

- Interactive Reinforcement Learning (Amershi et al., 2014; Thomaz et al., 2005)
- Learning from Demonstration (Schaal, 1997; B. Argall et al., 2007)
- Transfer Learning (Taylor & Stone, 2009)

Other streams worth noting that use external information in a significant portion of their literature include:

- Heuristic Reinforcement Learning (Celiberto Jr, Ribeiro, Costa, & Bianchi, 2007)
- Multiple Information Sources (B. D. Argall, Browning, & Veloso, 2009; Nunes & Oliveira, 2003)
- Bayesian Reinforcement Learning (Vlassis, Ghavamzadeh, Mannor, & Poupart, 2012)
- Preference-Based Learning (Förnkrantz, Hüllermeier, Cheng, & Park, 2012)
- Inverse Reinforcement Learning (Abbeel & Ng, 2004a)
- Ensemble Algorithms (Wiering & Van Hasselt, 2008)

This article provides the following definition for ‘*External Information*’: Information provided to the agent originating from outside of the agent’s representation of the environment. This may include demonstrations (Suay et al., 2016; Brys et al., 2015; Schaal, 1997), advice and critiques (Knox & Stone, 2010; Griffith et al., 2013), initial bias based on previously gathered data (Taylor & Stone, 2009), or highly-detailed domain-specific shaping functions (Randløv & Alstrøm, 1998).

As the use of externally-influenced Reinforcement Learning agents continues to rise, it has become important to define the approach clearly. This article presents “*Assisted Reinforcement Learning*” (Figure 3.2), a framework and taxonomy to be used to describe the practice of using external information. A clear and shared taxonomy is important as it fosters collaboration between the various Reinforcement Learning communities, improves comparability, allows a precise description of new approaches, and assists in identifying and addressing key

questions for further research. The framework presented in this article, and the definition of “*external information*” on which it is based, are used to distinguish methods that employ external information from techniques such as the foundational view of Reinforcement Learning, state adaptation, dynamic programming, and fully supervised methods.

This paper defines “*Assisted Reinforcement Learning*” (*ARL*) as an approach to Reinforcement Learning that utilises external information, either before, during, or after training, to improve performance. Assisted Reinforcement Learning techniques have a shared objective: to employ existing sources of information to improve agent training and scale to larger and more complex problems. While a defining trait of Reinforcement Learning is its ability to learn behaviours from a blank slate, Assisted Reinforcement Learning makes use of existing information and learnt behaviours. Some methods of improving agent performance using external information include:

- Direct altering of weights for actions and states (biasing) (Vlassis et al., 2012)
- Altering the state or action space (Erez & Smart, 2008)
- Critiquing past or future decision making (Thomaz & Breazeal, 2007)
- Dynamically altering reward functions (Knox & Stone, 2010)
- Directly modifying the policy (Griffith et al., 2013)
- Guiding exploration and action selection (Fernández & Veloso, 2006)
- Creating information repositories/models to supplement the environmental information (Price & Boutilier, 2003)

The next section presents the Assisted Reinforcement Learning Framework, a method for formalising and describing ARL techniques. This section also briefly discusses the similarities between existing methods, and why providing a shared framework for representing and developing these techniques is important.

There are many implementations of Reinforcement Learning that utilise external information in existing literature. Section 3.3 looks at current research and applies it to the presented Assisted Reinforcement Learning Framework. This section aims to provide context

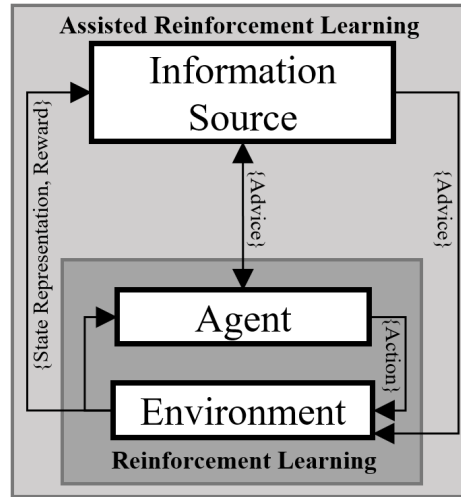


Figure 3.2: Assisted Reinforcement Learning.

to and examples for the use of the framework for the collaboration and comparison of ARL techniques. Finally, section 3.4 offers a brief discussion on the current state of ARL and some of the open questions in the field.

3.2 Assisted Reinforcement Learning

Reinforcement Learning’s strength is its ability to learn behaviour given no initial knowledge about the environment. With an appropriate reward function and enough interaction with its environment, a Reinforcement Learning agent can learn optimal behaviour. The agent’s current behaviour is defined by its policy; a solution to the reward function. The reward function’s sole responsibility is to promote desirable behaviour and penalise undesirable behaviour. In the traditional view of Reinforcement Learning, the reward function, and the rewards it produces are internalised by the environment. This approach, in which the environment is the sole provider of information to the agent performs well in small, bounded problems. However, it has difficulties when scaling up to large, unbounded environments. This issue of scaling is not uncommon in machine learning. In Reinforcement Learning, one approach to tackling the problem of scaling is to use external information to supplement the information that the environment provides (Suay & Chernova, 2011).

Information is considered external if it originates from outside of the agent’s interactions with the environment. Internal information is information that the agent can determine

solely through its interactions and observations with the environment. If a human is thought of as an agent, internal information would be anything the person can observe from the environment using their senses. The external information would be any information provided by peers, the internet, books, maps, and tutelage. For example, if a person eats some berries and later becomes sick they may determine that the berries are poisonous, this would be internal information. If instead, a peer advises the person that eating berries will make them sick, this would be external information.

Assisted Reinforcement Learning techniques utilise externally-sourced information to supplement the agent’s learning. Some common practices include the direct alteration of the agent’s understanding of the environment, focusing exploration efforts through critique and advice, or assisting the agent in decision-making. Existing Assisted Reinforcement Learning techniques include Learning from Demonstration, Interactive Reinforcement Learning, and Transfer Learning. In this section, a small summary of each is provided. A more detailed summary can be found in Section 3.3.

Interactive Reinforcement Learning (IntRL) extends the foundational Reinforcement Learning approach by involving an assistant directly into the agent’s training. An assistant, human or otherwise, can critique the agent’s decision making and advise on which actions to take in the future. The assistant provides their input at any time during the agent’s training, allowing them to respond to their own and the agent’s ongoing observations of the environment. By critiquing past actions or advising future decision making, both the agent and the assistant can quickly respond to changing environmental conditions (Suay & Chernova, 2011).

Reinforcement Learning from Demonstration (RLfD) is similar to Interactive Reinforcement Learning, differing primarily on information structure and timing. RLfD allows a teacher to provide examples of how to perform a task before the agent begins training. The agent uses these examples to learn a behaviour that generalises or mimics the provided demonstrations. The teacher acts as a supplemental information source, offering advice to the agent that assists it in making decisions and guiding exploration. RLfD utilises the demonstrations in two ways, by altering the reward the environment gives the agent, or by modifying the agent’s decision making to take the demonstrated actions. Both of these alterations guide

the agent to better mimic the demonstrations provided by the teacher.

Transfer Learning in Reinforcement Learning allows for agents to generalise information across tasks. Transfer learning assumes that training and testing environments may not have the same state space or distributions, but that behaviours learnt from one can be useful to the other. By identifying the relationships between domains, an agent can use known skills and behaviour to accelerate learning in new areas. The agent’s known skills and behaviour may be information it has gained itself from previous tasks or information given to it by other agents or sources about similar domains. The externally-sourced information that a transfer learning agent receives allows it to generalise known skills to new skills, often resulting in accelerated learning.

In the previously discussed Reinforcement Learning fields there exists an external information source that is used to assist the agent. These are examples of Assisted Reinforcement Learning, methods that use external information to supplement agent decision making and learning. The external information source is most commonly a human or another agent. Regardless of the source, the use of external information has shown improvements in the agent’s ability and learning speed in each of the fields discussed. The next section presents a framework and taxonomy for Assisted Reinforcement Learning.

3.2.1 A Conceptual Framework for Assisted Reinforcement Learning

This section presents a conceptual framework and taxonomy for Assisted Reinforcement Learning. The “*Assisted Reinforcement Learning Framework*” is designed to improved classification, comparability, and discussion between different externally-influenced Reinforcement Learning methods. To achieve this aim, the framework has been built using insights and observations from the many assisted Reinforcement Learning sub-fields. The result is a framework that can describe existing methods while also being flexible enough for future research. The framework is shown in Figure 3.3, with the accompanying taxonomy presented in Figure 3.4.

In Figure 3.4, there are four processing components denoted as angular rectangles, and three communication methods denoted as rounded rectangles. The processing components are responsible for providing, transforming, or storing information. The four processing

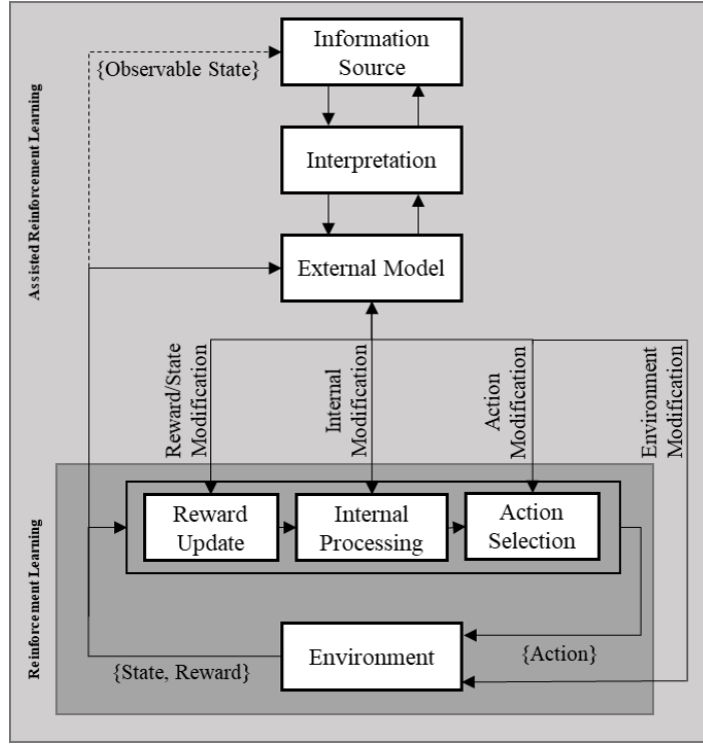


Figure 3.3: Detailed view of the Assisted Reinforcement Learning Framework.

components are depicted in both of the Assisted Reinforcement Learning framework diagrams. The communication components convey information or denote constraints on the data such as where or when to provide information. The communication components in Figure 3.4 that sit between the processing components represent the communication lines in Figure 3.3 that link the processing components together.

The Assisted Reinforcement Learning Taxonomy describes the transmission, modification, and modality of sourced information as it applies to the framework. The categories of the taxonomy are:

- **Information Source:** The information source is the origin of the assistance being provided to the agent. The source may be a human, repository, or another agent. There may be multiple information sources providing assistance to an agent.
- **Temporal:** The temporal component denotes both the time at which information is provided to the agent, and the frequency in which it is provided. Information may be provided, before, during, or after agent training, and occur multiple times throughout

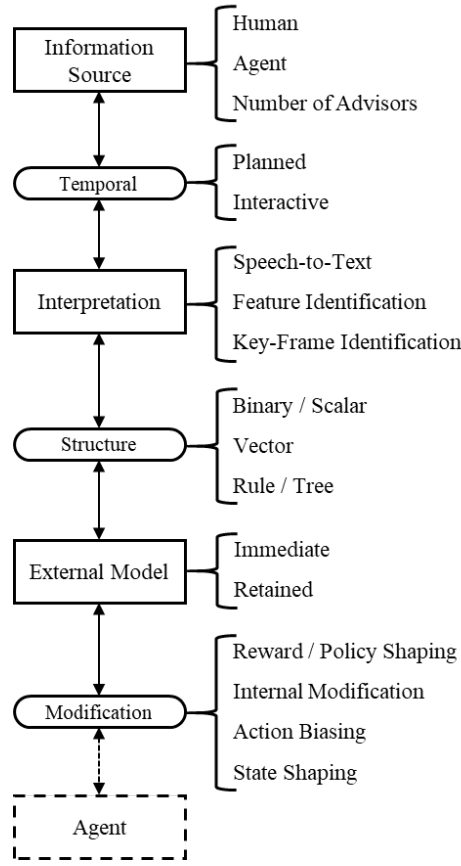


Figure 3.4: The Assisted Reinforcement Learning Taxonomy.

the an experiment.

- **Interpretation:** Denotes the process of transforming incoming information into a format better suited for the agent. This may involve extracting key frames from video, converting audio samples to rewards, or mapping information to states.
- **Structure:** The structure component describes how the incoming advice from the information source, and the information in the external model, is represented.
- **External Model:** The external model is a router for the information between the source and the agent. The agent may retain the information it receives in the model, using it for later decision making, or it may dump information it receives as soon as it has been used.
- **Modification:** The modification component denotes the approach that the agent uses to leverage the incoming information. The most common modification approaches are

reward shaping, using the information to alter the environmental rewards, and policy shaping, altering the agent’s behaviour or decision making directly.

- **Assisted Agent:** The Reinforcement Learning agent or agents that receive the external information.

The seven components are described in detail in the sections below.

3.2.2 Information Source

The information source is the defining factor that sets assisted Reinforcement Learning apart from other Reinforcement Learning approaches. It is responsible for introducing new information about the current task to the agent, supplementing the information the agent receives from the environment. The source is external to the agent and the environment, providing information that the agent may not have had access to, or would have eventually learnt itself. The information source may be able to observe the environment, the agent, or the agent’s decision-making process. The objective of the information source is to assist the agent in obtaining its goal.

There may be multiple information sources communicating with an agent. This may be many humans, agents, digital sources, or any combination of the three. The use of multiple sources offers a wider range of available information to the agents. However, more complex modification may be required to manage the information and handle conflicting advice (Isbell, Kearns, Kormann, Singh, & Stone, 2000)(Kamar, Hacker, & Horvitz, 2012).

There are many examples of external information sources in current Assisted Reinforcement Learning literature, the most common of which are humans and additional reward functions. Reinforcement Learning from Demonstration and Interactive Reinforcement Learning utilise human guidance to provide the agent with a generalised view of the solution. In Reinforcement Learning from Demonstration, a person provides the agent with an example of the desired behaviour. In Interactive Reinforcement Learning, a human may critique an agent’s past decision making, reinforcing its behaviour for future operation. Alternatively, the human may provide the agent with recommended actions to take in the future, guiding its behaviour and controlling exploration.

The use of additional reward functions is one of the earliest examples of Assisted Reinforcement Learning. In such cases, the overseer of the agent encodes some of their information about the environment or goal as additional rewards, supplementing the rewards given by the environment. An example of this can be seen in Randlov and Alstroms Bicycle experiment (Randløv & Alstrøm, 1998; Ng et al., 1999), in which, the team attempts to teach an agent to ride a bicycle towards a goal point. Without additional assistance, the Reinforcement Learning agent would only receive a reward upon reaching the termination state. The team encoded some of their information as a reward signal external to the environment, providing the agent with a reward, supplementing the environmental reward, if it is cycling towards the goal point. In this scenario, the system designers acted as an external information source, providing extra information to the Reinforcement Learning agent. The use of this external information results in the agent learning the solution faster.

Some other information sources include behaviours from past experiences or other agents, repositories of labelled data or examples, or distribution tables for initialising/biasing agent behaviour. Video, audio, and text sources may be used. However, these sources may require substantial amounts of interpretation and preprocessing to be of use.

The accuracy, availability, or consistency of the information source can affect the maximum utility of the information. Identifying inaccurate information before it is given to the agent can significantly improve performance. While the information source may perform validation and verification, the primary duty remains simply to act as a supplementary source of information. Validation and verification of information are functions better suited for the external model or agent.

3.2.3 Temporal

The temporal component, or ‘temporality’, refers to the time at which information is communicated by the information source. The information may be provided in full to the agent at a set time, either before, during, or after training. This is referred to as ‘planned assistance’. Alternatively, the information may be provided at any time during the agent’s operation, referred to as ‘interactive assistance’.

Planned assistance is common in assisted Reinforcement Learning methods. Additional

shaping functions, agent policy initialisation based on prior experiences or known distributions, and creation of sub goals that lead the way to a final solution are examples of planned assistance. These methods let the experiment designer endow the agent with initial information about the environment or the goal to be achieved. By providing this initial knowledge, the designer can narrow the agent’s need for exploration or focus the area of exploration.

The bicycle experiment discussed in section 3.2.2 is an example of planned assistance. In the experiment, the agent is learning to control a bicycle and must learn to steer it towards a goal. Without assistance, the agent only receives a reward if it reaches the goal state. In this experiment, the designers endowed the agent with additional information, a reward signal that correlates to the direction of the goal state. This planned assistance helps the agent to narrow the search space by giving it extra information about the environment. This small yet beneficial initial information results in a significant improvement in the agent’s learning speed.

Another example of planned assistance is found in heuristic Reinforcement Learning. Heuristic Reinforcement Learning is a method of applying advice to agent decision making. One experiment implements heuristic Reinforcement Learning to the RoboCup soccer domain, a domain known for its large state space and continuous state range (Celiberto Jr et al., 2007). In this environment, one team attempts to score a goal, while the other team tries to block the first team from scoring. In the experiment using heuristic Reinforcement Learning, the defending team is given some initial advice before training. This advice consists of two rules, if the agent is not near the ball then move closer, and if the agent is near the ball then do something with it. The experiment shows that a team that uses planned assistance performs better than a team that is given no initial knowledge.

Interactive assistance refers to information provided by the source repeatedly throughout the agent’s learning. Information may be provided repeatedly at any time before, during, or after training. Information sources that assist interactively often can observe the current agent’s state, or the environment the agent is operating in. In current literature, humans are more commonly used as information sources for interactive assistance. The human can observe how the agent is performing and its current state in the environment, and provide guidance or critique about the agent’s behaviour.

An example of interactive assistance is found in the Interactive Reinforcement Learning field. Sophie’s Kitchen (Thomaz & Breazeal, 2007) is an environment in which an agent, Sophie, attempts to bake a cake by interacting with the items and ingredients found in a kitchen. In an experiment using this environment, the agent tries to bake a cake and will receive a reward if successful doing so. At any point during the agent’s training, an observing human can provide the agent with an additional reward signal to supplement the reward signal given by the environment. If the agent performs an undesirable action, such as forgetting to add eggs to the cake, the human can punish the agent by providing an immediate negative reward. The human can also reward the agent for performing desirable actions, such as adding ingredients in the correct order.

In this experiment, the human is acting as an interactive information source. The agent could learn the task without any assistance; however, the addition of the human and their interactive advice allows the agent to learn the desired behaviour faster.

The benefit of using interactive advice rather than planned advice is that the source can react to the current state of the agent. Additionally, an interactive information source does not need to encode all possibly useful advice upfront, instead, choosing to provide relevant information only when required. This approach does have a significant cost; the information source needs to be constantly observing the agent and determining what information is relevant. Alternatively, the full repository of information could be made available to the agent before starting training. This approach lets the agent determine what information is relevant and when it should be used.

3.2.4 Interpretation

The interpretation stage of the framework denotes what transformations need to occur on the incoming information. The source provides information for the agent to use; this information may need to be translated into a format that the agent can understand. The information source may assist the agent in any number of forms. Some examples of the form information may take include audio and video, text, distributions and probabilities, or prior learnt behaviour from a different task or agent. This information needs to be adapted for use by the agent for the current task. The product of the interpretation stage depends on the

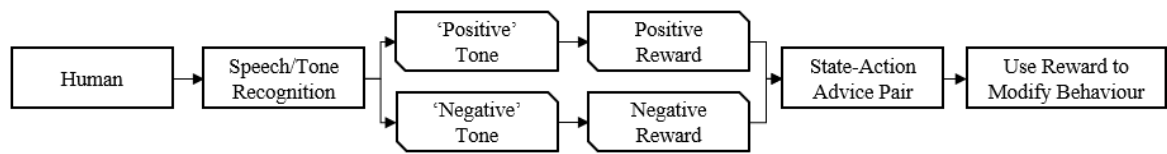


Figure 3.5: Audio interpretation example. Process for converting audio cue into a state-action advice pair.

structure that the agent or external model requires.

A field where the interpretation of incoming advice is crucial is transfer learning. The goal of transfer learning is to use behaviour learnt in a prior task to improve performance in a new, previously unseen task. A critical step in transfer learning is the mapping of states and observations between the old and new domains. The information source is providing information to the agent that does not fully align with its current task. It is crucial that the information provided can be sufficiently interpreted, so it is useful to the current domain. More commonly, this interpretation stage in transfer learning is performed by hand. However, more recent literature is attempting to automate this stage (Taylor, Kuhlmann, & Stone, 2008).

Another example of the use interpretation stage is with the sourcing of feedback for Reinforcement Learning agents. In Sophie’s Kitchen, the experiment discussed in Section 3.2.3, the agent can be given positive or negative feedback by a human regarding its choice in actions. In this experiment, the human draws a green (positive) or red (negative) box to represent the desired feedback to be given to the agent. These boxes are used to interpret the reward signal to give to the agent, with the colour of the box designating whether the reward is positive or negative, and the size of the box designating the magnitude of the reward. This type of feedback can be extended to audio (Cruz, Twiefel, Magg, Weber, & Wermter, 2015), where recording such as ‘Good’ or ‘Well Done’ is interpreted as positive rewards and cues like ‘Bad’ or ‘Try Again’ are interpreted as negative rewards (Tenorio-Gonzalez, Morales, & Villaseñor-Pineda, 2010). See Figure 3.5 for an example of interpreting audio to advice for an agent.

3.2.5 Structure

The structure component refers to the form that the agent or external model requires incoming information to take. The information that the agent uses can be represented in a number of ways. Some examples of assistance structure include Boolean values denoting human feedback, rules determining action selecting, matrices for mapping prior experiences to new states, case-based reasoning structures for the agent to consult with, or hierarchical trees to represent options for the agent to take (Kaplan, Oudeyer, Kubinyi, & Miklósi, 2002).

The simplest form of structure is binary, in which, the information takes one of two options. An example of the use of a binary structure is the TAMER-RL agent (Knox & Stone, 2010). TAMER-RL is an Interactive Reinforcement Learning agent that uses binary feedback from an observing human. The human can, at any time step, agree or disagree with the agent about its last action. This feedback, agree or disagree, is a binary structure.

A more complex structure is used in case-based Reinforcement Learning agents (Sharma et al., 2007). A case in case-based Reinforcement Learning represented a generalised area of the state-space and provides information about which actions to take in that state. The use of a case-based structure may allow the agent to gain more information from the information source compared to a binary structure, at a cost of more complex sourcing and interpretation approaches.

One of the more common structures for incoming advice is a simple state-action pair. A state-action pair consists of a single state, and an associated piece of advice. The associated advice may be an additional scalar reward or recommended action. Using a state-action pair, sourced information is interpreted to provide advice for a given state. In the bicycle experiment, discussion in sections 3.2.2 & 3.2.3, the information source is a reward function. This reward function observes the current state in the environment, and determines a reward to be associated with that state that correlates with the current direction of the goal. This state-action structure has also been used for other methods including TAMER-RL (Knox & Stone, 2010), Sophie’s Kitchen (Thomaz & Breazeal, 2007), and policy shaping approaches (Griffith et al., 2013).

3.2.6 External Model

The external model is responsible for retaining and routing information between the information source and the agent. The model receives interpreted information from the information source and may either retain the information for use by the agent when required or pass it to the agent immediately. The purpose of the model is to supplement the agent’s observations with information it has received externally.

A model may retain information provided by the information source, or it may pass it directly to the agent. An immediate model, one that passes information directly to the agent, may do so because the information received is only relevant to the current time step, or if the cost of reacquiring the information from the source is less than that of retaining the information. An example of an immediate model is the bicycle experiment discussed in Section 3.2.2. In this experiment, the information source is a readily available reward function. As the reward function is always available, there is no need to retain the information that it provides, as it is simple to require.

A retained model is an external model that stores all information provided by the information source. A retained model may be used if the cost of acquiring information is greater than the cost of storing it, if the information provided is general or applies to multiple states, or if the information is gathered incrementally. In instances where information is gathered incrementally, using a retained model allows the agent to build up a knowledge base over time. The agent may consult with the model at any time to determine if a reward signal is to be altered, or if there is any extra information that may assist with decision making.

The external model may have different functions depending on its implementation. In inverse Reinforcement Learning, the external model is a substitute for the reward function (Abbeel & Ng, 2004a). Heuristic Reinforcement Learning hosts a model that stores rules and advice that generalise over sections of the state-space (Dorigo & Gambardella, 2014). In transfer learning, the external model may hold information regarding past experiences and policies from problems similar to the current domain (Taylor & Stone, 2009) (Banerjee, 2007).

3.2.7 Modification

The modification stage of the framework denotes how the information that the external model contains is used to assist the agent in achieving its goal. It is responsible for supplementing the agent’s reward, altering the agent’s policy, or helping with the decision making process.

A popular method for utilising external information into agent learning is shaping. Shaping is a common method for altering agent performance by modifying parameters in the learning process. Erez and Smart (Erez & Smart, 2008) propose a list of techniques in which shaping can be applied to Reinforcement Learning agents. These include altering the reward, the agent’s policy, environmental dynamics, and agent learning parameters.

The modification of the reward the agent receives is a straightforward method for influencing an agent’s learning. Known as ‘Reward Shaping’, the external information is used to bias the agent’s behaviour (Ng et al., 1999). Care must be taken to ensure that any modification of the reward signal remains zero-sum to avoid the agent exploiting the environment in ways that do not align with the desired goal. Recent work by (Harutyunyan, Devlin, Vrancx, & Nowé, 2015) shows how non-potential-based reward shaping can be transformed to potential-based. Shaping has also been used to alter state-action pairs (Wiewiora, Cottrell, & Elkan, 2003), for dynamic situations (Devlin & Kudenko, 2012; Harutyunyan, Devlin, et al., 2015), and for multi-agent systems (Mason, Mannion, Duggan, & Howley, 2016; Mannion, Duggan, & Howley, 2016; Devlin & Kudenko, 2011).

Policy shaping is the modification of the agent’s behaviour (Griffith et al., 2013). This modification can be done either by influencing how the agent makes decisions or by directly alter the agent’s learnt behaviour. A simple method of policy shaping involves forcing to take certain actions if advice from the information source has recommended them. This allows the external information source to guide the agent and take direct control over exploration/exploitation. Alternatively, the information source can choose to alter the agent’s behaviour directly by changing q-values or installing rules that override the actions for chosen states. This method of modification can improve agent performance rapidly, as it can give the agent partial solutions. If the policy is permanently modified with incorrect advice, then the agent may never learn a solution to the task.

Environmental modification is an indirect method for influencing a Reinforcement Learning agent. Altering the environment is not always achievable and may be a technique better suited for digital or simulated environments. Some examples of modifying the environment include altering or reducing the state space and observable information, reducing the action space, modifying the agent’s starting state, or altering the dynamics of the environment to make the task easier to solve.

The first environmental modification, reducing the state space, can speed up the agent’s learning as there is less of the environment to search. While the agent cannot fully solve the task with an incomplete environment representation, it can allow the agent to learn a basic behaviour. The environment can then be increased, allowing the agent to drift into the correct behaviour.

The second environmental modification, reducing the action space, is similar to the first. The agent’s available actions are limited, and the agent attempts to learn the best behaviour it can with the actions it has available. Once a suitable behaviour has been achieved, new actions can be provided, and the agent can begin to learn more complex solutions.

The third environmental modification, modifying the agent’s starting space, alters where in the environment the agent begins learning. Using this approach, the agent can begin training close to the goal. As the agent learns how to navigate to the goal the starting state is incrementally moved further away. This allows the agent to build upon its past knowledge of the environment.

The final environmental modification discussed in this section, altering the environment dynamics, involves changing how the environment operates to make the task easier for the agent to learn. By altering attributes of the environment such as reducing gravity, lowering maximum driving speed, or reducing noise, the agent may learn the desired behaviour faster or more safely. After the agent learns a satisfactory behaviour, the environment dynamics can be changed to more typical levels.

Internal modification is a method of altering the parameters of the agent that are essential to its learning. Parameters such as the learning rate (α), discount factor (γ), and exploration percentage (ϵ), are all internal to the Reinforcement Learning agent and may be altered to affect its performance. For example, if an advisor observes that an agent is repeating

actions and not exploring enough then the exploration percentage or learning rate may be temporarily increased. Internal modification is a simple method to implement. However, it can be difficult at times to know which parameters to adjust, and to what degree they are to be adjusted. These are just a sample of the modification methods currently used in Assisted Reinforcement Learning.

3.2.8 Agent

The final component of the Assisted Reinforcement Learning Framework is the Reinforcement Learning agent. A key importance of the framework is that the agent, in the absence of any external information, should operate the same as any Reinforcement Learning agent would. Given no external information, the agent should continue to explore and interact with its environment and continue to achieve its goal.

An in depth look at some Assisted Reinforcement Learning methods are provided in the next section. Here, each method is described as applied to the framework that has been presented in this section.

3.3 Forms of Assisted Reinforcement Learning

This section presents an in depth analysis of some popular Assisted Reinforcement Learning techniques. Each technique discussed is applied to the framework set out in section 3.2. These analyses are conducted on fields of assisted Reinforcement Learning using current literature in the respective fields for examples.

3.3.1 Heuristic Reinforcement Learning

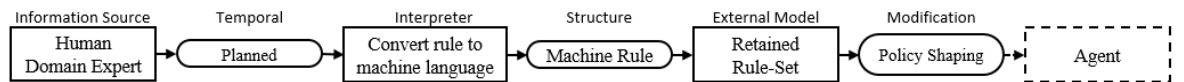


Figure 3.6: Heuristic Reinforcement Learning.

Heuristic Reinforcement Learning uses pieces of information that generalise over an area of the state space. The information is used to assist the agent in decision making and reduce

the searchable state space. An example of a heuristic is a rule. A rule can cover multiple states, making its use efficient at delivering advice to an agent.

One experiment applies heuristic Reinforcement Learning to the RoboCup soccer domain, a domain known for its large state space and continuous state range (Celiberto Jr et al., 2007). See section 3.2.3 for more information of heuristic Reinforcement Learning. The following is an analysis of heuristic Reinforcement Learning applied to the Assisted Reinforcement Learning Framework.

- **Information Source:** The information source for the RoboCup experiment is a human. In this case, the human has experimented with the robot soccer domain and can advise the agent with some rules that will speed up learning.
- **Temporal:** The advice for the agent is given before training begins. Once training has begun the human does not interact with the agent again. This is an example of planned assistance, where information is given to the agent at a fixed time, and the information is known by the information source ahead of time.
- **Interpretation:** The information needs to be understandable by the agent. In the robot soccer domain, the human gives two rules; if not near the ball then move towards the ball, and if near the ball do something with the ball. These rules are understandable by the human but need to be translated into machine code so that agent can use them. This is usually a task easily performed by a knowledgeable human operator. The result may resemble ‘IF NOT close_to_ball() THEN target_and_move()’ and ‘IF close_to_ball() THEN kick_ball()’.
- **Structure:** The structure of the advice after being interpreted will be a new rule. The rule will need to be compatible with the agent, including the ability to substitute variables and evaluate expressions.
- **External Model:** The external model used by a heuristic Reinforcement Learning agent such as the agent used in robot soccer domain will be a rule set. The model will retain all rules given to it. The model may also retain statistics about the rule relating to confidence, number of uses, and state space covered.

- **Modification:** Heuristic Reinforcement Learning uses the rule set to assist the agent in its decision making. If a rule applies to the current state, then the action that the rule recommends will be taken by the agent. This is a form of policy shaping as the agent’s decision making is directly manipulated by the external information.
- **Agent:** The Reinforcement Learning agent operates as usual. When it is time to decide on an action to take it consults the external model. The external model will test all the rules it has and checks to see if any apply to the current state. If a rule applies to the current state, otherwise the agent’s default decision-making mechanism is used.

3.3.2 Interactive Reinforcement Learning

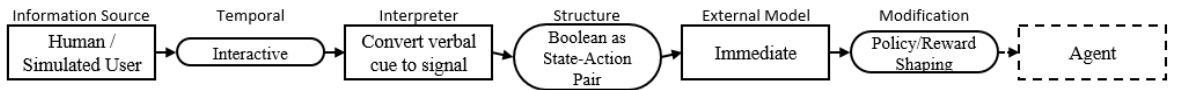


Figure 3.7: Interactive Reinforcement Learning.

Interactive Reinforcement Learning is an application of Assisted Reinforcement Learning that employs an information source (Thomaz et al., 2005). Most commonly, the information source is an observing human or a substitute for a human, such as an oracle or simulated user. The human provides assessment and advice to the agent, reinforcing the agents past actions and guiding future decisions. The human can assess past actions in two ways, by stating that the agent’s chosen action was correct or incorrect, or by telling the agent what the correct action to take was in that instance. Alternatively, the human can advise the agent on what actions to take in the future. The human can recommend actions to take or to avoid, or provided more information about the current state to assist the agent in its decision making.

Notable Interactive Reinforcement Learning applications include having a human provide additional reward information (Knox & Stone, 2010), and having a human or agent provide action advice (Torrey & Taylor, 2013; Taylor, Carboni, Fachantidis, Vlahavas, & Torrey, 2014; Zhan, Ammar, & Taylor, 2016; Amir, 2016; Griffith et al., 2013). All of these methods are real-time. All operate similarly, differing in the modification stage, and are good representations of the other work in the Interactive Reinforcement Learning field. The following

applies these methods to the Assisted Reinforcement Learning Framework.

- **Information Source:** The information source is a human or simulated user. A simulated user is a program, analogous to a human, that acts how a human would in a given situation. The human can observe the agent's current and past states, past actions taken, and what action the agent recommends it takes.
- **Temporal:** Interactive Reinforcement Learning agents operate interactively. The human can provide information to the agent before, during, or after learning, and repeatedly throughout the learning process. This allows the human to react to current information and iteratively supply the agent with relevant advice.
- **Interpretation:** The human provides either boolean agreement about past actions taken, recommendations about actions to take, or a reward signal. Computer simulated agents can receive this information as key presses. However, physical agents may receive this information through audio input. Audio input may be simple commands such as Correct or Go Right which can be translated to a form the agent can understand.
- **Structure:** A common structure of advice the agent requires is simply a state-action pair. Using this structure the human can assign advice to a state for the agent to use, such as In this state, do this.
- **External Model:** A simple retained model is more commonly used. This retained model tracks what advice/feedback has been received for each state. The agent can use this model to determine the human's accuracy, consistency, and discount for each piece of advice received. The model acts as a lookup for the agent, if advice exists for the current state, then the agent can use it. Alternative methods may not retain information given by the human and only use it for the current state.
- **Modification:** The most common methods of using the human's advice to modify the agent's learning process are reward and policy shaping. Reward shaping uses assessment/critique gathered from the human to alter the reward given to the human. If the human disagreed with a past action, then the reward received for that state-action pair is decreased. If the human recommends an action to take in the future, then policy

shaping can be used to override the agents usual action selection mechanism. One method of implementing policy shaping for interactive advice is probabilistic policy reuse (Fernández & Veloso, 2006) .

- **Agent:** The Reinforcement Learning agent. The agent operates as a Reinforcement Learning agent would, and should continue to do so even if no advice from the human is given.

3.3.3 Reinforcement Learning from Demonstration

Learning Reinforcement Learning from Demonstration (RLfD) is a term coined by Schaal (Schaal, 1997). It refers to the setting where both a reward signal and demonstrations are available to learn from, combining the best of the fields of Reinforcement Learning and Learning from Demonstration. In the former, an objective evaluation for behaviour is given, and thus in theory optimal behaviour can be achieved. Such an objective evaluation of behaviour is not present in Learning from Demonstration (B. D. Argall, Chernova, et al., 2009), where only expert demonstrations are available to be mimicked and generalised. The student can thus not surpass its master. Yet, Learning from Demonstration is typically much more sample efficient than Reinforcement Learning. Hence the combination, aiming to combine fast Learning from Demonstrations with objective behaviour evaluation and theoretical guarantees from Reinforcement Learning.

In his groundbreaking paper, Schaal (Schaal, 1997) proposed two approaches to using demonstrations in a Reinforcement Learning setting that were later further developed by other researchers. The first is the generation of an initial value-function for temporal difference learning by using the demonstrations as passive learning experiences for the RL agent. This was later expanded upon by Smart and Kaelbling (Smart & Kaelbling, 2002). The second approach Schaal proposed was to derive an initial policy from the demonstrations and to use that to kickstart the RL agent. This approach was also picked up and further developed (Taylor, Suay, & Chernova, 2011; Brys et al., 2015; Suay et al., 2016). Below we describe how the Human-Agent Transfer algorithm, or HAT, as proposed by Taylor et al. (Taylor et al., 2011) fits into the Assisted Reinforcement Learning framework.

- **Information Source:** An expert of the task (human or otherwise) which can pro-

vide sample behaviour by demonstrating its execution of the task. Preferably these demonstrations are efficient and successful executions of the task.

- **Temporal:** Planned: demonstrations are recorded and given to the learning agent before it starts training.
- **Interpretation:** The received demonstrations must be first transformed into the agent’s perspective by encoding them as sequences of state-action pairs. These are then processed using a rule-based classifier, which serves as the Learning from Demonstration component, creating an approximation of the demonstrator’s policy using rules.
- **Structure:** The information is encoded as a rule system that maps states to the actions the demonstrator is hypothesized to execute in those states.
- **External Model:** The generated rules are stored in the external model and not modified anymore. The external model can be queried with a state and responds the hypothesized demonstrator action in that state.
- **Modification:** The action proposed by the demonstrator can be integrated in the agent through three methods: 1) attributing a value bonus to the Q -value for that state-action pair, 2) extending the agent’s action set with an action that executes the hypothesized demonstrator action, and 3) probabilistically choosing to execute the action suggested by the model.
- **Agent:** During its decision making (when and how depends depends on the implemented modification method) the agent has the option to consult the external model to obtain the action that the demonstrator is assumed to take.

3.3.4 Transfer Learning

The idea of transferring information between tasks, rather than learning every tasks from scratch, seems to be obvious in retrospect. While transfer between different tasks has long been studied in humans, it has only gained popularity in RL settings in the last decade or so (Taylor, Stone, & Liu, 2007). We consider three distinct settings where transfer learning (TL) can be useful.

First, an agent may have learned how to perform a task and a new agent must learn to perform that same task. If the two agents have different state features (i.e., different sensors) or different action spaces (i.e., different actuators), an *inter-task mapping* (Taylor, Stone, & Liu, 2007) can be hand specified or (in some cases) learned from data (Taylor, Whiteson, & Stone, 2007; Ammar, Eaton, Ruvolo, & Taylor, 2015) to relate the new *target* agent to the existing *source* agent. One of the simplest ways to reuse such knowledge is to embed it into the target task agent, e.g., directly re-use the Q-values that the source agent had learned (Taylor, Stone, & Liu, 2007) (see Figure 3.8).

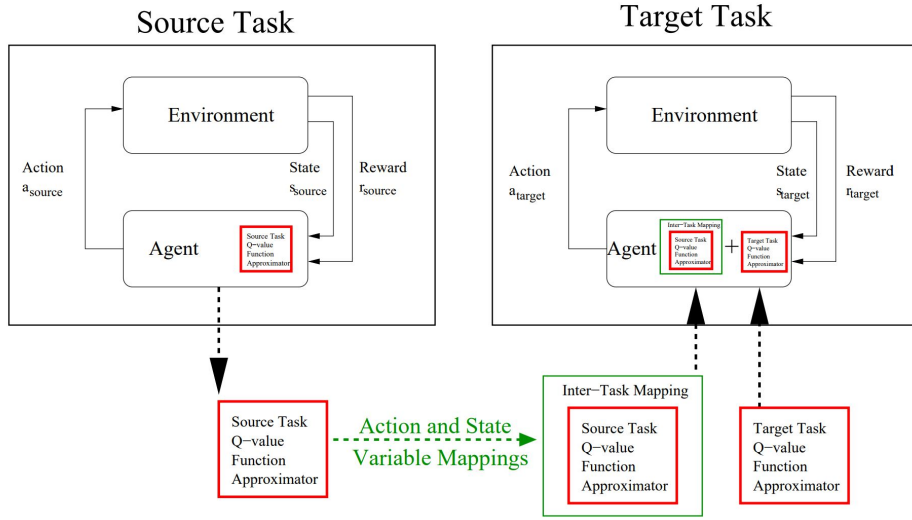


Figure 3.8: One method for transfer is for the target task agent to directly re-use Q-values learned by the source agent, possibly modified by an inter-task mapping.

Second, consider that the world may be non-stationary. In TL settings, it is common to assume that the agent is notified when the world (or task in that world) changes — a TL agent typically does not need to detect changes (Hernandez-Leal, Zhan, Taylor, Sucar, & Munoz de Cote, 2016) or worry about the world changing slowly over time (e.g., as is done in concept drift (Akila & Zayaraz, 2015)). Instead, consider the case where an agent is presented with a new task and it can either reuse its past knowledge or choose to ignore its past knowledge. As in the previous setting, the agent may want to modify the information, e.g., by using an inter-task mapping, to relate the two tasks. In addition, the agent may decide not to use its prior knowledge at all, e.g., to avoid *negative transfer* because the tasks are too dissimilar (Taylor, Stone, & Liu, 2007).

Third, TL could be a critical step within a curriculum learning approach (Taylor, 2009; Bengio, Louradour, Collobert, & Weston, 2009). For example, our previous work showed that learning a sequence of tasks that gradually increase in difficulty can be faster than directly training on the final (difficult) task (Taylor, Stone, & Liu, 2007). In addition to curricula that are created by machine learning experts, we have also considered curricula that are constructed by naive human participants (Peng et al., 2017). Others (Narvekar, Sinapov, & Stone, 2017) consider a complimentary problem, where the learning agent autonomously creates a curriculum. In all cases, the difficulty is scaffolding correctly so that the agent can learn quickly on a sequence of tasks. This approach is distinct from multi-task learning (Fernández & Veloso, 2006), where the agent wants to learn over a distribution of tasks, and lifelong learning (Chen & Liu, 2016), where learning a new task should also improve performance on previous tasks.

- **Information Source:** The information comes from an agent with a different capabilities or the same agent that has trained in a different task.
- **Temporal:** Transfer typically occurs when a task changes or when an agent first faces a novel task. In both cases, the source agent transfer knowledge to the target agent before the target agent begins learning. If the inter-task mapping is initially unknown, some time may be spent trying to learn an inter-task mapping or estimate task similarity to previous tasks. However, the more time spent before transfer, the less impact transfer can have.
- **Interpretation:** There are many types of information that can be transferred, including Q-values, rules, a model, etc. (Taylor, Stone, & Liu, 2007). TL methods assume the target agent has access to the source agent’s “brain,” an assumption that may not always be true (e.g., if the designer of the source agent has not provided an API) or cannot be true if the source agent were a human.
- **Structure:** The structure of the transferred knowledge is as varied as the types of knowledge that can be provided.
- **External Model:** The target task agent typically uses the transferred information to bias its learning. Because the source task knowledge is not necessarily optimal, it is

important for the target task agent to be able to learn to outperform the transferred information.

- **Modification:** The transferred knowledge is not typically modified. Instead, the target task agent builds on top of the knowledge, learning when to ignore it and instead follow the knowledge it has learned from the environment.
- **Agent:** The agent is a typical Reinforcement Learning agent that can take advantage of one or more types of prior knowledge.

3.3.5 Multiple Information Sources

While the majority of work in assisted RL is based on a single source of advice, several researchers have considered scenarios where multiple sources of advice may exist. The introduction of multiple advisors may have benefits for ARL agents, particularly in scenarios where each individual advisor has knowledge which is limited in some way - for example, perhaps individual advisors may have expertise covering different sub-areas of the problem domain. However it also introduces additional problems for the agent, such as handling inconsistencies or direct conflicts between the guidance provided by different advisors, or learning to judge the reliability of each advisor, possibly in a state-sensitive manner. In the extreme case an agent may even need to be able to identify and ignore the advice provided by deliberately malicious advisors (Nunes & Oliveira, 2003).

- **Information Source:** Prior research has identified several scenarios in which an agent may have access to multiple sources of external information. (B. D. Argall, Browning, & Veloso, 2009) argue that when robots are applied to tasks within society in general, it is very likely that multiple users will interact with and guide the behaviour of a robot. In the context of transfer learning, multiple sources of information may be derived either from experience on varying MDPs (Parisotto, Ba, & Salakhutdinov, 2015), or on alternative mappings from a single prior MDP to the current environment (Talvitie & Singh, 2007). In multi-agent systems, each agent may serve as a potential source of information for every other agent (Nowé, Verbeeck, & Peeters, 2006; Mason et al., 2016; Nunes & Oliveira, 2003).

- **Temporal, Interpretation, Structure:** The majority of work so far on assisted RL from multiple information sources has assumed that these sources are homogeneous in terms of the timing and nature of information provided. However this need not be the case, and for heterogeneous information sources the Temporal, Interpretation and Structure aspects of the framework may differ. For example, (B. D. Argall, Browning, & Veloso, 2009) consider two different sources of information, in the form of teacher demonstrations, and teacher feedback on trajectories generated by the learner. The former may be provided in advance of learning, and will consist of complete state-action trajectories, while the latter occurs on an interactive basis during learning, and structurally consists of a subset of the learner’s actions being flagged as correct by the teacher.
- **External Model:** An assisted RL agent must choose whether to maintain a separate model for each information source, to combine the information from all sources into a single model, or a combination of both. An example of the latter approach is the Inverse RL system of (Karlsson, 2014), which learns a model of each information-source in the form of a feature-weighting function, and then forms a combined feature-weighting via averaging. As noted by (Karlsson, 2014) single-model approaches may encounter difficulties if dealing with information sources which are fundamentally incompatible with each other. An additional benefit of maintaining independent models is that these can also be augmented by additional data on characteristics of each information source, such as the reliability or consistency of its advice (Talvitie & Singh, 2007; B. D. Argall, Browning, & Veloso, 2009).
- **Modification:** Any of the modification approaches discussed in the earlier sections of this paper may also be applied in the context of multiple information sources – for example, modification methods from inverse Reinforcement Learning (Karlsson, 2014; Tanwani & Billard, 2013), learning from demonstration (B. D. Argall, Browning, & Veloso, 2009), transfer learning (Talvitie & Singh, 2007; Parisotto et al., 2015), and reward shaping (Brys, Nowé, Kudenko, & Taylor, 2014). The main additional consideration is how these methods may be affected by the presence of multiple external models. The main methods examined so far either use a combination of the models,

either weighted or unweighted (Karlsson, 2014; B. D. Argall, Browning, & Veloso, 2009) or select a single best model to use (Talvitie & Singh, 2007).

- **Agent:** In most circumstances the operation of the agent itself is largely unaffected by the presence of more than one information source. However (Tanwani & Billard, 2013) consider the task of performing inverse RL from demonstrations provided by multiple experts, operating according to different strategies or preferences. To address the potential incompatibilities between these strategies, the agent may attempt to learn a set of multiple policies, so as to be able to satisfy any policy expert strategy, including those not provided to the agent. This approach is closely related to multi-policy algorithms developed for multiobjective Reinforcement Learning (Rojers, Vamplew, Whiteson, & Dazeley, 2013).

3.4 Future Work

This section proposes some possibilities for future work in the field of Assisted Reinforcement Learning. These open questions were identified from the current literature in the field.

3.4.1 Incorrect Assistance

A common assumption that Assisted Reinforcement Learning methods make is that all external information that the agent receives is accurate. Accurate information would be information that assists the agent in completing its goal. The assumption that information will always be of use to the agent is flawed, especially when the information source is an observing human (Efthymiadis, Devlin, & Kudenko, 2013).

Humans may deliver advice late, and the agent will relate it to the wrong state, the advice may be of short term use to the agent but prevent it from achieving optimal performance, or the human may be malicious and actively attempting to sabotage the agent's performance. Incorrect information can be introduced by other sources. Some examples for non-human incorrect advice include behaviour transferred from another domain that does not align correctly, rules that generalise over multiple states may cover exception states and

noisy or missing information from audio-visual sources.

Information given to agents may be correct initially, but over time no longer be the optimal solution (Akila & Zayaraz, 2015) . Other advice may be mostly accurate or correct for most states, however, there can exist states of exception to the advice. This exception states can be the critical difference between an ordinary solution and the optimal solution. Finally, there may be entire information sources that always provide incorrect information, such as malicious humans who actively attempt the agent from learning. There is a need for research into how to identify and mitigate incorrect information in these scenarios.

3.4.2 Multiple Information Sources

Many Assisted Reinforcement Learning methods utilise only one information source. The use of multiple information sources can increase the agent’s knowledge of the environment, and increase confidence in decision making if the different sources agree on an action. However, the use of multiple sources raises additional questions.

- What if the different sources disagree on the best action to take?
- How can the agent identify the best information source to listen to?
- How can the agent manage conflicting information?
- How can the agent measure trust in the different information sources?

Additionally, the use of multiple sources may be extended to crowd sourcing. Crowd sourcing refers to the enlistment and utilisation of a large number of people, either paid or unpaid and can range in size from tens to tens of thousands. Typically, crowd sourcing is performed via the internet. This can raise challenges of malicious users, anonymity, and large uncertainty in the value and reliability of information.

Despite the challenge of implementing multiple users, it remains an open question in the field of Assisted Reinforcement Learning.

3.4.3 Interpretability

So far in this paper, the term interpretability has been about translating the source’s information into a form the agent could understand. In this section, interpretability refers

to translating the agent’s information into a form the human can understand. The reasons why an agent develops certain behaviours can sometimes be difficult to understand. When combining the Reinforcement Learning method with policy modification methods such as rules, expert assistance, external models, and policy shaping, understanding why an agent chooses to take an action becomes even more difficult. Developing methods for understanding agent learning and its decision making is important as it allows the human to remain informed of the agent’s motivations and decisions, and keep track of the accountability of the actions taken. This can be beneficial for artificial intelligence ethics, human-computer teaching, and other fields.

3.4.4 Two-Way Communication

Two-way communication refers to the ability for the information source and the agent to converse with each other, perhaps multiple times before making a decision. Two-way communication can allow the information source, presumably human, and the agent to ask questions of each other, request more information, and to clarify decision making and its reasoning. Most current assisted Reinforcement Learning methods do not have two-way communication to the extent that non-expert human advisors can interact with the agent freely. For two-way communication to apply to non-expert human advisors the issue of interpretability (discussed in Section 3.4.3) will need to be addressed. Other issues that need addressing are timing and agent initiation.

Timing refers to the time it takes to communicate back and forth. Agents sometimes have a fixed time limit, during which they need to learn, communicate, and decide on the next action. Methods for reducing the time it takes to interact with the human and reducing the number of interactions needed with the human are two areas open for research.

Agent initiation refers to the ability for the agent to initiate communication with the human source itself. The agent may choose to do this so to request clarification on information, or request assistance for decision making. A challenge for agent initiation is to determine when and how often the agent should request assistance. The requests for assistance should be high enough to make use of the information source while not becoming a nuisance to the human, or detracting from learning time.

3.5 Conclusion

This article presented ‘Assisted Reinforcement Learning’, a group of Reinforcement Learning methods that utilise external information. Assisted Reinforcement Learning methods use external information to supplement the information the agent receives from the environment to improve performance and decision making.

To describe the different Assisted Reinforcement Learning methods we proposed a framework, with an associated taxonomy, to classify the different functions of an externally-influence Reinforcement Learning agent. Through analysis of current assisted literature, we found seven key features that make up an assisted Reinforcement Learning technique. A definition and examples of each of these seven features have been presented.

Additionally, we demonstrated the applicability of the framework on current Assisted Reinforcement Learning fields. These areas include Interactive Reinforcement Learning, inverse Reinforcement Learning, heuristic Reinforcement Learning, Reinforcement Learning from demonstration, transfer learning, and multi-agent systems. Each of these fields was analysed and described as applied to the presented framework. Finally, we discussed some areas for future research in the field of Assisted Reinforcement Learning.

Chapter 4

Experimental Methodology

This chapter details the environments and experimental methodology used throughout this dissertation. The experiments performed in this research each use one of the following environments. To avoid repetition in the dissertation, and to aid the readers comprehension, the environment definitions have been placed in a separate chapter. Future chapters will refer to the sections in this chapter when necessary.

4.1 Environment Descriptions

There are many environments that are used in the Reinforcement Learning community for testing and comparing methods, and what an agent defines as a state can differ between these environments. This section describes several testing environments, and how an agent may represent the states in each.

4.1.1 State Space Representation

Reinforcement Learning attempts to find the optimal action to take for each state in an environment. Mapping states to actions are simple for discrete problems such as board games (Tesauro, 1994) and scheduling (W. Zhang & Dietterich, 1995), where state observations are readily distinguishable. However, most real-world environments have continuous state spaces, such as 3-D navigation (Ng et al., 2006), car driving (Lillicrap et al., 2015; Abbeel & Ng, 2004b; Michels, Saxena, & Ng, 2005), or tasks requiring fine motor control such as pancake-flipping (Kormushev, Calinon, & Caldwell, 2010). In continuous state spaces, the difference between one state and another can be minuscule, but the agent may see it as a completely different state with no relationship to the prior¹, possibly creating an infinite state space.

¹For example, the agent will see the numbers 42.0001 and 42.0002 as two very different numbers.

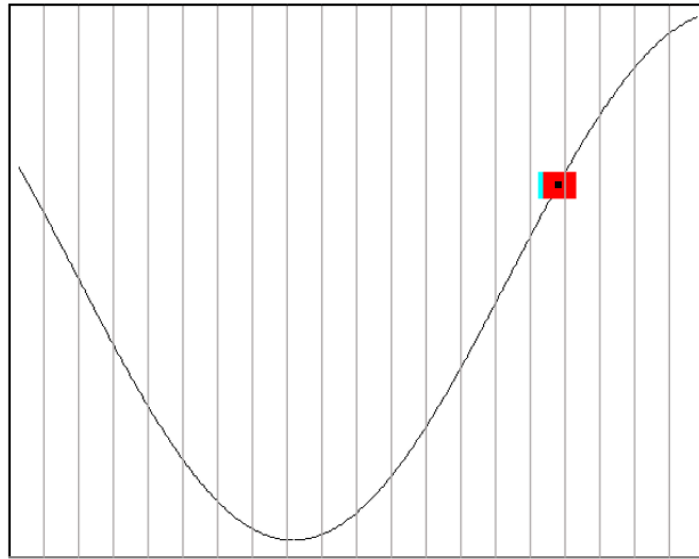


Figure 4.1: A graphical representation of the Mountain Car problem (Tanner & White, 2009). The position of the car (shown in red) is a continuous value ranging from -1.2 to $+0.6$. The position is discretized into 20 ‘bins’, each 0.09 in length.

In applications of Reinforcement Learning, efforts have been made to reduce some of the complexities of continuous state spaces. These include function approximation (Sutton et al., 2000; Sutton, 1996), adaptive state partitioning (Moore, 1994), aggregation methods (Pareigis, 1998; Singh & Bertsekas, 1997), and discretization (Doya, 2000). A common approach is discretization, breaking up the state space into discrete ranges and treating all states that fall into each range as a single state. The individual states that encapsulate a section of the continuous space may be referred to as a bin. An example of a discretized continuous state variable is shown in Figure 4.1.

Discretization may be the most common approach to handle continuous state spaces, and the simplest to implement, however, it does have some drawbacks (Doya, 2000).

- If the discretization is too coarse, the action the agent learns for the state may not be the optimal action for every environment position encapsulated in the state.
- If the discretization is too fine, the number of states increases and learning becomes slower.
- To generate an ideal discretized state space, intimate knowledge of the environment's dynamics may be needed.

For this project, discretization is used for continuous state spaces in all experiments. Further explanation of how the agent represents the state space is given for each environment.

4.1.2 Mountain Car

The Mountain Car environment is a standard continuous-state testing domain for Reinforcement Learning (Sutton & Barto, 1998; Moore, Birnbaum, & Collins, 1991). In the environment, an under-powered car must drive from the bottom of a valley to the top of a steep hill. Since the gravity in the environment is stronger than the cars engine, the car cannot drive straight up the side of the mountain. In order for the car to reach the top of the mountain the car must build up enough inertia and velocity. Figure 4.2 illustrates the mountain car environment.

A Reinforcement Learning agent controls the actions of the car. Figure 4.2 shows the key features of the environment. The agent will begin at a random position and with a low velocity somewhere within the starting position. In order to reach the goal the agent must

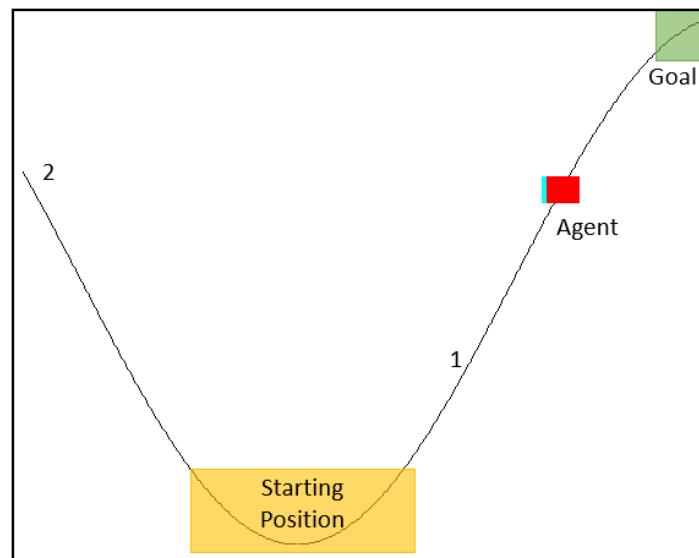


Figure 4.2: A detailed graphical representation of the Mountain Car environment. The agent begins on the line at a random position within the yellow box and must travel to the green goal state. To do so, the agent accelerates towards the first (1) key position until its velocity is reduced to zero by gravity. At this point the agent turns and accelerates towards the second (2) key position, again, until its velocity is reduced to zero. Finally, the agent accelerates down the hill again, building up velocity to reach the goal state.

build up enough momentum. To do so, the agent accelerates towards the first (1) key position until its velocity is reduced to zero by gravity. At this point the agent turns and accelerates towards the second (2) key position, again, until its velocity is reduced to zero. Finally, the agent accelerates down the hill again, building up velocity to reach the goal state. Should the agent not reach high enough up the mountain to reach the goal, it should repeat the actions of accelerating in the opposite direction until a zero velocity is reached and turning around.

The key to the agent solving the mountain car problem is to increase its own velocity (v). The agent's mass, magnitude of acceleration (a), and the force of gravity (G) are constant. As the agent's acceleration is lower than the gravity acting upon it, pulling the agent to the lowest point of the environment, the agent must accelerate at the correct moments, and in the correct direction, to increase its velocity. The optimal solution to the mountain car problem is to accelerate in the current direction of travel and take a random action when velocity is zero. The formulate denoting this behaviour is shown in Example 4.3.

$$A_t = \begin{cases} +1, & v > 0 \\ -1, & v < 0 \\ \in \{-1, 1\}, & v = 0 \end{cases}$$

Figure 4.3: An example of a rule stating the agent should accelerate left if moving left, accelerate right if move right, and take a random action if velocity is 0.

Agent's State Representation

The Mountain Car is a continuous, two-dimensional state space: position and velocity (Sutton, 1996; Singh & Sutton, 1996). The agent controlling the car has three actions to choose from in any state, to accelerate left, accelerate right, or not to accelerate at all, the graphical representation of which is shown in Figure 4.4. At each step, the agent receives a reward of -1, and no reward for reaching the goal state. This encourages the agent to reach the goal in as few steps as possible to reduce the reward lost.

The two state variables, position and velocity, are represented as decimal numbers. The position variable represents the agent's position within the environment, and ranges linearly from -1.2 to 0.6, with the lowest point of the environment residing at -0.53. The velocity of the agent has a range of -0.07 and 0.07. A velocity greater than zero indicates the agent is

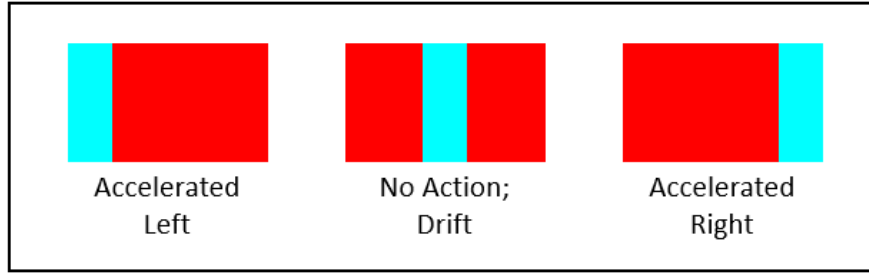


Figure 4.4: A graphical representation of the Mountain Car agent. The entire rectangle (blue and red) is the car. The blue box indicates which action the agent has chosen to perform, either to accelerate left, right, or not to accelerate at all and continue moving in it is current direction of travel.

travelling to the right, or increasing its position. If the agent collides with the edge of the environment on the left ($p=-1.2$) then the agent's velocity is set to 0.

To represent the continuous state space the Reinforcement Learning agent used in this research discretizes the two state variables. For the purposes of this research, unless otherwise noted, 20 bins for each state variable has been used, creating a total of 400 (20×20) states. Of these 400 states, there are many that may never be visited by an agent, for example, it is impossible that the agent will be at top of the left mountain ($p = -1.2$) and have a high positive velocity ($v = 0.07$).

4.1.3 Self-Driving Car

The self-driving car (SDC) environment is a control problem in which a car, controlled by the agent, must navigate an environment while avoiding collisions and maximising speed. The car has collision sensors positioned around it which can detect if an obstacle is in that position, but not the distance to that position. Additionally, the car can observe its current velocity. Using just the observations from these sensors the agent attempts to learn to drive as fast as possible while not crashing. The final behaviour learnt depends on the layout of the environment, which the agent cannot observe.

All observations made by the agent (car) come from its reference point, this includes the obstacles (e.g., there is an obstacle on my left) and cars current speed. This implementation is conceptually similar to a blind person with a cane, tapping nearby surroundings to determine obstructions. The agent cannot observe its position in the environment. For example, the

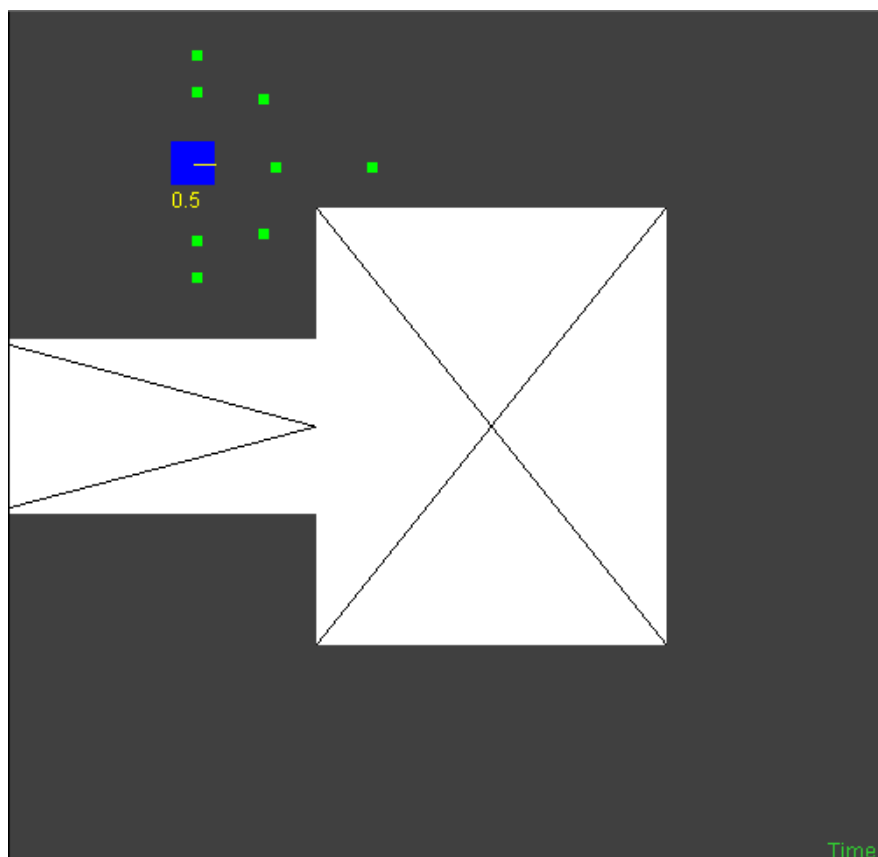


Figure 4.5: A graphical representation of the Simulated Car agent. The small blue square in the top left is the car. The yellow line within the car/square indicates the current direction and the number below is the current velocity. The smaller green squares surrounding the car/square are collision sensors and will always align with the cars current direction. The large white rectangles are obstacles. If the blue box representing the car collides with a wall or the obstacles, the episode terminates.

agent cannot determine if it is in the top-right section of the map, as it has no reference to its current position. Additionally, the agent does not attempt modelling the environment to build a belief in the layout of the map.

Each step, the environment provides the agent reward equal to its current velocity. A penalty of -100 is awarded each time that the agent collides with an obstacle. Along with the reward, the agent's position resets to a safe position within the map, velocity resets to the lower limit, and the direction of travel is set to face the direction with the longest distance to an obstacle. These values are chosen to give the agent the safest possible start to its learning, conditions reasonably assumable to be chosen in the case of a real self-driving car.

Figure 4.5 shows the map used for the self-driving car experiments performed in this

body of research. This map challenges the agent to learn a behaviour that maximises velocity while avoiding collisions by using a layout that prohibits turning at high speeds at the narrow corridors on the top, right, and bottom of the map. The only two sections of the map that allow for high-velocity turning are the large empty sections on the left side. It is important to remember that the agent cannot see the grand layout of the map, only whether there are possible obstacles nearby. As with any Reinforcement Learning agent, the aim is not to learn what the environment looks like, but how to best respond to its current observations and how to act to improve its future situation.

Figure 4.5 also provides a representation of the agent and the positioning of the collision sensors around it. These collision sensors return a boolean response as to whether there is an obstacle at that position though not the distance to that obstacle. Additionally, the agent does not know the position of its sensors in reference to itself. The only information the agent has regarding the sensors is whether each is currently colliding with an obstacle. The agent also knows its current velocity. The possible velocity of the agent is capped at 1m/s at the lower end, and 5m/s at the higher end. Having the lower cap above a zero velocity prevents the agent from moving in reverse or standing still. This lower limit reduces the state space and prevents an unintended solution, that standing still is an excellent method for avoiding collisions. The upper limit of 5m/s is set so that velocity is not limitless and further reduces the state space, while still being high enough that it exceeds the limit for a safe turn anywhere in the environment. An action that attempts to exceed the velocity thresholds set by the environment will return the respective limit.

There are five possible actions for the agent to take within the self-driving car environment. The five actions are:

- (i) Accelerate. The agent will increase its velocity by 0.5 meters per second. If the agent chooses to accelerate when it is already travelling at the top speed of 5.0 m/s, then no change will be made to the agent velocity. However, equivalent to the ‘Do Nothing’ action, this will still register as an action taken and the agent’s position is updated.
- (ii) Decelerate. The opposite of accelerating. The agent’s velocity will decrease by 0.5 meters per second. If the agent chooses to decelerate below the lower velocity limit of 1.0m/s, then no change is made to the agent’s velocity, but the agent’s position will be

updated, equivalent to the ‘Do Nothing’ action.

- (iii) Turn Left. The agent will alter its direction of travel by 5 degrees to the left. The current velocity does not affect how much the agent can turn by, only how much distance is travelled while altering its direction. After choosing a turning action and the facing direction has been changed the agent’s facing direction is not altered again unless another turning action is taken. The only time when the agent’s direction of travel is changed is in response to a turning action being performed.
- (iv) Turn Right. This action operates with the same dynamics and constraints as turning left, but the agent will turn right instead.
- (v) Do Nothing. The agent’s velocity or direction of travel is not altered. When performing this action the only change is the agent’s position, based on current velocity, position, and direction of travel. Actions that attempt to accelerate or decelerate the agent beyond the velocity bounds of the environment will perform equivalent to this action.

Figure 4.6 shows the process the environment follows to perform the action chosen by the

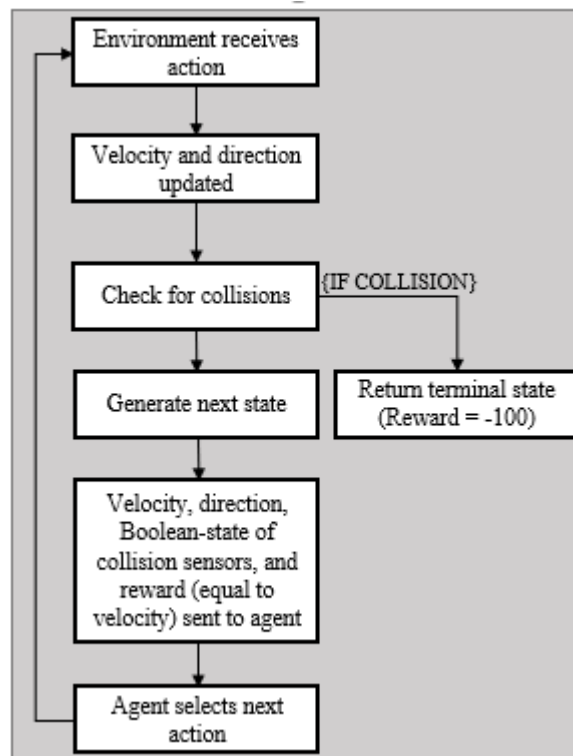


Figure 4.6: The process flow of the simulated car environment.

agent and return the new observation. The process executes in the following order.

- (i) The environment receives the action selected by the agent.
- (ii) The agent's velocity or direction of travel updates according to the action selected. If the action requests that the agent's velocity exceed the bounds set by the environment, then the velocity is set to the corresponding bound.
- (iii) The agent's position is updated based on the current velocity and direction of travel. If the agent collides with an obstacle, then a terminal state is returned. The terminal state has no observation and a substantial penalty reward. The process ends here if a terminal state is returned.
- (iv) The state information is collected from the environment. In addition to the agent's current velocity, each of the agent's collision sensors is checked, and the result of each is added to the state information.
- (v) The environment sends the current state to the agent. The agent uses this state information to make a decision on the action to perform and the process repeats.

Agent's State Representation

The Self-Driving Car environment has eight state features, one for each of the collision sensors on the car, and the current velocity of the car. The collision sensor state features are boolean, representing whether they detect an obstacle at their position. The velocity of the agent has nine possible values, the upper and lower limits, plus every increment of 0.5 value in between. With the inclusion of the five possible actions, this environment has 5760 state-action pairs.

The reward function defined by the environment promotes the agent to learn a behaviour that avoids obstacles while attempting to achieve the highest velocity the environment allows. The most natural solution to learn that achieves these conditions is to drive in a circle, assuming that the path of the circle does not intersect with an obstacle. The map chosen for use in these experiments allows an unobstructed circle path to be found, but only at low velocities. If the agent is to meet both conditions that achieve the highest reward, a more complex behaviour must be learnt, see Figure 4.7.

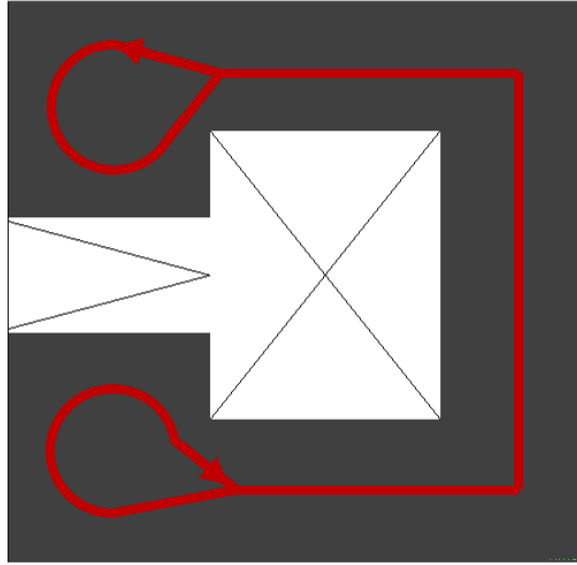


Figure 4.7: Optimal path for Simulated Car environment. Turning at the wide corners allows the agent to maintain a higher velocity.

4.1.4 Mario

The Mario environment is a benchmark problem based on Infinite Mario Bros (Goschin, Weinstein, Littman, & Chastain, 2013), which is an implementation of the 80's NES game Super Mario Bros ®. In this problem, the agent controls Mario and is attempting to achieve the highest game score possible. Mario will gain points for killing enemies, finishing each level, and collecting items such as coins. Mario can avoid losing points by not dying and by completing each map/level quickly. Each second Mario suffers a penalty to the game score, to encourage faster completion. As points can be won or lost by many aspects of the game, the optimal behaviour is challenging to determine ahead of time for both the agent or an observer.

There are two tactics for gaining a high score that an observer may select. The first is to finish each level as fast as possible. Mario is awarded a large number of points for completing each map, and by completing it quickly, the time penalty can be lowered. The second tactic is to kill every enemy and collect every coin before finishing the level. This tactic assumes that points can be achieved at a faster rate than the time penalty will remove. The correct tactic to take will often depend on the number of coins and enemies on each map, as well as the difficulty of collecting them and getting to the end of the level, all of which cannot be

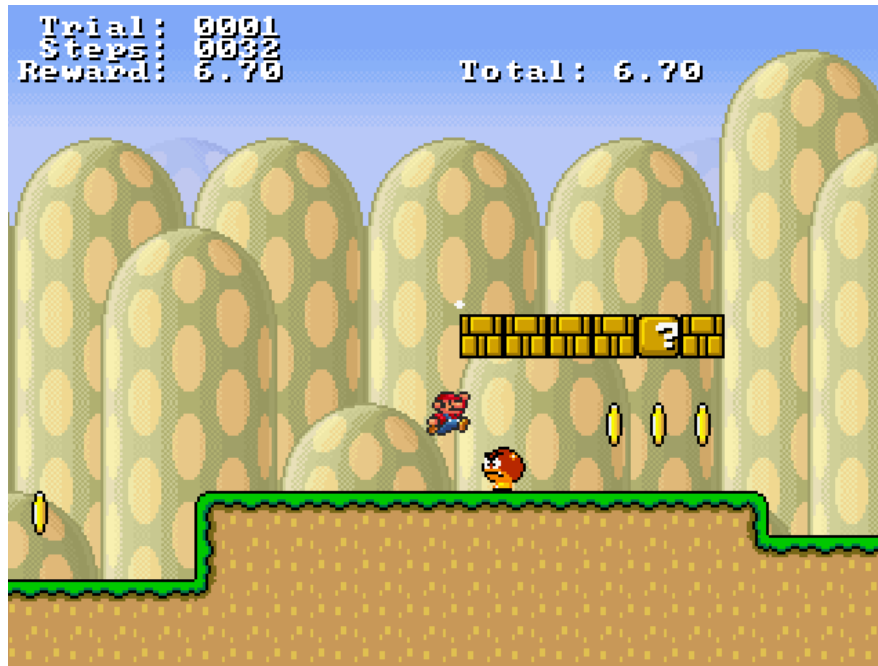


Figure 4.8: RL-Glue Infinite Mario environment.

determined ahead of time. The tactic that Mario, controlled by RL, will choose can largely depend on any additional reward functions, advice, or the discount factor of the agent.

The state features for the Mario environment depend on the implementation. The Mario experiments in this thesis use three different implementations over the course of the research, Littman² (Goschin et al., 2013), Brys (Brys, 2016; Harutyunyan, Brys, Vrancx, & Nowé, 2015) (with a small addition), and a combination of the two.

The Littman implementation of Infinite Mario has no less than 359 state features, at least three integer values, at least three double values, and a set of 352 character values. The number of state features increases as the number of observable entities increases. An observable entity is any entity that is visible on the screen, and is limited to:

- Mario
- Enemies Red Koopa, Green Koopa, Goomba, Spiky, Pipe Flower
- Items Mushroom, Fire Flower, Fireball, Shell

²There are multiple authors for the paper introducing the Super Mario environment, Sergiu Goschin being the first author. However, the RL implementation used acknowledges Michael Littman as the author of the original RL environment.

The first integer value is the distance in the number of blocks between Mario and the left-most side of the map. The second and third integers represent the entity type and whether that entity can fly. These two values will be present in the state features for each state for each observable entity. As Mario is always observable, there will always be at least three integer values. If there are an additional two observable entities, then the number of integer state features would grow to seven, while they each remain observable.

The same instance of expanding state features applies to the features represented as double values. For each observable entity, the following four state features, concerning the representing entity, are added to the state:

- Current X coordinate
- Current Y coordinate
- Current X velocity
- Current Y velocity

Again, as Mario is always observable, there will always be at least four double values. For each additional entities, a further four double values are added to the state feature list for the current state.

Finally, there are 352 state features represented as character values. This set of 352 does not expand like the integer or double state features. The character set represents the observable blocks present on the screen. These blocks include coins, bricks, the ground, and Mario itself. The observable screen is 21 blocks by 16 blocks, These 336 blocks together with a delimiter character denoting the end of each row, represents the screen that is currently observable. Figure 4.1 shows a detailed description of each of the state features for the Littman implementation.

The Brys implementation used in the experiments in this body of research has been slightly changed compared to the original(Brys, 2016; Harutyunyan, Brys, et al., 2015). The original implementation has 27 state features. These state features mostly ignore the presence of the coins and other collectable items in the map. Instead, the Brys implementation provides only crucial information needed to solve the level as quickly as possible. Four addition state features are added to the Brys state space used in this research. These additional features

MARIO STATE DEFINITION (LITTMAN IMPLEMENTATION)		
# OF FEATURES	TYPE	DESCRIPTION
1	INTEGER	DISTANCE IN TILES THAT MARIO HAS MOVED RIGHT, FROM LEFT-MOST SIDE OF ENVIRONMENT.
2 * OBSERVABLE ENTITIES	INTEGER	FOR EACH ENTITY VISIBLE ON THE SCREEN, TWO FEATURES ARE ADDED. THE FIRST REPRESENTS THE ENTITY TYPE, THE SECOND REPRESENTS WHETHER THE ENTITY CAN FLY.
4 * OBSERVABLE ENTITIES	DOUBLE	FOR EACH ENTITY VISIBLE ON THE SCREEN, FOUR FEATURES ARE ADDED. THE FIRST AND SECOND ARE THE X AND Y COORDINATES OF THE ENTITY. THE THIRD AND FOURTH ARE THE CURRENT VELOCITY, X AND Y AXIS, OF THE ENTITY.
352	CHARACTER	EACH CHARACTER REPRESENTS A TILE CURRENTLY VISIBLE ON THE SCREEN. 16 TILES ACROSS AND 21 TILES DOWN, PLUS A NEW LINE CHARACTER AFTER EACH 16 CHARACTERS.

Table 4.1: Feature space of Infinite Mario. Littman Representation.

provide the agent with information about the nearest collectable items on the map. As the original list of state features introduced by Brys has been altered, the remainder of this thesis refers to the altered implementation as Brys+, to indicate the addition of the extra information. With the addition of the extra information, Brys+ has 31 distinct state features.

Rather than provide a complete list of all 336 blocks observable on the screen, Brys+ gives a less detailed view to the agent. Instead, the presence of enemies on the map is represented as a boolean value for each of the cardinal and inter-cardinal directions around Mario, both near and far. Figure 4.9 indicates the layout of the eight directions, and the distance from Mario in which each generalised zone is positioned.

In addition to these sixteen state-features that point out the rough location of enemies, an two additional state features identify the distance to the nearest enemy by the number of blocks away from Mario on the X and Y planes. The four state features that differ between Brys and Brys+ operate in the same way, indicating the distance to the nearest coin, and the nearest item, on both the X and Y planes. The items identified by the last state feature are limited to items that provide a positive reward to Mario, namely breakable blocks, 1-Up mushrooms, and the fire flower. Figure 4.2 gives a breakdown of each of the state features for the Brys+ state feature implementation of the Infinite Mario environment.

There is a third implementation of the state features used in the experiments in this thesis.

MARIO STATE DEFINITION (Brys IMPLEMENTATION)		
FEATURE NUMBER	TYPE	DESCRIPTION
1	BOOLEAN	IS MARIO ABLE TO JUMP?
2	BOOLEAN	IS MARIO ON THE GROUND?
3	BOOLEAN	IS MARIO ABLE TO SHOOT FIREBALLS
4-5	INTEGER	MARIO'S DIRECTION IN THE HORIZONTAL AND VERTICAL PLANES -1, 0, 1
6-9	BOOLEAN	IS THERE AN OBSTACLE IN ONE OF THE FOUR VERTICAL GRIDS CELLS IN FRONT OF MARIO?
10-17	BOOLEAN	IS THERE AN ENEMY WITHIN ONE GRID CELL REMOVED FROM MARIO IN ONE OF THE EIGHT DIFFERENT DIRECTIONS (SEE FIG- URE 4.9
18-25	BOOLEAN	IS THERE AN ENEMY WITHIN TWO TO THREE GRID CELLS RE- MOVED FROM MARIO IN ONE OF THE EIGHT DIFFERENT DIREC- TIONS (SEE FIGURE 4.9
26-27	INTEGER	THE RELATIVE HORIZONTAL AND VERTICAL POSITIONS OF THE CLOSEST ENEMY (-10, 10), PLUS ONE VALUE INDICATING AB- SENCE OF ENEMIES
ADDITIONS TO Brys IMPLEMENTATION (Brys+)		
FEATURE NUMBER	TYPE	DESCRIPTION
28-29	INTEGER	THE RELATIVE HORIZONTAL AND VERTICAL POSITIONS OF THE CLOSEST MYSTERY BLOCK (-10, 10), PLUS ONE VALUE INDICAT- ING ABSENCE OF MYSTERY BLOCKS.
30-31	INTEGER	THE RELATIVE HORIZONTAL AND VERTICAL POSITIONS OF THE CLOSEST ITEM (-10, 10), PLUS ONE VALUE INDICATING ABSENCE OF ITEMS.

Table 4.2: Feature space of Infinite Mario. Brys+ Representation.

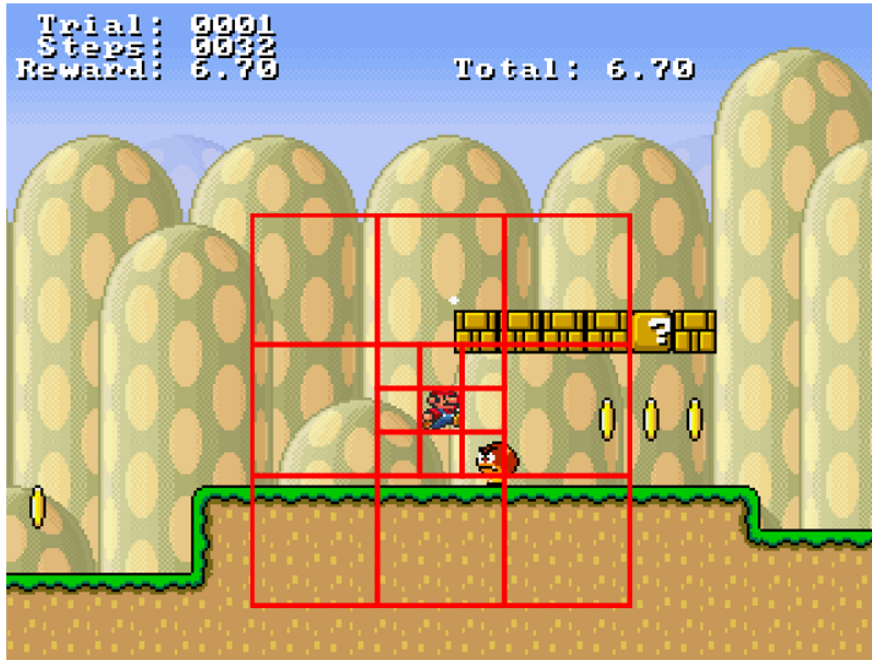


Figure 4.9: Infinite Mario generalisation grid overlay example for Brys+ representation

The implementation is a combination of the both Brys+ and Littman state feature definitions, where the agent uses Littman, and an advice-giver uses Brys+. Further information about this third implementation is provided in Chapter 8.

The reward function is the same for all implementations presented above. A reward is given for each item collected, block broken, level completed, and enemy killed. Penalties are given for each death of the agent, and each second that the agent is alive. Table 4.3 lists each reward scenario and the reward given for the Infinite Mario environment.

REWARD CONDITIONS FOR INFINITE MARIO	
REWARD	CONDITION
1.00	COLLECT A COIN
1.00	KILL AN ENEMY
100.00	REACH LEVEL GOAL
-10.00	DEATH
-0.01	STEP PENALTY

Table 4.3: Reward conditions for Infinite Mario

Despite the name of the environment, each map in Infinite Mario is a total of 320 blocks in length, with 21 block distance laterally visible at any point in time (Figure 4.1.4). The map of the environment is randomly generated each time a new experiment or episode is started. Each map has a flag at the 320th position. Mario cannot move beyond this flag. If Mario reaches this position, a reward is given from completing the level, and a new map is generated. If Mario is killed, then a new map is generated, and Mario must begin again from the start. The starting position is located at the first block of the map, and Mario cannot move further away from the flag at this point. Each map has enemies, coins, breakable and solid blocks, pits, and walls randomly distributed throughout. By building and populating each level randomly, Mario is forced to learn a complex behaviour to achieve competence in playing the game.

The Mario agent has twelve actions available at all times. Each action is a combination of three sub-actions, the direction the agent chooses to travel (left, right, none), whether to jump (yes, no), and whether to fire (yes, no). The combination of these three sub-actions constitutes the twelve actions ($3 * 2 * 2 = 12$) the agent may take at each time step. Figure 4.4 provides a complete list of actions for the Infinite Mario environment.

Some sub-actions benefit from being repeatedly chosen without a break. For example, the duration in which the jump sub-action is taken will increase the agent’s jump height up to a maximum height, at which point the jump sub-action needs to be not selected for at least one step before a second jump to be started.

The fire sub-action operates on the same dynamic. However, the result of choosing the fire sub-action depends on whether a fire flower has been collected by the agent previously in the current episode. If the agent has collected a flower, when the agent chooses to fire then a fireball will be thrown. Similar to the jump action, a second fireball cannot be thrown until the agent has not selected that sub-action for at least one step. If a fire flower has not been collected, then the choice of this sub-action decides whether Mario should run. For as long as the agent chooses the fire sub-action it will gain a speed boost. When the sub-action is no longer selected, then Mario slows to its usual pace. Figure 4.10 shows the process the Infinite Mario environment follows to perform the action chosen by the agent and return the new observation. The process executes in the following order:

ACTION SPACE FOR INFINITE MARIO			
#	DIRECTION	DO JUMP?	DO FIRE?
1	MOVE LEFT	JUMP	FIRE
2	MOVE LEFT	JUMP	DON'T FIRE
3	MOVE LEFT	DON'T JUMP	FIRE
4	MOVE LEFT	DON'T JUMP	DON'T FIRE
5	MOVE RIGHT	JUMP	FIRE
6	MOVE RIGHT	JUMP	DON'T FIRE
7	MOVE RIGHT	DON'T JUMP	FIRE
8	MOVE RIGHT	DON'T JUMP	DON'T FIRE
9	DON'T MOVE	JUMP	FIRE
10	DON'T MOVE	JUMP	DON'T FIRE
11	DON'T MOVE	DON'T JUMP	FIRE
12	DON'T MOVE	DON'T JUMP	DON'T FIRE

Table 4.4: Action space for Infinite Mario

- (i) The environment receives the action selected by the agent. The action is made up of three sub-actions, the direction to move, whether to jump, and whether to fire.
- (ii) The agent's velocity and direction of travel updates according to the direction the agent choose to move, and whether the jump or fire sub-actions were chosen. The fire sub-action defaults to a speed boost if the agent has not found a fire flower previously on the current map.
- (iii) The agent's position is updated based on the current velocity and direction of travel. If the agent collides with an enemy at an intersection that does not result in the enemy's death, then Mario is killed, and the environment returns a terminal state. The environment also returns a terminal state if Mario falls into a pit, or reaches the flag at the end of the current map. The process ends here if a terminal state is returned.
- (iv) The current state information is generated, ready to be sent to the agent. The imple-

mentation used in an experiment defines what state information is observable by the agent. The implementations that may be used include Littman, Brys+, or both. Tables 4.1 and 4.2 describe the difference between the two.

- (v) The environment sends the current state to the agent. The agent uses this state information to make a decision on the action to perform and the process repeats.

Agent's State Representation

This section describes how the agent represents the state information provided. The state features the agent can observe depends on whether the Littman or Brys+ implementation is being used (See Tables 4.1 and 4.2).

None of the state features in the Brys+ implementation are continuous values, and the values for all features lie within a predefined range, resulting in a set number of states that may exist making a tabular approach to state representation manageable. The Littman implementation has many continuous state features, several without predefined boundaries. Function approximation is well suited for this type of problem. However, to ensure accurate

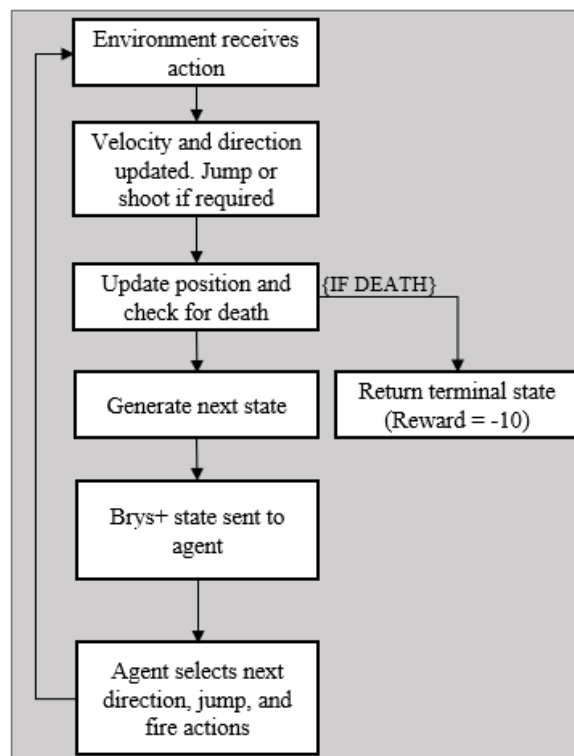


Figure 4.10: The process flow of the Infinite Mario environment.

comparison against the Brys+ implementation, and other experiments using this methodology presented in this thesis, a tabular approach is used. Each continuous state feature is discretized using a step size of 0.5.

4.2 Experiment Toolkit

4.2.1 Community Sourced Toolkit

RL-Glue

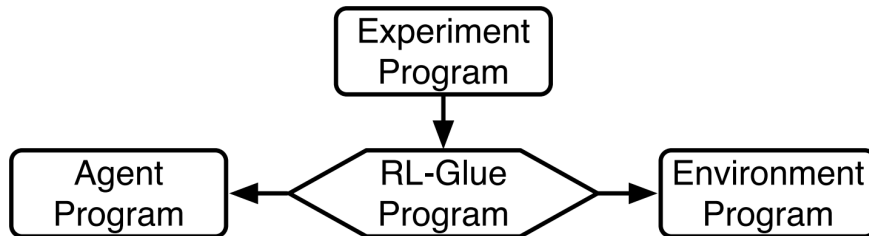


Figure 4.11: The four programs specified by the RL-Glue Protocol. Arrows indicate the direction of the flow of control. Figure sourced from Tanner and White (2009)(Tanner & White, 2009).

RL-Glue is a software framework that facilitates the communication and data collection between Reinforcement Learning agents and environments (Tanner & White, 2009). The RL-Glue interface allows agents, environments, and experiments, written in various programming languages to connect using a server-client networking relationship (Figure 4.11). The interface acts as a harness between the different RL components, reducing the need for having to continually rewrite the connecting code, and reducing the potential for software bugs to occur. The purpose of the RL-Glue project is to aid researchers/developers to adhere to shared community guidelines designed to make sharing and comparing agents and environments simpler. This approach to designing Reinforcement Learning programs allows for several benefits:

- Allows agents, environments, and experiments to be written in any networking-capable programming language.
- Allows easier sharing of code with other members of the Reinforcement Learning community.

- Allows knowledge to be isolated from the environment and the agent, reducing the potential for the agent to have access to information it should not have, such as the reward function.
- Can be extended to handle custom data collection, incorporate multiple agents, and be adapted to work with existing applications and robots.
- Existing environments and agents are available from the Reinforcement Learning community.

The RL-Glue project is available at <http://library.rl-community.org>.

RL-Viz

Accompanying RL-Glue is RL-Viz, a visualisation package that extends RL-Glue to provide a graphical user interface. RL-Viz wraps around the agents and environments of the RL-Glue framework, providing a real-time view of the environment and the agents within. RL-Viz implementations are provided by the Reinforcement Learning community for many of the environments listed in Section 4.1. While the use of RL-Viz, and the graphical user interfaces that it offers, make no difference to the functionality and performance of a Reinforcement Learning agent, they do serve to improve the experience for any human users watching or interacting. The graphical display of the agent and the environment allow the human to observe, in real-time, how the agent is performing, what actions it is choosing, and can serve as an interface for the human to interact with the agent. The RL-Viz project is available at <http://library.rl-community.org>.

Recent Projects

RL-Glue remains a prominent framework for Reinforcement Learning and is continuing to be utilised in recently published research (Gillespie, Gonzalez, & Schrum, 2017; Barreto, Precup, & Pineau, 2016; Abdallah & Kaisers, 2016). Since the introduction of the RL-Glue framework, some major developments have occurred in the Reinforcement Learning community, the biggest being deep learning (Mnih et al., 2015, 2013; Lillicrap et al., 2015). The RL-Glue framework has continued to be relevant in these areas, and researchers have

plans for further extensions to the framework (Vamplew, Dazeley, Berry, Issabekov, & Dekker, 2011).

The RL-Glue framework has also been shown to be compatible with Interactive Reinforcement Learning (Knox & Stone, 2010). A critical feature that Interactive Reinforcement Learning requires is the ability for human users to observe the agent’s state and be able to interact with the agent in real-time. The addition of RL-Viz to RL-Glue makes it a useful framework for Interactive Reinforcement Learning research.

Other Software Frameworks

There are frameworks for Reinforcement Learning research other than RL-GLue (Tanner & White, 2009). Two other frameworks worth noting are BURLAP (MacGlashan, 2015) and OpenAI Gym (Brockman et al., 2016).

BURLAP (Brown-UMBC Reinforcement Learning And Planning) is a Java-based software project. The project includes a wide range of Reinforcement Learning agents, environments, and supports multi-agents. It also provides visualisation tools to facilitate real-time observation of the agent and environment.

OpenAI Gym is a recently released toolkit for developing and comparing Reinforcement Learning algorithms. The project focuses on providing a wide range of environments for its users to build agents for and compete with other users. The environments offered in the project include simple toy problems, AAA title computer games, and not yet solvable control problems. OpenAI Gym is attracting researchers from Deep Reinforcement Learning, as the environments and problems often require image processing to solve.

For the experiments performed in this research, the RL-GLue framework has been chosen and used. RL-Glue has been selected due to the available documentation, community, and multi-platform/multi-language server-client functionality. BURLAP was not chosen despite its more comprehensive range of features because of the lack of community, publications, and multi-language functionality. OpenAI Gym was not chosen because of its focus on Deep Reinforcement Learning rather than Reinforcement Learning, and due to its release being well into the duration of this research project.

Chapter 5

Human-Sourced Advice

The focus of this dissertation is on the effect of rule-based human-sourced advice on the Interactive Reinforcement Learning process. Specifically, aiming to improve the learning speed of the agent while reducing the engagement with the human, compared to existing methods. This chapter aims to address some fundamental questions regarding the dissertation focus.

- (i) What is human-sourced advice?
- (ii) What teaching styles are used to deliver advice in Interactive Reinforcement Learning?
- (iii) How do these teaching styles compare against each other?

This chapter has two sections, the first of which addresses what human-sourced advice is and identify styles in which humans deliver advice. The second section details the methodology and results of a human-trial designed to compare the methods humans use to deliver advice.

5.1 Human-Sourced Advice

Learning from scratch can be a challenging task. While humans and Reinforcement Learning agents are both capable of learning tasks from scratch, it is evident that any extra information regarding the task can significantly reduce the learning time (Taylor & Stone, 2009; Sharma et al., 2007; Taylor, Stone, & Liu, 2007).

For humans, we can get advice from peers, teachers, the Internet, books, and videos. Without a doubt, these sources help us to learn skills faster. By incorporating advice, humans can learn what the correct behaviour looks like, build upon existing knowledge, evaluate

current behaviour, and ultimately reduce the amount of time spent performing the wrong actions.

For Reinforcement Learning agents, the benefits of advice are the same. Advice is used to construct or supplement the reward function, resulting in an improved evaluation of the agent's actions or increased the utility of the reward function requiring fewer experiences to learn a behaviour(Knox & Stone, 2010; Griffith et al., 2013). Advice can also be used to influence the agent's policy, either directly or through the action selection method, improving exploration and reducing the search space.

There are many possible information sources for agents to use. External information can come from databases, labelled sets(Goodfellow, Bengio, Courville, & Bengio, 2016; LeCun, Bengio, & Hinton, 2015), cases(B. Kang et al., 1995; Compton et al., 1991), past experiences(Taylor & Stone, 2009), other agents(Littman, 1994; Tan, 1993), and from humans(B. Argall et al., 2007). Human-supplied advice is contextually relevant information that comes from a human as a result of observation or awareness of the agent's current behaviour or goal. Human-sourced advice is information that is the result of an observing human giving information that may assist the agent in the current context. This information is commonly used to supplement, construct, or alter the Reinforcement Learning process. Human-sourced advice can be more noisy, inaccurate, and inconsistent than other information sources. However, the critical benefit is that the advice is contextually relevant and can be applied to aid the agent in its current situation or goal.

The use of advice in Reinforcement Learning is not new. Reinforcement Learning fields such as Transfer Learning(Taylor & Stone, 2009), Learning from Demonstration(B. Argall et al., 2007), and Interactive Reinforcement Learning(Thomaz et al., 2005) have been utilising human-sourced information for many years.

5.1.1 Transfer Learning

Transfer Learning attempts to accelerate the learning of an agent by reusing information sourced from an agent with different capabilities or the same agent trained on a different problem. The goal of the Transfer Learning field is to design agents capable of reusing internal knowledge in new domains(Taylor & Stone, 2009).

While the focus of the Transfer Learning field has been on transferring advice from one machine learning agent to another, the use of humans in the field has been consistent. Human-sourced advice is commonly used in source task selection and task mapping. The process of source task selection involves identifying a set of experiences from other domains/agents to transfer to a new agent. Task mapping is the process of identifying the similarities between the source task and the new task so that the agent can quickly incorporate the relevant behaviours. Both activities benefit from human involvement due to our ability to quickly draw connections between the tasks.

Many other fields of Reinforcement Learning that use external information are conceptually similar to Transfer Learning. Fields such as Imitation Learning(B. Argall et al., 2007), Heuristic Reinforcement Learning(Celiberto Jr et al., 2007), and Interactive Reinforcement Learning(Griffith et al., 2013; Knox & Stone, 2010) all attempt to use information or behaviours sourced from another agent. Where these fields differ is in the definition of an agent, and the methodologies used to transfer information between them.

5.1.2 Reinforcement Learning from Demonstration

Reinforcement Learning from Demonstration(B. Argall et al., 2007), also known as Imitation Learning, involves a Reinforcement Learning agent attempting to mimic a demonstration from another agent, usually a human. A demonstration is typically a display of the desired behaviour typically provided as a sequence of state-action pairs. This sequence can be generated from experience from another agent, video of the desired behaviour, or tutelage by a human. Whatever the format, these demonstrations can constitute the human-supplied advice and objective evaluation of the desired behaviour. The sequence of state-action pairs, or demonstration, can be used to generate an initial value function for the Reinforcement Learning agent. Alternatively, an initial policy can be derived from the demonstration that can be used to kickstart the Reinforcement Learning agent. This second process is conceptually similar to transfer learning, with the difference being the source of the initial information. In this case, the source is a demonstration rather than mapping from another agents policy.

5.1.3 Interactive Reinforcement Learning

Interactive Reinforcement Learning uses human-sourced advice to directly interact with the agent while it is learning/operating, thus the term ‘interactive’ (Thomaz et al., 2005). Humans interact with the agent by providing additional rewards in response to the agent’s performance, or by recommending actions to the agent to guide the exploration process.

Like many fields of Reinforcement Learning that utilise external information, interactive reinforcement overlaps quite a lot with transfer learning and learning from demonstration. As such, existing transfer learning methods may be interactive, and existing Interactive Reinforcement Learning methods may be implementations of transfer learning.

The focus for Interactive Reinforcement Learning is limited to the use of advice during the Reinforcement Learning process, not before or after. This limitation requires interactive techniques to be easy for an agent to get information from, and for humans to add information to so that the learning process is not slowed down. This limitation also means that the agent or policy should not be reset when new information is provided, as that is conceptually similar to creating a new agent rather than interacting with an existing one.

5.2 Advice Style

Transfer Learning, Reinforcement Learning by Demonstration, and Interactive Reinforcement Learning each use human-sourced advice in different capacities. Transfer learning uses direct human interaction to identify common skills, humans in Reinforcement Learning by demonstration provide examples of the desired behaviour, and Interactive Reinforcement Learning asks humans to tutor the agent while it operates (Suay et al., 2016; Brys et al., 2015). While implementations using advice differ between fields, each implementation falls into one of two categories. Either the human is evaluating the current behaviour of the agent, or they are informing the agent about what behaviour should be learnt. This distinction of advice delivery styles, evaluative vs informative, is one contribution of this dissertation and is the focus of the remainder of this chapter.

5.2.1 Evaluative Advice

Evaluative advice is information that critiques current or past behaviour of an agent. Advice that supplements, improves, or creates a reward function is considered to be evaluative as it is a reaction to an agent's behaviour rather than an influencer of an agent's decision making. The source of the advice is what separates evaluative advice from a reward function. A typical reward function is defined by the environment, whereas evaluative advice originates from an observer of the agent or other external sources.

Evaluative advice tends to be the easier of the two advice types to implement. Humans providing evaluative advice do not need to know the solution to a problem, it is enough for to be able to assess the result of an action and then decide whether it was the correct action to take. As the saying goes, hindsight is twenty-twenty. A machine-learning manifestation of this saying is TAMER(Knox & Stone, 2010, 2009), in which, a human continually critiques a Reinforcement Learning agent's actions. The human observes an agent, and in response to the agent's actions, provides a simple yes/no evaluation of its choice in action. The Boolean evaluation acts as an additional reward signal, supplementing the reward function of the environment. This bare minimum of human influence is enough to significantly decrease the time required by the agent to learn a suitable behaviour.

5.2.2 Informative Advice

Informative advice is information that aids an agent in decision making. Advice that recommends actions to take or avoid, suggests exploration strategies, provides information about the environment or proactively alters what action an agent may take is considered to be informative. Informative methods primarily focus on transferring information from the human and encoding into the agent's policy, either directly, by altering the policy, or indirectly by influencing the agent's decision-making process.

Utilising informative advice can be challenging for two reasons, the first of which is the human factor. Informative advice typically requires the human to know what the correct action is for a given state ahead of time. Not only does this require a greater understanding of the environment and the agent's position within it, but it also requires a more substantial commitment of time and effort to provide the advice. The time and effort required increases

as the size of the environment, and the available actions increases. The second reason utilising informative advice is challenging is that encoding information sourced from a human into a form an agent can understand can be a complicated process, as it is more informationally dense than evaluative advice.

An implementation of informative advice in Interactive Reinforcement Learning is the ADVISE algorithm (Griffith et al., 2013). In ADVISE, a human observing an agent in operation can recommend actions to take at any given step, which the agent may choose to follow. This methodology allows the human to guide the agent through parts of the environment in which they are familiar with. This can result in a significant improvement over existing Interactive Reinforcement Learning methods and a reduced need for exploration.

5.2.3 Evaluative versus Informative

Evaluative advice has seen more use in past research as implementations have not typically needed real-time interaction between the human and the agent. Instead, the advice is sourced ahead of time and encoded into the agent before operation. Reinforcement Learning from Demonstration and Transfer Learning are good examples of this, in which the human provides their input ahead of time. In the case of Reinforcement Learning by Demonstration, the examples are collected before the agent begins training, and are encoded to create a reward function or kickstart the agent’s value function.

Evaluative advice is also more straightforward to encode ahead of time as the focus tends to be on the result of a decision rather than on what decision should be made. This is due to it being easier to determine if an action was the correct or incorrect action to take once the result of the action is available. Most implementations of evaluative advice alter or supplement the reward function of the environment. Encoding information to alter the reward function is straightforward, as the primary focus is on whether to increase or decrease the reward given to the agent, as opposed to informative implementations that attempt to alter the decision-making policy. Additionally, providing an evaluation requires less human effort than determining what information or action is relevant for a given state, as the information sought is typically a Boolean or scalar measurement. Overall, evaluative advice is more straightforward to source, implement, and encode than the informative counterpart.

Informative advice tends to be more informationally dense than evaluative advice. While this does make sourcing and encoding the information difficult, it does provide more benefit to the agent. Evaluative advice only reinforces behaviour after that behaviour has been exhibited, whereas informative advice can promote or discourage behaviour before it is presented. Advice that recommends taking or avoiding actions will reduce the search space for the agent, resulting in improved learning time. The downside of this is that if the agent never performs actions that are pre-emptively discouraged, and the advice is not optimal, then the optimal policy may not be found.

A direct comparison of the two types is difficult as the implementations of human-sourced advice vary. However, one paper has performed a direct comparison of the effects of informative versus evaluative advice on agent learning. Griffith *et.al* (2013) compared their algorithm, ADVISE, against a state of the art evaluative algorithm of the time, TAMER. Both algorithms utilise Interactive Reinforcement Learning agents and source advice on a step by step basis. The ADVISE algorithm prompts the human for a recommended action which the agent can then follow, while TAMER prompts the human for a Boolean evaluation on the previous action taken. In the experiments, each agent is assisted by a simulated human, making the advice comparable. However, the action set in each of the environments used each consist of four actions.

The ADVISE algorithm allows the human to recommend an action ($n=4$), while TAMER allows the human to provide a Boolean evaluation ($n=2$); therefore the information gain from ADVISE is greater than TAMER and may bias the results. If this is ignored, the experiments show that informative advice is more beneficial to the agent regardless of advice accuracy for the majority of cases. The experiments in the ADVISE paper use an oracle, a simulated human that can provide consistent advice and does not suffer from biases introduced by real humans. As a result, information regarding actual human-sourced information, such as accuracy and engagement, is not available for Interactive Reinforcement Learning agents.

5.2.4 Human Engagement

Studies on human engagement and teaching styles when engaging with interactive machine learning agents, independent of the type of advice, have been performed (Amershi et al.,

2014; Kamar et al., 2012). A recent comprehensive study looking at the engagement between humans and interactive machine learning presents some case studies demonstrating the use of humans as information sources in machine learning (Amershi et al., 2014). Highlighted in this study was the need for increased understanding of how humans engage with machine learning algorithms, and what teaching styles the users preferred.

A study by Thomaz and Breazeal (2008), later confirmed by Knox and Stone (2012) found that human tutors tend to have a positive bias when teaching machines, opting to reward rather than punish Reinforcement Learning agents. This positive bias leads to agents perusing the rewards provided by the human over the reward function of the environment. This positive bias has been tested and observed in agents that receive evaluative advice, as it tends to be provided as a reward. No such bias has been tested for or observed yet in informative-assisted agents. Knox and Stone (2013) later mitigated the consequence of the positive bias in Reinforcement Learning agents by developing an agent that valued human-reward gained in the long term rather than the short term (Knox & Stone, 2013).

A 2010 study performed by Cakmak and Thomaz (Cakmak & Thomaz, 2010), investigated the strategy of teachers when tutoring machine learning agents. The study found that humans paid to provide advice to a system over an extended period experienced frustration and boredom when bombarded with questions from the agent. The stream of questions to the teachers caused some participants to turn their brain off or lose track of what they were teaching according to self-reports (Cakmak, Chao, & Thomaz, 2010). This observation was confirmed by a movie recommendation system developed for Netflix, where participants were repeatedly asked to state if the system was right or wrong (Guillory & Bilmes, 2011b, 2011a).

These studies suggest participants do not like being prompted for input repeatedly, mainly when the input can be repetitive. Current Interactive Reinforcement Learning systems do not prompt the user for information, instead, allowing the human to step in whenever they wish. However, input into these systems is repetitive and requires the users to provide advice on a state by state basis, leaving current systems susceptible to the same issues of frustration and interruptibility as the active learning systems reported. Whether these issues translate to Interactive Reinforcement Learning is yet to be observed.

5.3 Experiment

Human engagement is a measure of the commitment of the human to the objective. While machine learning aims to create faster agents capable of solving more complex problems, for agents that interact with humans, additional objectives exist. Assisted Reinforcement Learning agents aim to gather as much information from the human as possible, as this can equate to improved performance within the environment. The higher the human engagement, the higher the opportunity to transfer knowledge to the agent. The accuracy of the advice and information gain as a result of the advice provided is also important, as they contribute to the policy being learnt by the agent.

This experiment aims to measure the human engagement, accuracy of advice, and the information gain for evaluative and informative advice for Interactive Reinforcement Learning. The performance of the agent, or its ability to solve the problem, is not a concern of this experiment. A comparison of evaluative and informative advice on the performance of the agents has been performed in a prior study (Griffith et al., 2013).

In the context of this chapter, human engagement is a measure of the number of interactions, the total time spent constructing interactions, and the distribution of interactions over the time the agent is operating. The human observing is given an opportunity to provide information once per step of the agent, and if the human does provide some advice during that step, then an interaction is recorded. A measure of the number of interactions is not sufficient, as the time and effort required to provide an interaction may differ between informative and evaluative advice methods. As a result, interaction time is also recorded. The accuracy of the information provided to the agent affects its performance within the environment. Advice accuracy is a measure of how accurate the information provided by the human is, compared to the optimal action to take for each state the agent encounters.

5.3.1 Methodology

An experiment has been designed to compare human engagement for evaluative and informative advice delivery styles, which measures human engagement and advice accuracy. Twenty participants, ten for each advice delivery style, communicate with a Reinforcement

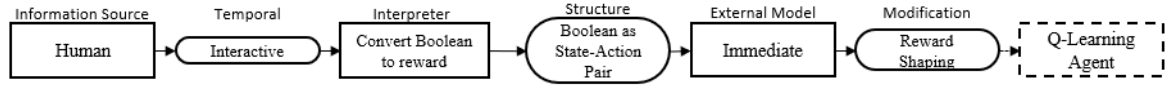


Figure 5.1: Definition of the evaluative assisted experimental agent using the Assisted Reinforcement Learning framework.

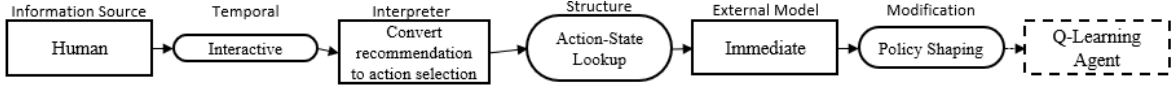


Figure 5.2: Definition of the informative assisted experimental agent using the Assisted Reinforcement Learning framework.

Learning agent while observing its current state and performance.

The experiment consists of two Q-Learning Interactive Reinforcement Learning agents attempting to solve the Mountain Car problem, one accepting evaluative advice (Figure 5.1), and the other accepting informative advice (Figure 5.2). A participant interacting with the evaluative agent may provide a yes/no agreement to the agent’s choice of action for the last time step. The boolean evaluation is then used by the agent to supplement the reward it receives from the environment. A positive evaluation adds +1 to the reward, while a negative evaluation subtracts from the reward. A participant interacting with the informative agent may recommend an action for the agent’s next step, either left or right. If the agent is recommended an action then that action will be taken, otherwise, the agent uses an e-greedy policy. Each participant, regardless of teaching style, has three possible options each step. For the evaluative advice participants this is agree, disagree, or do nothing. For the informative advice participants, the options are to recommend left, right, or to do nothing.

This trial is not concerned with the performance of the two advice delivery style as that has been compared in existing literature (Griffith et al., 2013), only in the differences in the participant’s level of engagement and accuracy for each advice style. The agents were given a low learning rate, manually tuned to extend the time in which the agent would take to learn a suitable behaviour on its own. This was chosen so that the focus would be on the human’s input rather than on the agent’s capabilities. Both agents were given a learning rate of 0.25, a discounting of 0.9, and used an e-greedy action selection strategy with an epsilon of 0.05.

The Mountain Car environment has been chosen as an optimal policy for the problem is known. Having an optimal policy for the environment allows the accuracy of the human-

sourced information to be measured. Additionally, the Mountain Car problem has a low state and action space, allowing for the humans to observe the impact of their interactions relatively quickly, as the agent is likely to encounter the same state-action pairs frequently.

The participants chosen for the experiment have not had significant exposure to artificial intelligence, and are not familiar with the Mountain Car environment. Before beginning the experiment, each participant is given a five-minute induction to the Mountain Car problem, and then asked to complete a short questionnaire. The induction introduces the aim of the agent, the dynamics of the environment, the action space, and most significantly, what the optimal solution to the problem is. The solution for the environment is described to the participant to give all participants an equal understanding and to reduce the time that they spend exploring the environment themselves and focus on assisting the agent. The script for describing the optimal solution is outlined below.

“The car [agent] begins at the bottom of the valley, between two mountains. The car aims to drive to the top of the mountain on the right side. However, the car does not have the power to drive directly up the mountainside; instead, it needs to build up momentum. Momentum is gained by driving as high as possible on one side of the mountain, then turning around and accelerating in the opposite direction. When the car reaches the highest point it can on the opposite side, the process is repeated. Eventually, the car will gain enough speed to reach the top of the mountain.”.

When the induction is complete, the participant is asked to begin a questionnaire. The questionnaire consists of seven questions, the first two of which aim to gauge the participants level of understanding of the Mountain Car problem (See Appendix A). After completing the first two questions, the participant is ready to begin the experiment.

The participant is given 500ms to provide advice to the agent each step. To provide advice to the agent, the participant presses one of two keys on the keyboard to indicate either approval/disapproval of the agent’s last choice in action, or to recommend the left/right action for the agent to take next. The input mechanism is dependent on the advice delivery style being tested. If the human provides advice within the 500ms window, an interaction has taken place and the time taken to create that interaction is recorded. If the human does not provide advice within the window provided, then no interaction is recorded, and the agent

operates as usual. Additionally, the human can change the duration of the window by 25% during the experiment by pressing the +/- keys.

The experiments run until the participant believes the agent has learnt the correct behaviour, or until they have had enough, at which point the agent is terminated.

After the participant has chosen to stop providing advice, they are asked to complete the remainder of the questionnaire (See Appendix A). The remaining five questions are aimed to gauge the participants understanding of the Mountain Car problem now that they have experienced the environment. It also aims to capture their perception about their level of engagement, the accuracy of their advice, and the agent’s understanding of the advice supplied.

5.3.2 Results

Before each participant began interacting with the agent, they were asked to answer two questions from the questionnaire (See Appendix A). The purpose of the questionnaire is to gauge the participants understanding of the problem environment and there interactions with the agent. The first question asked if the participant has been involved in a machine learning study in the past. None of the twenty participants reported being involved in a machine learning study previously.

Before starting the experiment and answering the questions on the questionnaire, participants were talked through the environment dynamics and the optimal behaviour the agent was to learn. After this brief explanation, participants were asked to rate their level of understanding of the environment on a scale from zero to ten. After interacting with the agent, participants were asked the same question again. Table 5.1 shows the average responses from the two groups of participants.

Interestingly, there is a small difference in the participants self reported understanding of the environment before they begin interacting with the agent. The only difference in the explanation given to the two groups was the details on how they will interact with the agent. The participants in the evaluative group were asked to rate the agent’s choice of action as good or bad, while the informative participants were asked to recommend an action, either left or right. The difference in reported understanding before the experiment may be too small

SELF-REPORTED UNDERSTANDING OF ENVIRONMENT				
ADVICE/AGENT	BEFORE EXPERIMENT	AFTER EXPERIMENT	CHANGE	
EVALUATIVE	3.8	6.5	+2.7	
INFORMATIVE	1.9	6.2	+4.3	
BOTH	2.85	6.35	+3.5	

Table 5.1: The participants self-reported understanding of the solution and dynamics of the Mountain Car environment. Participants rated their understanding on a scale of 0 to 10 before and after assisting the agent.

to be significant, but could indicate that informative advice delivery is easier to understand.

A change in the level of participants self-reported understanding is observed after the experiment. Both groups reported a greater understanding of the environment after assisting the agent. There was no difference between the two group however.

SELF-REPORTED LEVEL OF ENGAGEMENT			
ADVICE/AGENT	ANSWER A	ANSWER B	ANSWER C
EVALUATIVE	2	8	0
INFORMATIVE	1	8	1
BOTH	3	16	1

Table 5.2: The participants self-reported level of engagement with the agent. Participants reported that they (A) could have spent more time with the agent, (B) were happy with how much time they provided, or (C) spent too much time with the agent.

After the experiment, participants were asked to report how they felt about their level of engagement with the agent. They were given three options.

(A) I could have spent more time interacting with the agent.

(B) Im happy with how much time I interacted with the agent.

(C) I spent too much time interacting with the agent.

Table 5.2 shows the participants reported level of engagement with the agent. There is no significant difference between the two groups. The majority of participants were content with the level of engagement they had with the agent.

Participants were asked to report what they thought their level of accuracy was throughout the experiment. Participants were given six options, ranging from always incorrect to

SELF-REPORTED LEVEL OF ACCURACY				
ADVICE/AGENT	SOMETIMES INCORRECT	SOMETIMES CORRECT	MOSTLY CORRECT	ALWAYS CORRECT
EVALUATIVE	5	2	3	0
INFORMATIVE	0	2	7	1
BOTH	5	4	10	1

Table 5.3: Participants self-reported level of accuracy. Participants rated the accuracy of the advice they provided from '*Always Incorrect*' to '*Always Correct*'. The columns '*Always Incorrect*' and '*Mostly Incorrect*' are not shown as no participants reported these options.

always correct. Table 5.3 shows the self-reported results. The results indicate that informative participants were more confident in the advice they provided.

PERCEPTION OF AGENT'S ABILITY TO FOLLOW ADVICE	
ADVICE/AGENT	AVERAGE
EVALUATIVE	6.6
INFORMATIVE	7.9
BOTH	7.25

Table 5.4: Average of participants self-reported sentiment of how well the agent followed the advice provided. On a scale from zero (Never), to ten (Always), participants scored the agent's ability to follow advice.

Participants were asked to rate how well they thought the agent followed their advice. On a scale from zero (Never), to ten (Always), participants scored the agent's ability to follow advice. The results, shown in Table 5.4, show that participants using informative advice perceived the agent to follow advice better than the evaluative agent.

AVERAGE EPISODES THAT ASSISTANCE WAS PROVIDED			
ADVICE/AGENT	AVERAGE	MIN	MAX
EVALUATIVE	5.4	1	21
INFORMATIVE	12.9	2	30
BOTH	9.15	1	30

Table 5.5: The average number of episodes that participants provided advice for on the Mountain Car environment.

Table 5.5 displays the number of episodes that each set of participants interacted with the agent for. An episode that contained at least one interaction is recorded as being an interaction episode. The data collected shows a large variation in the length of engagement between the two types of advice types. On average, participants providing informative advice,

advice that recommends an action to take, informed over two times as many episodes than participants providing evaluative advice, critiquing a past action.

AVERAGE INTERACTION PERCENTAGE			
ADVICE/AGENT	AVERAGE	MIN	MAX
EVALUATIVE	26.860 %	0.999 %	73.223 %
INFORMATIVE	47.316 %	19.123 %	93.611 %
BOTH	37.088 %	0.999 %	73.223 %

Table 5.6: Percentage of steps that participants provided advice for the Mountain Car problem.

As demonstrated in previous work (Griffith et al., 2013), agents assisted by informative advice learn quicker than agents assisted by evaluative advice. The increase in learning speed results in fewer steps per episodes for environments with a termination condition, such as the Mountain Car environment used in these experiments. This decrease in steps per episode for informative assisted agents gives fewer opportunities for the user to provide advice, as only one interaction may occur each step. As a result, the number of interactions per episode is not a suitable measure of engagement. Instead, the number of steps in which an interaction occurred is used to measure engagement. Table 5.6 shows the average interaction percentage for the two sets of participants. The interaction percentage is the ratio of interactions to interaction opportunities. Using this measurement, the length of episodes is disregarded. The results show that participants using an informative advice delivery method interaction almost three times as often as their evaluative counterparts. Despite the higher rate of interactions, both groups reported they were happy with their level of engagement with the agent (Table 5.2). This may indicate that participants prefer to provide informative advice, or that the sample size was not large enough.

Table 5.7 displays the accuracy percentage of the advice provided by each of the groups of

ACCURACY OF ADVICE PROVIDED			
ADVICE/AGENT	AVERAGE	MIN	MAX
EVALUATIVE	48.27 %	36.00 %	57.14 %
INFORMATIVE	94.81 %	92.67 %	96.94 %
BOTH	71.54 %	36.00 %	96.94 %

Table 5.7: The percentage of interactions in which the advice provided was optimal for the state/action.

participants. An accurate interaction is one which provided the optimal advice for the agent in that state. Accuracy is a measurement of the number of correct interactions compared to the total interactions. Accurate informative interactions were observed almost twice as much as accurate evaluative informative interactions. These results reflect the self-reported accuracy shown in Table 5.3.

One hypothesis for the large difference in accuracy is latency. Latency is the time it takes for the human to decide on the advice to provide, and then input it into the agent. It is possible that if the human is too late in providing advice, then the advice will inadvertently be provided to the state after the one intended. For the Mountain Car environment, a late interaction is more likely to remain accurate in the next state for informative advice than it is for evaluative advice. This is due to the layout of the state-space and the nature of untrained agents. The optimal action for a state in the Mountain Car environment is likely to be the same as its neighbouring states. This is due to the optimal behaviour being to accelerate in a single direction until velocity reaches 0. A recommended action that is received in the state after the one intended is likely to be the correct action, regardless of latency. This doesn't apply to evaluative advice. The participants assisting the evaluative agent are not providing a recommended action, instead they are critiquing the agent's last choice in action. An untrained agent has a largely random action selection policy, and is therefore not likely to choose the same action twice in a row. As the agent's chosen action may have changed by the time it receives advice from the user, the accuracy suffers.

This hypothesis is supported by the state breakdown of the advice accuracy. Figure 5.3 shows the accuracy of participants' advice for each state in the environment. The darker the colour, the more accurate the advice supplied by the participants for that state. The comparison of the two heat-maps supported the earlier observations of the two accuracies; informative is much more accurate than evaluative advice. The informative advice heat map shows that the states with the most inaccuracy are at the middle of the environment, where the optimal action changes. This inaccuracy is likely not due to poor participant knowledge, rather providing advice late, after the agent has moved beyond the centre threshold.

The heat map for the evaluative advice shows that accuracy differs wildly across the environment and does not have an obvious pattern. The poor result for accuracy of evaluative

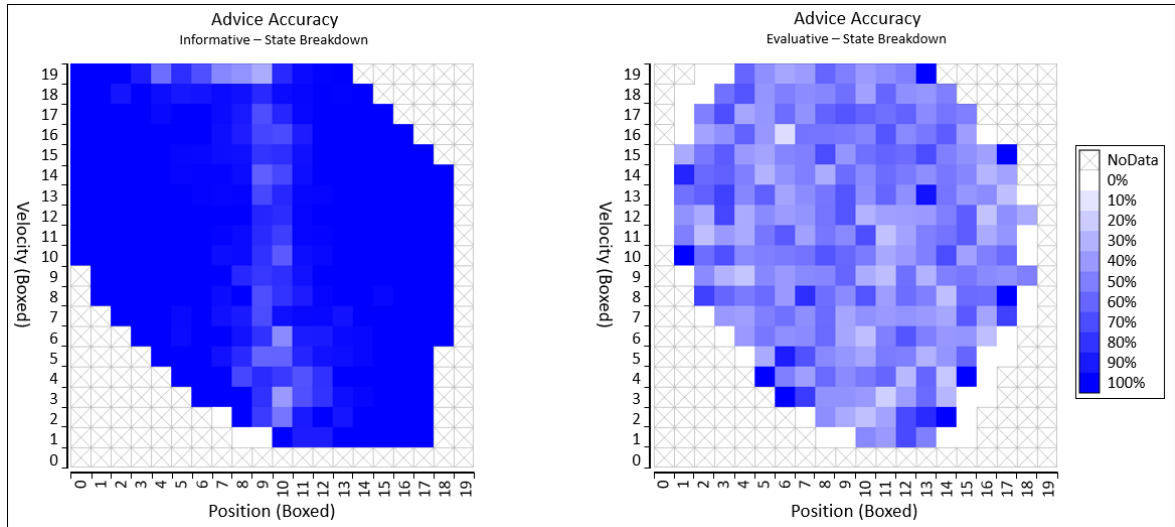


Figure 5.3: State-based accuracy of informative and evaluative participants for Mountain Car environment.

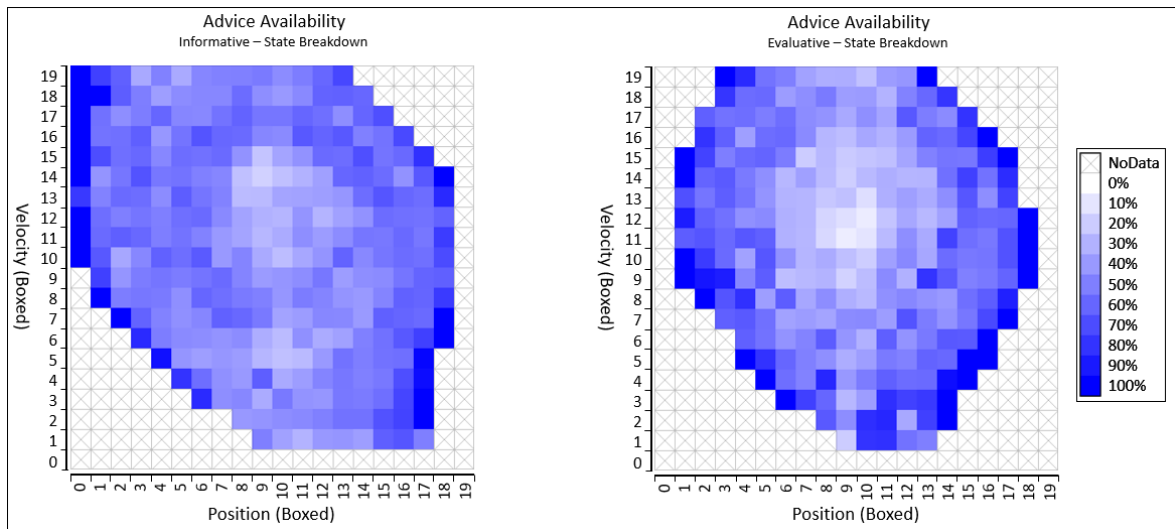


Figure 5.4: State-based availability of informative and evaluative participants for Mountain Car environment.

advice is likely due to the latency of advice delivery coupled with the lower probability that the advice will still be accurate to the following state compared to informative advice. Additionally, evaluative advice may have lower accuracy as it requires the human assessing each state-action pair. Informative advice giving may require less time assessing each state, as the human may be following a set of rules for action recommendation, and that the next state is easier to predict compared to the agent’s next choice in action.

Figure 5.4 shows the availability of advice for each state. Availability here is a measure of how often the user provided advice in a state compared to the number of times the agent visited the state. The darker a state is on the heat map, the more often the user provided advice for that state. The agent that was assisted by informative advice was able to achieve higher velocities in the environment, and as a result, visited more states. One pattern that can be observed is that the states on the edges of the heat maps have a higher availability than those in the centre. These edge states are visited when the agent has learnt a suitable behaviour, making the evaluation and recommendation of actions easier on the user, and increasing engagement. The edge states tend to be the last few states the users provided advice to, before voluntarily ending the experiment.

Figure 5.5 and Table 5.8 show the reward bias of the participants providing evaluative advice. A deviation from fifty percent indicates reward bias. The results collected show that all participants provided more positive evaluation than negative evaluation, an observation backed up by existing literature (Amershi et al., 2014).

EVALUATIVE ADVICE BIAS			
ADVICE/AGENT	AVERAGE	MIN	MAX
EVALUATIVE	66.22 %	57.14 %	100.00 %

Table 5.8: Reward bias of evaluative advice. Above 50% means that the advisor provided more positive evaluation than negative evaluation.

5.3.3 Conclusion

The human trial performed in this chapter served to investigate the engagement of human advice givers when assisting Interactive Reinforcement Learning agents. This chapter identifies two methods of providing assistance, evaluative and informative. Evaluative assis-

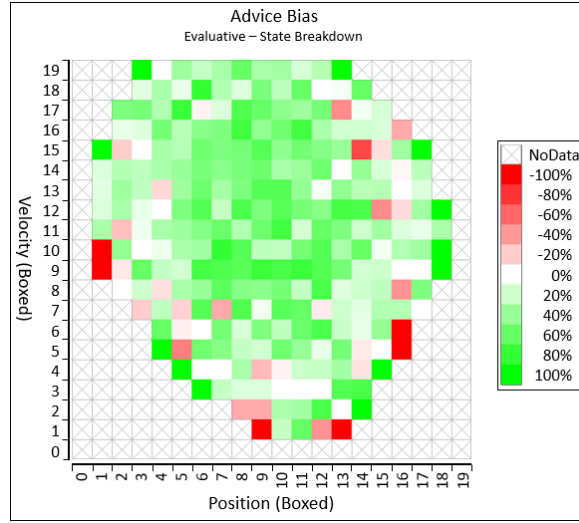


Figure 5.5: Reward bias of evaluative advice. State-based. Above 50% means that the advisor provided more positive evaluation than negative evaluation.

tance assesses the past performance of an agent, while informative assistance supplements future decision making. Previous work in the field has compared the performance of Interactive Reinforcement Learning agents under the influence of each assistance method, finding that informative-assisted agents learn faster. However, studies on human engagement when providing advice using each assistance method have not been performed.

In this chapter, the results from the human trial showed that advice-givers providing informative advice outperformed those that used evaluative advice. Humans using an informative advice-giving method demonstrated more accurate advice, assisted the agent for longer, and provided more advice per episode. Additionally, informative advice givers rated the ability of the agent to follow advice higher, perceived their own advice to be of higher accuracy, and were similarly content with their engagement with the agent as the evaluative advice-giving participants.

In the next chapter, simulated users are introduced as a method of replicating the participants from this chapter, without the need for a time-consuming human trial or accompanying ethics approval trial.

Chapter 6

Simulated Users

Assisted Reinforcement Learning methods utilise external information sources to evaluate decisions and accelerate learning. Previous chapters showed that human advice could significantly improve an agents performance. When creating Reinforcement Learning algorithms, it is common to repeat experiments as parameters are altered or to gain a sufficient sample size. To require human interaction every time an experiment is restarted is undesirable, particularly when the expense in doing so can be considerable. Additionally, reusing the same people for the experiment introduces bias, as they will learn the behaviour of the agent and the dynamics of the environment. Ideally, for early experiments when indicative results or testing is sufficient a human analogue would be used, and only when testing and development is completed would a real human trial take place.

This chapter presents a methodology for assessing assisted Reinforcement Learning agents using simulated users. Simulated users allow human knowledge, bias, and interaction to be simulated. The use of simulated users allow the development and testing of Reinforcement Learning agents, and can provide indicative results of agent performance under defined human constraints. While simulated users are no replacement for actual humans, they can provide a reliable and practical alternative for developing and evaluating assisted agents.

6.1 Role of the Human

The purpose of the human in Interactive Reinforcement Learning is to provide contextually relevant information to the agent. The information is used to leverage the agent’s learning process, either by supplementing the reward function, altering the policy, or adjusting how the agent makes decisions. One aim of Interactive Reinforcement Learning is to make the process of providing advice to an agent as simple as possible, by using methods intuitive

to the advice-giver, and by increasing the utility of each piece of advice given to reduce the need for continued interactions. While human advice can be beneficial to an agent, as shown Chapter 5 and other existing research (Griffith et al., 2013; Knox & Stone, 2010; Brys et al., 2015), it does come with many challenges, the first of which is the variability of advice givers and the information they provide.

6.1.1 Characteristics of Human Interactions

There are several characteristics of human advice givers that define the applicability and quality of the information they provide. These characteristics of human-sourced information may need to be considered when utilising such information.

- **Accuracy:** Accuracy is a measure of how appropriate information is to the current situation. An information source may be inaccurate due to confusion, poor knowledge, noise, or intentional maliciousness.
- **Availability:** The information source may not be available all the time or may not respond in the time provided.
- **Concept Drift:** The intentions of the agent and the intentions of the information source may shift over time, such that each is attempting to work towards a different goal or with a different understanding of the environment.
- **Reward Bias:** Advisors may have a preferred teaching style, or favour positive or negative reinforcement.
- **Cognitive Bias:** An advisor’s preconceived thoughts about the nature of the agent and the knowledge they have available to advise the agent in decision making. Advisors are likely to provide advice related to the areas of the domain that they know about and neglect the areas where they know little.
- **Knowledge Level:** An advisor may have little information about all aspects of a problem (breadth), or expert information about a single aspect (depth). Knowledge level may change over time as the advisor observes the dynamics of the agent or environment.

- **Latency:** Latency is a measure of the time taken to retrieve information from the information source. If the latency is too high, then the information may be applied to the wrong state.

These characteristics of human-sourced information can present difficulties when attempting to utilise humans as information sources. Difficulties interacting with people is a fact of life, as such, any agent that interacts with a human is going to experience these issues, and practices should be in place to manage or mitigate them.

6.1.2 Problems with Human Testing

There are some challenges that need to be considered when using human-sourced information in experiments, the first of which is expense. Acquiring and employing people for use as an information source can be expensive. The expense rises as the number of participants and the required domain expertise increases. The second concern is the time requirements and constraints. The process of sourcing and training participants can be considerable, as well as the time they spend interacting with the agent. The time to perform the experiment can be substantial, but in academia, the time required to get ethics approval is often far greater.

A third concern is the repeatability of experiments involving humans. Repeating experiments is essential to gather sufficient sample sizes and identify results with statistical significance, but results in increased time and expense. Additionally, participants become increasingly biased as they familiarise themselves with the processes and dynamics of an experiment or become tired or uninterested. These biases can affect interaction characteristics such as knowledge level, latency, cognitive bias, and availability. One solution is to use new participants for each experiment. However, participants with the required skills can be difficult to source and increase time and expenses. This leads to the final concern, variability. Variability between participants can lead to a wide disparity in results, depending on their various interaction characteristics. Some characteristics such as knowledge level, latency, or accuracy can be mitigated by pretesting participants, while others such as cognitive bias are difficult to identify. All pre-tests and screening add time and expense to the experiment. If variability between participants can not be reduced, then larger sample sizes are required to

achieve statistically reliable results.

Expense, time, repeatability, and variability are all barriers when performing human trial, when training a Reinforcement Learning agent for real-world use, the benefit can out way the cost. For testing and development however, having to repeat human trials each time a parameter is changed, or when the agent is under development and testing, the cumulative cost may not be justifiable. Simulated users offer a consistent, reliable, and quantifiable method for replicating human interactions to a degree suitable for providing indicative results.

6.2 Simulated Users

A simulated user is an automated human analogue designed to replicate the functions of a human user. The purpose is to allow rapid and controlled training and testing. Instead of relying on human assistance, the agent relies on a simulated user whose source of expertise is defined ahead of time. They offer a quantitative method for representing and simulating human interactions for the evaluation and training of assisted machine learning methods.

While simulated users are not a replacement for actual humans, they do offer a suitable method for gathering indicative results regarding agent performance when assisted. Simulated users are not a method for acquiring new information about a behaviour or problem, instead, they require an existing solution or collection of pertinent information to be of use to the agent. As such, simulated users are limited to the testing and evaluation of agents on existing problems and not against new domains. A criticism of simulated users is that if a solution to the problem is known, why is a Reinforcement Learning agent needed? The answer is that simulated users function as a means of testing and comparing agents to identify their strengths and weaknesses before conducting a rigorous human trial.

A simulated user is designed to act as a human would in predefined circumstances. Depending on the complexity and characteristics required, simulated users can, and should, be designed to reflect the qualities of the humans that would otherwise be aiding the agent. It is important to exhibit the characteristics of human-sourced information so that accurate evaluations can be performed. Human-sourced information is noisy, and the simulated users should reflect this. While it is impractical to assume that the range of human characteristics can be completely and accurately replicated, simulated users can provide the necessary

functions required to develop and test ARL systems in place of actual human testing. Furthermore, simulated users can enforce consistency and repeatability in evaluation, something that humans cannot provide,

6.2.1 Current Applications of Simulated Users

The field of expert systems use simulated users to evaluate knowledge acquisition methods because they can facilitate controlled experiments (Cao & Compton, 2005; Dazeley & Kang, 2004a). Expert systems face the same challenges as Interactive Reinforcement Learning; expense, time, repeatability, and variability. Users organise their knowledge and priorities quite differently from one another, complicating the method in which controlled studies are performed. Simulated users assist in solving this issue in expert systems, accounting for both variability and scarcity of users (B. H. Kang, Preston, & Compton, 1998; Compton, Preston, & Kang, 1995; Dazeley & Kang, 2004a). Compton (Compton et al., 1995) suggests that the use of simulated user evaluation is possibly the only way to reliably and empirically compare different expert systems.

Spoken dialogue systems (SDS) also employ simulated users for evaluation (Papaioannou & Lemon, 2017; Georgila, Henderson, & Lemon, 2006; Scheffler & Young, 2002). The development, training and evaluation of SDS requires extensive time interacting with humans, an expensive and time-consuming practice. In response, the field adopted the practice of ‘user modelling’. User modelling, like simulated users, attempts to design a representation of the intended audience of an SDS. (Misu, Georgila, Leuski, & Traum, 2012). The advantage of adopting user modelling and simulated users to produce training data is that the characteristics of the simulated user can be modified to represent different intended audiences, allowing for better evaluations of the dialogue policies created.

Expert systems and spoken dialogue systems are leading the development of simulated users. Both areas are challenged by the issues arising from human involvement (Georgila, Henderson, & Lemon, 2005; Compton et al., 1995). These fields have shown the benefits that simulated users have for training and evaluation in their respective fields. The development of the simulated user field in these two areas is progressing independently of each other; this implies a lack of structure and terminology about simulated users. The remainder of this

chapter begins to address this issue, by identifying the different models of simulated users, the principles that they adhere to, and a straightforward method for reproducing some of the characteristics of the users they represent.

6.2.2 Evaluation Principles

The ability for a simulated user to adequately replicate human interactions relies on its model of the user it is representing. It is important to build a comprehensive and accurate model for the simulated user if reliable indicative results are to be gathered. The strength of a simulated user can be assessed by its adherence to three fundamental principles. These principles are consistency, completeness, and variation. (Rieser & Lemon, 2006) first proposed these principles as a novel way of assessing the ‘naturalness’ of spoken dialogue systems. These principles are well-suited to the evaluation of simulated users also.

- The **Principle of Consistency** states that simulated users should not take actions or provide information that the intended user would not. This principle is constrained to the context of the system being developed and the experimental parameters being tested.
- The **Principle of Completeness** states that simulated users should produce every possible action that the intended user may take. The more complete the action range of the simulated user is the more thorough and accurate the evaluation can be.
- The **Principle of Variation** states that simulated users should behave like the users they are modelled from, while not replicating average behaviour completely. To effectively replicate a real user, simulated users must produce outliers and perform unintended actions that, while unlikely, real users may perform.

A simulated user that adheres to these three principles can create a comprehensive system for evaluation. However, while this system is complete in the sense of interaction, it still does not completely reflect the full range of human characteristics, as described in Section 6.1.1.

6.2.3 Representing Human-Sourced Information

The ability to represent characteristics of human interaction is crucial for designing simulated users to a standard adequate for indicative evaluation. The intended user of a system is likely not perfect, so the simulated user should not be either. Simulated users that model characteristics inherent to human interaction allow a broader evaluation of the agent, as the values of the characteristics can be changed, and the effect measured. For example, a simulated user that can model accuracy of the intended user may have the level of accuracy decreased after each experiment, and the effect of interaction accuracy on the performance of the agent can be measured. Three methods for modelling user behaviour have been identified. The models used may be probabilistic, heuristic, stochastic, or any combination of the three. The models are used to designate how the simulated user will respond to the observation given to it by the agent or environment.

- A **probabilistic model** uses a data-driven approach for representing the intended user of the system (Scheffler & Young, 2001; Hofmann, Schuth, Whiteson, & de Rijke, 2013). The simulated user's behaviour is defined by probable action choices, probabilities determined by observations of real user behaviour. For example, if users were observed to take action A in 40% of cases, and action B in the remaining, then this would be replicated in the simulated user.
- A **heuristic model** is a deterministic approach for representing the behaviour of a simulated user. Among the most common methods for representing information deterministically are hierarchical patterns (Cuayáhuatl, Renals, Lemon, & Shimodaira, 2006) and rule sets (Celiberto Jr et al., 2007). Heuristic models are simple to create and maintain, and require little effort to modify. This approach works well when there is little information known about the intended user, but that information is thought to be accurate and reliable.
- A **stochastic model** is an approach used to simulate processes that fluctuate over time, often within a boundary. While it may appear to be like the probabilistic model, stochastic models have a random probability distribution. Examples of stochastic processes include speech and audio signals, data such as temperature and pressure, and

medical signals such as EEG and EKG (Liang, Balasingham, & Byun, 2008). This approach to modelling is useful for representing complex data and simulating indeterminacy from the intended user.

Each model’s ability to represent the characteristics of human interaction identified in Section 6.1.1 will vary. A survey or design of methods for representing all the interaction characteristics has not been performed and is an area for future research. Section 6.4 illustrates a method for representing accuracy and availability for advice, and other chapters that use simulated users in their respective experiments describe the generated models in greater detail.

6.3 Evaluative Methodology using Simulated Users in Interactive Reinforcement Learning

The primary contribution of the chapter is a methodology for evaluating Interactive Reinforcement Learning agents. Simulated users offer a method for replicating human interactions with a Reinforcement Learning agent, speeding up testing and development, and removing the need for human trials. While a simulated user is not a replacement for a human for comprehensive evaluation, they allow reliable and comparable indicative evaluation when real human trials are not justifiable.

Simulated users enable a methodical and empirical approach to developing IntrRL techniques. This approach is faster and cheaper than using human users, particularly when a broad evaluation of human characteristics is to be tested. Additionally, the use of simulated users does not require human trials or ethics approval, both of which are time consuming and potentially expensive. Additionally, simulated users provide control over the characteristics of human interaction such as accuracy and knowledge level. This control reduces the potential for bias that is often introduced into experiments involving participants. Control of the simulated users model also allows evaluation into the effect of an interaction characteristic on the agent’s performance. For example, how varying levels of interaction frequency effects agent performance.

The methodology for implementing a simulated user is straightforward, consisting of three

phases, construction, implementation, and testing. During the first phase, construction, requirements of the analogue are identified, and the user model is created. The model used to represent the simulated user depends on the interaction characteristic being replicated, as models are better suited to some characteristics than others. Accuracy for example, may be best represented using a probabilistic model, allowing the level of accuracy to be quickly and easily altered. However, knowledge level may be represented heuristically, as a set of rules can be used to generalise a solution for a large state space. Simulated users may be models from results collected from human trials, generated from datasets, or reverse engineered from environment dynamics. Alternatively, multiple models may be generated to cover a range of possible values for a characteristic. For example, rather than gaining a baseline accuracy of advice from human trials, instead a series of simulated users may be generated with varying degrees of accuracy. The results from the set of simulated users can then be used to infer what performance the agent would achieve if assisted by a human of a variable accuracy.

The second phase is implementation. Implementation of the simulated user depends entirely on the field of Assisted Reinforcement Learning it is being used for, and the role the human is to play in the specific implementation. Whatever the field of Reinforcement Learning, the simulated user is used in the same capacity that a real human would be. Interactive Reinforcement Learning employs simulated users to provide evaluation or assisted at the time of learning, while transfer learning uses simulated users to define common behaviours between two domains before learning commences. Section 6.4 provides an illustrative experiment showing the use of an Interactive Reinforcement Learning agent using a simulated user.

The final phase is testing. Testing of the agent is performed in the same way as normal human trials, however now the delivery of advice and the human interaction characteristics can be controlled using the simulated user. As the characteristics of the simulated user can be controlled, the bias introduced from real human trials is reduced. The simulated user is reset after each experiment, allowing repeated experiments without the human analogue becoming more familiar with the problem, or introducing its own bias into the results. Additionally, after each set of experiments, the simulated user can be altered to gather data regarding how the change in participant effects the performance of the agent.

After the experiments have been completed the information collected can not only show the agent’s performance, but this can be compared to the simulated user’s characteristics. This information can allow new insights into agent behaviour, such as how it handles varying degrees of advice accuracy, human availability, concept drift, or knowledge levels.

6.4 Experiment

This section presents an experiment in which an Interactive Reinforcement Learning agent is assisted by a simulated user. The aim of the experiment is to demonstrate the use of a simulated user acting as an analogue for a real user of the system. Furthermore, the experiment evaluates the effect that interaction accuracy and availability have on the performance of a Q-Learning agent in the Mountain Car domain.

The Mountain Car problem is a common benchmark in the Reinforcement Learning field. It has been chosen for this experiment as it the solution is intuitive to human observers, the dynamics of the problem can be generalised with rules, and the results can be compared against the data gathered from the human trial performed in Chapter 5. Full details about the mountain car problem are provided in Chapter 4.

To create the simulated user for this experiment three pieces of information are needed: a model of the information the user will provide, a method for altering the accuracy of the advice, and a method for altering the availability of the user. The simulated user requires a model containing at least partial information about the environment or policy so it can automatically evaluate or assist the agent. This limits the use of simulated users to the evaluation of Reinforcement Learning agents on solved, at least partially solved, problems.

For the Mountain Car problem, a complete solution is known and can be used to create a model for the simulated user. For the following mountain car experiment, the simulated user employs a heuristic model with a set of rules. The simulated user will agree with the agent if the agent took an action that would accelerate it in its current direction of travel, otherwise, the user disagrees. The rule used is *“recommend the action that accelerates the car in the direction of current velocity”*.

The agent used for the experiment is a boolean-evaluated ARL agent (Knox & Stone,

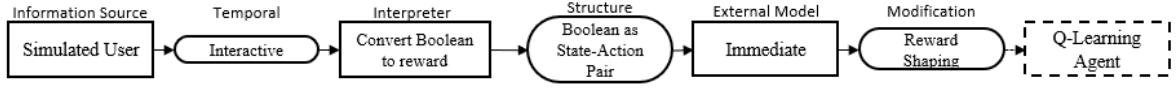


Figure 6.1: Definition of the evaluative assisted experimental agent using the Assisted Reinforcement Learning framework

2008). The agent was given a learning rate of 0.25, a discounting of 0.9, and used an e-greedy action selection strategy with an epsilon of 0.05. A simulated user assists the agent by assessing the agent’s previous action and if user agrees with the action, a reward shaping signal of +1 is given, if it disagrees, then the inverse is given. If the simulated user has no advice to give, then no additional reward is given.

The simulated user designed for this experiment is replicating two of the human interaction characteristics described in section 6.1.1, accuracy and availability. Both characteristics are represented as percentages. When accuracy is at 100%, the simulated user provides accurate advice, and as accuracy decreases the simulated user has an increasing chance of providing incorrect advice. Similarly, when availability is at 100% the simulated user can assess the agent at every time step, as availability decreases the user’s opportunities to provide advice decreases also.

A set of simulated users are created, each with the same knowledge model for the environment but each with incrementing levels of advice accuracy and availability ranging from 20% to 100%. Each characteristic incremented by 20%. The time required to create such users is negligible compared to the time required to seek ethics approval and perform a similar number of experiments with real users. 100 experiments are performed for each of the users, for a total of 2500, trials a monumental task if real humans were to be used. The average number of steps the agent takes to complete the mountain climbing task are collected during learning. The agent is given a maximum of 1000 steps each episode to achieve its goal, and the agent is given 100 episodes to learn the task. A full description of the Mountain Car environment and the agent parameters used is provided in Chapter 4. The experiments produce results showing insights into how the accuracy and availability of the simulated user alter the performance of the ARL agent.

Figure 6.2 is an example of the evaluation that can be performed using this methodology. In this diagram, accuracy and availability of advice is plotted, with the opposite characteristic

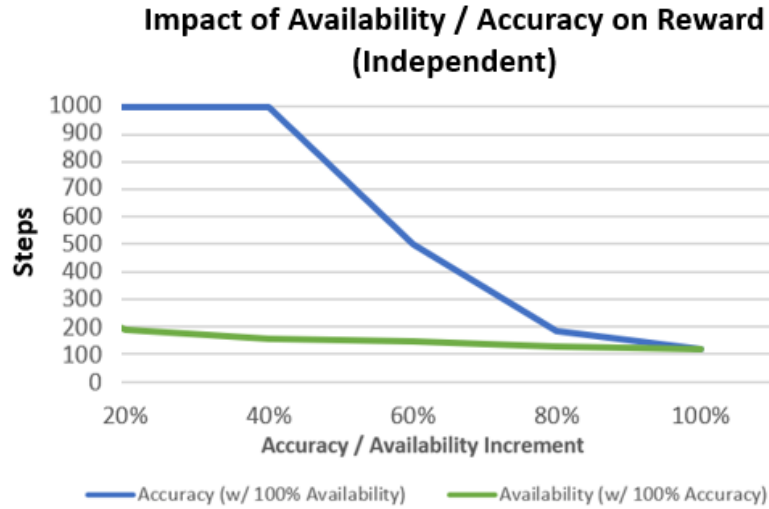


Figure 6.2: Evaluation of an Assisted Reinforcement Learning agent using a simulated user using varying levels of availability and accuracy (Independent). Each characteristic is incremented by 20% for each experiment.

set to maximum. This graph shows how the agent’s online performance is affected by the accuracy of the advice given by the simulated user. In this case, agent performance quickly degrades as the accuracy of the advice worsens. The largest impact to performance occurs when accuracy falls to 40%, at this point the user is giving incorrect advice more than half of the time. The figure also shows that advice availability, when 100% accurate, has a very large impact on agent performance, but the rate of change is diminishing as availability increases.

The contour graph shown in Figure 6.3 is a method for presenting the relationship between two characteristics of human-sourced information and their effect on the performance of the agent. In Figure 6.3 the average steps of the agent are plotted, showing the change in performance can be observed as the simulated users accuracy and availability are altered. From this figure, multiple observations can be made regarding how accuracy and availability of the user impact the average performance of the agent. For example, it can be observed that a small amount of advice can have a large impact on the agent’s performance; however, there are diminishing returns as frequency increases. The figure also shows that inaccuracy of advice has less of an impact as the frequency of advice is increased.

These methods of evaluation allow for greater insight into the performance of ARL agents when advised by different users. The use of different simulated users can show how an ARL agent can perform under various conditions. The application of simulated users for this

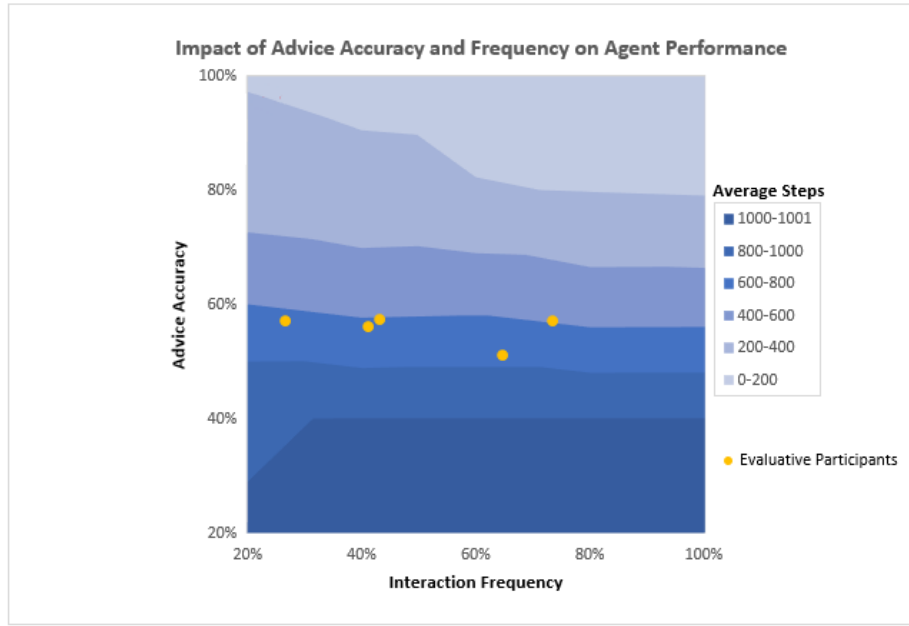


Figure 6.3: Evaluation of an Assisted Reinforcement Learning agent using a simulated user using varying levels of availability and accuracy (Dependent). Each characteristic is incremented by 20% for each experiment.

method of evaluation can identify the strengths and weaknesses of an ARL agent and the user providing advice, while also performing the experiments cheaper and faster than actual humans.

6.5 Conclusion

This chapter introduced the use of simulated users to Interactive Reinforcement Learning. Different types of simulated users were discussed, some characteristics of human interactions were identified and defined, and evaluation principles for simulated users were presented.

An illustrative experiment was performed to show how simulated users can be employed to mitigate the challenges introduced into testing when using human advice, and how the results gained from simulated users can provide indicative observations on real human performance.

In the next chapter, the concept of persistent advice is presented. Persistence is the practice of advice reuse and retention in Interactive Reinforcement Learning, in an attempt to improve agent learning and reduce demand on human advisors. Simulated users are employed, modelled from the results from the human trials performed in Chapter 5, to evaluate the persistent agents.

Chapter 7

Persistent Advice

Interactive Reinforcement Learning allows humans to guide or evaluate an agent’s behaviour. The assistance provided by the human reinforces the behaviour the agent is learning and shapes the exploration policy, resulting in a reduced search space. Current Interactive Reinforcement Learning techniques discard the advice sourced from the human shortly after it has been used (Knox & Stone, 2009; Griffith et al., 2013), increasing the reliance on the advisor to repeatedly provide the same advice to maximise the agents use of it.

This chapter introduces persistence to Interactive Reinforcement Learning, a method for information retention and reuse. Persistent agents attempt to maximise the value extracted from the advice by replaying interactions that occurred in the past, rather than relying on the advisor to repeat an interaction. Through experimentation, this chapter shows that agents that retain advice require less interactions with an advisor than non-persistent counterparts to achieve similar or improved performance, thus reducing the burden on the advisor to provide advice on an ongoing basis.

7.1 Persistence

Current applications of Interactive Reinforcement Learning allow humans to provide advice to an agent to reinforce or promote behaviour during learning (Knox & Stone, 2010; Griffith et al., 2013). Leading applications are limited to state-specific interactions, such as an evaluation of the agents last action choice or recommendation on the next action to take (Knox & Stone, 2008) . It has been shown that such interactions can have a substantial positive impact on the agents performance (Griffith et al., 2013). However, in current implementations, the advice received is discarded after the agents initial use. This can be detrimental to the agent as the complete value of an interaction is not incorporated in a single

use, especially if the agent does not have a high learning rate. Therefore, it is proposed that agents retain and replay interactions to maximise the utility of provided advice.

Retention and reuse of advice is an intersection of the inverse and Interactive Reinforcement Learning fields. Implementations of inverse Reinforcement Learning demonstrate algorithms for constructing reward functions from observation, while Interactive Reinforcement Learning allows humans to supplement the reward function already in use. Both fields have the same goal, to tweak the reward function such that the optimal policy reflects that of the demonstrator. Interactive Reinforcement Learning has since begun moving away from reward shaping, opting for policy shaping, which demonstrates greater utility with lower interactions. The method for retention and reuse proposed here combines the concept of modelling demonstrations from inverse Reinforcement Learning, with the evaluative and informative interaction methodology from Interactive Reinforcement Learning. This combination, resulting in retained advice, allows an agent to maximise the utility of each interaction.

A persistent agent keeps a record of each interaction and the conditions in which it occurred. When the conditions are met in the future, the interaction is replayed. This results in improved utility of the advice and consequently, improved performance of the agent. Additionally, less interactions with the human are required, as there is no need for advice to be repeatedly provided for each state. The human can be confident that the agent will learn from the evaluation or recommendation that it was provided.

However, a naive implementation of persistence can introduce flaws into the reward shaping process. These flaws, if unaddressed, may cause the agent to never learn an optimal policy. Prior work on reward shaping (Randløv & Alstrøm, 1998) has shown that while reward shaping can accelerate learning, it can also result in the optimal policy under the shaping reward differing from that which would be optimal without shaping. Ng *et al.* (1999) demonstrated that this issue can be avoided by using a potential-based approach to constructing the shaping reward signal. This guarantees that the rewards obtained along any path from a state back to itself are zero-sum so that the agent won't find a loop in the environment that will provide infinitely accumulating rewards without termination (Devlin & Kudenko, 2011), for non persistent interactive reinforce shaping agents the reward given as part of evaluation is temporary as the human does not have to provide the supplemental

reward upon revisiting the state. Assuming that the human will eventually stop providing advice, the reward signal will become zero-sum. A persistent agent records the supplemental reward from the initial interaction and reapplies it the next time the state is visited, resulting in a non-zero-sum path. There has been work into potential-based assisted Reinforcement Learning (Harutyunyan, Devlin, et al., 2015), however, the interactive nature of this field makes its implementation exceedingly difficult, especially when the environment dynamic or reward function is unknown.

For Interactive Reinforcement Learning agents that use informative advice, i.e., recommendations on which action to perform next, a straightforward implementation of persistence will work if the advice is correct. However, as demonstrated in previous chapters, human advice is rarely 100% accurate. Inaccuracy can result from negligence, misunderstanding, latency, maliciousness, and noise introduced when interpreting advice. Furthermore, if the agent always performs the recommended action, then it is not given the opportunity to explore and discover the optimal action. An agent that retains and reuses inaccurate advice will not learn an optimal policy. Therefore, it is important that the agent be able to discard or ignore retained knowledge.

These two issues with persistence, non-potential reward shaping and incorrect advice, result in persistent agents being unable to learn the optimal policy. The issue of inaccurate advice with persistence has two possible solutions, either identify the incorrect advice and discard it, or discard all advice after a period regardless of its accuracy. To know the accuracy of a piece of advice a full solution to the problem must be known, and if this is achievable then a Reinforcement Learning agent is not needed. Instead, a policy of discarding or ignoring advice after a period allows a persistent agent to function with potentially inaccurate advice, while still maximising the utility of each interaction. This method also solves the issue of non-potential evaluative advice, as the frequency of the supplemental reward is reduced over time until zero. Once the supplemental reward is reduced to zero, the cumulative shaping reward function becomes zero-sum once again.

To solve the issue of incorrect advice in persistent Interactive Reinforcement Learning, a method for discarding or ignoring advice after a period of time is needed. Probabilistic policy reuse is a technique that aims to improve Reinforcement Learning agents that use

guidance (Fernández & Veloso, 2006). PPR relies on using probabilistic bias to determine which exploration policy to use when multiple options are available, the goal of which is to balance random exploration, the use of a guidance policy, and the use of the current policy.

For the persistent agent scenario, there are three action selection options available, random exploration, the use of retained advice from the human, or the best action currently known. PPR assigns each of the three options a probability and priority of selection (Fernández & Veloso, 2006). Over time, the probability of using guidance or retained information decreases, and trust in the agents own policy increases. Using PPR, the guidance provided by the human is used for more than a single time step, with a decreasing probability over time, until the value of the advice is captured by the agents own policy. Once encapsulated by the agent, self-guided exploration and exploitation of the environment continues.

7.2 Experiment

The remainder of this chapter demonstrates the use of persistent advice with probabilistic policy reuse, and the impact its use has on agent performance and human reliance. The experiments have been designed to test several levels of human advice accuracy and availability, with and without retention of received advice.

The Mountain Car environment is used in the following experiments. The environment is a common benchmark problem in Reinforcement Learning due to it having sufficient complexity to effectively test agents, being simple enough for human observers to intuitively calculate the correct policy, and its long history of use as an RL test bed. Additionally, the mountain car environment was used in the human trial introduced in Chapter 5, the results of which inform parameters used in the following experiments. For more information on the mountain car environment, see Chapter 4.

Five agents have been designed for the following experiments, these are listed below.

- (i) **Benchmark Q-Learning Agent** A Q-learning agent used for capturing a baseline for performance on the Mountain Car environment. This agent is unassisted, receiving no guidance or evaluation from the human. For the agent to represent the continuous two-dimensional state space of the environment it has been discretized into twenty bins for each state feature, creating a total of four-hundred states, each with three actions.

The expected-reward values have been initialized to zero, an optimistic value for the environment. The agent is given a learning rate of 0.25, a discounting of 0.9, and used an e-greedy action selection strategy with an epsilon of 0.05. This agent uses e-greedy action selection, taking a random action 10% of the time, and the best known action the remainder.

- (ii) **Non-Persistent Evaluative Advice Q-Learning Agent** This agent is assisted by a user. The user may provide an additional reward each time step to evaluate the agent's last choice of action. For this non-persistent agent, the supplemental reward is used in the current steps learning update, and then discarded. The action selection policy and learning parameters are the same as a benchmark agent.

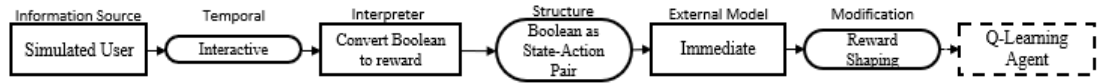


Figure 7.1: Definition of the non-persistent evaluative assisted experimental agent using the Assisted Reinforcement Learning framework.

- (iii) **Persistent Evaluative Advice Q-Learning Agent** This agent is assisted by a user. The user may provide an additional reward each time step to evaluate the agent's last choice of action. For this persistent agent, the evaluation provided is retained, and upon performing the same state-action pair in the future, the evaluation may be automatically provided to the agent, with a probability defined by the PPR action selection policy. The learning parameters are the same as a benchmark agent.

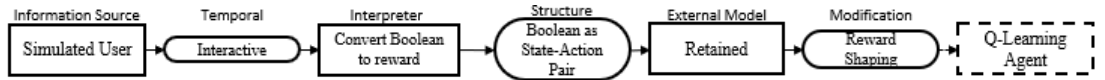


Figure 7.2: Definition of the persistent evaluative assisted experimental agent using the Assisted Reinforcement Learning framework.

- (iv) **Non-Persistent Informative Advice Q-Learning Agent** - This agent is assisted by a user. The user may recommend an action for the agent to perform for the current time step. When the agent is recommended an action, that action is taken on that time

step, and then the advice is discarded. This non-persistent agent, when visiting the same state again in the future, will not recall the recommended action and will perform e-greedy action selection. The action selection policy and learning parameters are the same as a benchmark agent.

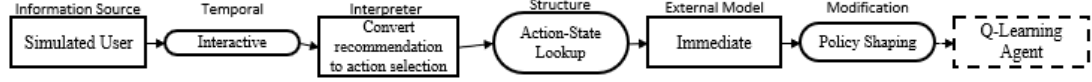


Figure 7.3: Definition of the non-persistent informative assisted experimental agent using the Assisted Reinforcement Learning framework.

- (v) **Persistent Informative Advice Q-Learning Agent** - This agent is assisted by a user. The user may recommend an action each time step for the agent to perform. If recommended an action from the user, the agent will take it on that time step and retain the recommendation for use when it visits the same state in the future. When the agent visits a state in which it was previously recommended an action, it will take that action with the probability defined by the PPR action selection policy. The learning parameters are the same as a benchmark agent.

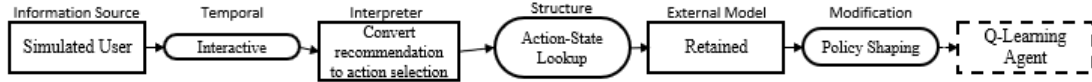


Figure 7.4: Definition of the persistent informative assisted experimental agent using the Assisted Reinforcement Learning framework.

The agents adopting a persistent model are employing probabilistic policy reuse for action selection. As depicted in Figure 7.5, the probabilistic policy reuse action selection begins with an 80% chance of reusing advice provided to the agent in the past. The probability of reusing advice will decrease by 5% each episode. For the remaining 20% of the time, or if no advice has been provided for the current state, an e-greedy action selection policy is used. The e-greedy action selection policy will perform a random action 10% of the time and perform the best action it knows of the remaining 90% of the time.

For each agent, one hundred experiments are run. At the beginning of each experiment,

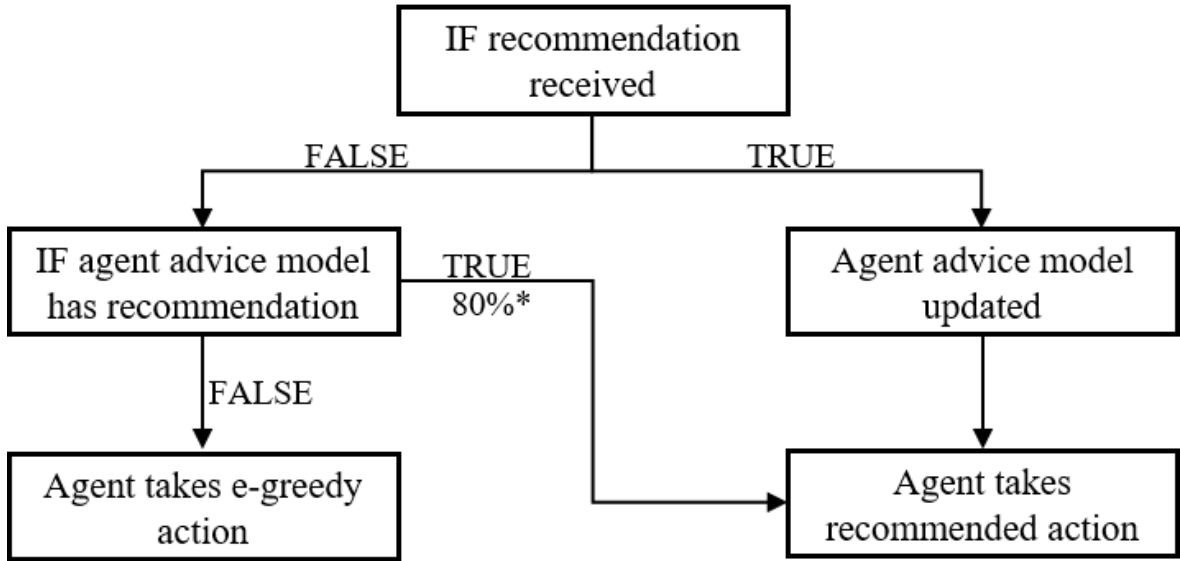


Figure 7.5: Probabilistic Policy Reuse (PPR) for an Interactive Reinforcement Learning agent using informative advice. If the user recommends an action on the current time step then the agent’s advice model updates and the action is performed. If the user does not provide advice on the current time step, then the agent will follow previously obtained advice 80% of the time (*decays over time), and uses its default exploration policy the remaining time.

the environment, the agent, and the agent’s model of provided advice are reset. Each experiment runs for five hundred episodes, each episode with a maximum of one thousand steps before it terminates. The number of steps performed, interactions performed, and reward received are recorded. An interaction is recorded if the user provides advice to the agent, not when the agent uses advice it has stored from a previous interaction. All data shown in the tables and figures following is collected from online agents, agents that are using action selection policies that include a degree of exploration.

Six different simulated users have been created. Each simulated user will either provide evaluative or informative advice. Evaluative advice-giving users, always paired with evaluative interactive agents, provide either a positive or negative supplemental reward corresponding to the agent’s choice in action on the last time step. Informative advice-giving users, always paired with informative agents, provide a recommended action for the agent to perform on the current time step. Simulated users that are advising a persistent agent will not provide advice for a state, or state-action for evaluative agents, that it has previously advised on, as it is assumed that if the agent is retaining information it should not need

repeated advice for the same conditions. This does not apply to non-persistent agents.

Additionally, each simulated user will have either optimistic, realistic, or pessimistic values for advice accuracy and availability. Accuracy is a measure of how correct the advice is provided by the user. Accuracy of an interaction is altered by, with specified probability, replacing the recommended action with an action that is not optimal for the current state. Availability is a measure of how frequently the user provides advice to the simulated user. The availability of the simulated user is altered by specifying a probability that the user will interact with the agent on any given time step.

ACCURACY/AVAILABILITY LEVELS OF HUMAN ADVISORS FOR MOUNTAIN CAR		
ADVICE/AGENT TYPE	ACCURACY	AVAILABILITY
EVALUATIVE	48.470%	26.860%
INFORMATIVE	94.870%	47.316%

Table 7.1: Observed accuracy and availability for human advisors in the Mountain Car environment. See Chapter 5.

Optimistic simulated users have 100% accurate advice, and will provided advice on every time step that the agent does not have retained knowledge of. Realistic simulated users have accuracy and availability modelled from the human trial performed in Chapter 5, as summarised in Table 7.1. The recorded accuracy and availability of human advice gives differs depending on the type of advice being provided, evaluative or informative. Previous chapters and papers have compared evaluative and informative advice/agents, and as such is not in scope of this chapter. Instead, this chapter compares the non-persistent and persistent agents of each type against each other. Lastly, pessimistic simulated users are given accuracy and availability values half that of the realistic users. Table 7.2 shows the accuracy and availability values for each of the six simulated users (3 evaluative users, 3 informative users).

Table 7.3 lists all the agent/simulated user combinations tested in this chapter. There are a total of thirteen agents, six persistent agents, six non-persistent agents, and an unassisted Q-Learning agent used for benchmarking. Included next to each agent/user combination is a short name. This short name is used in the tables in the next section, as the full name is too long to include in each figures legend.

SIMULATED USERS		
NAME	ACCURACY	AVAILABILITY
EVALUATIVE OPTIMISTIC	100%	100%
EVALUATIVE REALISTIC	48.470%	26.860%
EVALUATIVE PESSIMISTIC	24.235%	13.43%
INFORMATIVE OPTIMISTIC	100%	100%
INFORMATIVE REALISTIC	94.870%	47.316%
INFORMATIVE PESSIMISTIC	47.435%	23.658%

Table 7.2: The six simulated users designed for the chapters experiments. There are three of each advice type, evaluative and informative, to be used with the matching agent type. The realistic simulated users are set with values observed from the human trials performed in Chapter 5. The pessimistic values are half that of the realistic values. The advice delivery methods, evaluative and informative, are not intended to be compared against each other in this chapter, rather, they are to be compared against their persistent counterparts. This allows simulated users to be individually set for each each advice delivery style that more accurately represent the type of user that would be advising each type of agent.

AGENT/USER COMBINATIONS FOR PERSISTENT AGENT TESTING		
SHORT NAME	AGENT	SIMULATED USER
UQL	UNASSISTED Q-LEARNING AGENT (BENCHMARK)	NONE
NPE-O	NON-PERSISTENT EVALUATIVE ADVICE AGENT	EVAL. OPTIMISTIC
NPE-R	NON-PERSISTENT EVALUATIVE ADVICE AGENT	EVAL. REALISTIC
NPE-P	NON-PERSISTENT EVALUATIVE ADVICE AGENT	EVAL. PESSIMISTIC
NPI-O	NON-PERSISTENT INFORMATIVE ADVICE AGENT	INFO. OPTIMISTIC
NPI-R	NON-PERSISTENT INFORMATIVE ADVICE AGENT	INFO. REALISTIC
NPI-P	NON-PERSISTENT INFORMATIVE ADVICE AGENT	INFO. PESSIMISTIC
PE-O	PERSISTENT EVALUATIVE ADVICE AGENT	EVAL. OPTIMISTIC
PE-R	PERSISTENT EVALUATIVE ADVICE AGENT	EVAL. REALISTIC
PE-P	PERSISTENT EVALUATIVE ADVICE AGENT	EVAL. PESSIMISTIC
PI-O	PERSISTENT INFORMATIVE ADVICE AGENT	INFO. OPTIMISTIC
PI-R	PERSISTENT INFORMATIVE ADVICE AGENT	INFO. REALISTIC
PI-P	PERSISTENT INFORMATIVE ADVICE AGENT	INFO. PESSIMISTIC

Table 7.3: Agent/User combinations for persistent agent testing, including short names for reference.

7.3 Results

7.3.1 Probabilistic Policy Reuse

The first experiment performed compares the use of PPR as an action selection method, compared to always using advice when available. Previously, a claim was made that the use of persistence in Reinforcement Learning introduces a critical flaw. Specifically, if provided advice is retained and reused, and that advice is incorrect, then the agent will not be able to learn a solution to the current problem. Figure 7.6 shows the results of three agents, an unassisted Q-Learning agent for benchmarking, and two persistent Interactive Reinforcement Learning agents using informative advice. These two agents are identical except that one is using PPR for action selection (*PI-R PPR*), while the other will always take a recommended action if one exists for the current state (*PI-R No PPR*). Both interactive agents are assisted by a simulated user created with realistic values of accuracy and availability.

Figure 7.6 shows that both assisted agents immediately outperforms an unassisted agent (*UQL*, Blue). Both agents are retaining the recommended actions from the user, and cannot differentiate between correct and incorrect advice. The No-PPR agent (Red) will always

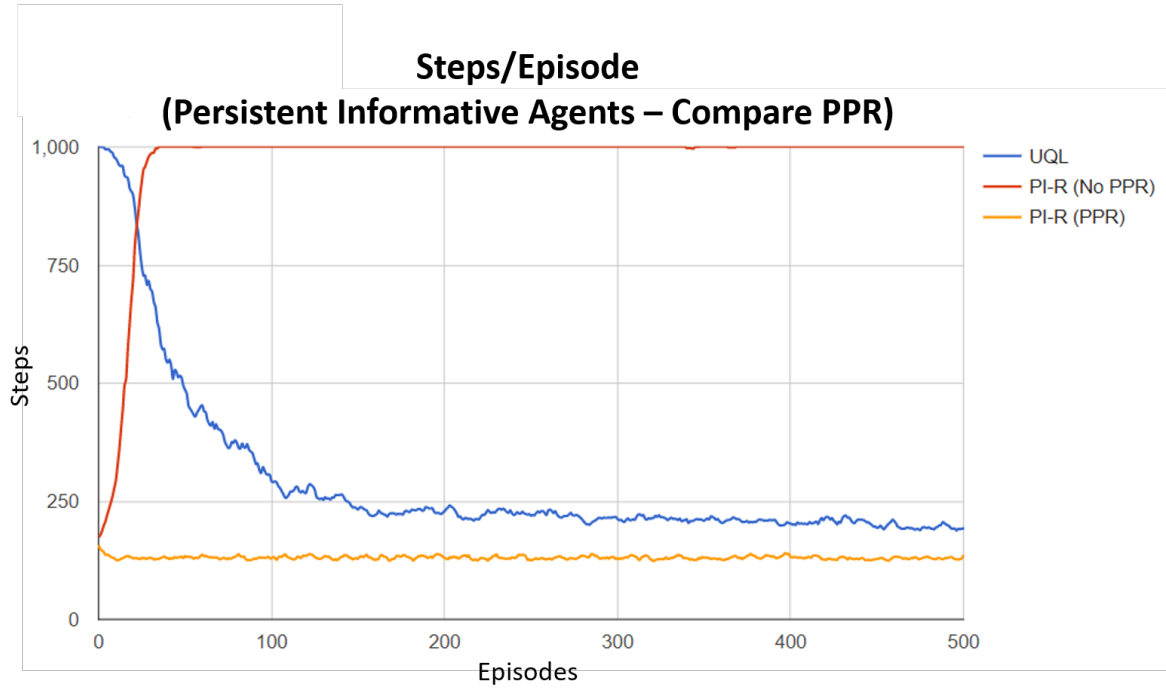


Figure 7.6: Probabilistic Policy Reuse (PPR) versus direct-use action selection for Interactive Reinforcement Learning using retained informative advice for the Mountain Car environment. Both assisted agents are using simulated users using realistic values for accuracy and availability, and are both retaining advice provided to them.

take the recommended action for the current state, if available. This works well for the first few episodes, as small amounts of correct advice can have a large positive impact on agent performance and small amounts of incorrect can be ignored because of the momentum the agent builds in the Mountain Car environment. However, as the amount of incorrect recommended actions retained increases, the effect on the agent’s performance increases. Eventually, the impact of taking the wrong action will have such an effect that the agent cannot build the required momentum to solve the Mountain Car environment.

The agent using probabilistic policy reuse (PPR) continues to outperform both the unassisted agent (UQL), and the other assisted agent (*PI-R No PPR*). The PPR agent will initially take the users advice in high regard, taking recommended actions 80% of the time. Over time, the agent pays less attention to the retained advice of the user, and more to its own learnt policy. This allows the agent to disregard incorrect advice, as its own value estimations will show the correct action to take, while correct advice will accelerate the discovery of the true

value estimation of the correct action in advised states.

7.3.2 Persistent Advice on the Mountain Car Domain

Figure 7.7 shows the performance over time for both non-persistent evaluative and informative agents, at varying levels of user accuracy and availability. The purpose of this figure is not to compare the two advice delivery styles against each other, but to compare them against their persistent counterparts that follow. As evaluative advice is evaluating actions that have already been taken, there is a short delay between the action being taken and the application of the advice to the agent. This delay causes a latency in the effect of the advice on the agent’s learnt policy. Figure 7.7 shows this delay for the evaluative agent, where most of the advice is given in the first few episodes, but it takes around twenty episodes before the agent has fully utilised the advice and converges to an optimal path. The agent using informative advice on the other hand does not suffer from this delay. This agent is receiving recommendations on which action to take next, and if a recommendation is provided, then the action is taken.

Table 7.4 shows the number of interactions that occurred, and the percentage of interactions compared to the number of steps performed, for all agent/simulated user combinations. These values align with the percentage of availability of each of the simulated users. This is expected, as the agents are not retaining any advice received from the user, so users are

RECORDED INTERACTION PERCENTAGE FOR EACH USER/NON-PERSISTENT AGENT COMBINATION	
AGENT	#INTERACTION (% INTERACTIONS / TOTAL STEPS)
UQL	0.00%
NON-PERSISTENT EVALUATIVE AGENT w/ OPTIMISTIC USER	58355 (100.00%)
NON-PERSISTENT EVALUATIVE AGENT w/ REALISTIC USER	486503 (26.86%)
NON-PERSISTENT EVALUATIVE AGENT w/ PESSIMISTIC USER	500499 (13.43%)
NON-PERSISTENT INFORMATIVE AGENT w/ OPTIMISTIC USER	54083 (100.00%)
NON-PERSISTENT INFORMATIVE AGENT w/ REALISTIC USER	134590 (47.31%)
NON-PERSISTENT INFORMATIVE AGENT w/ PESSIMISTIC USER	193170 (23.65%)

Table 7.4: Average number of interactions performed per experiment, and the percentage of interactions compared to the steps taken, for each non-persistent agent/user combination in the Mountain Car environment.

required to continually give the same advice if the agent is to maximise its utility. An observation can be made that as the availability of the simulated user decreases, the number of interactions increases. Availability is the likelihood that the user will interact with the agent on any given time step, so it would be assumed that the higher the availability then the higher the number of interactions. However, this is not what is observed.

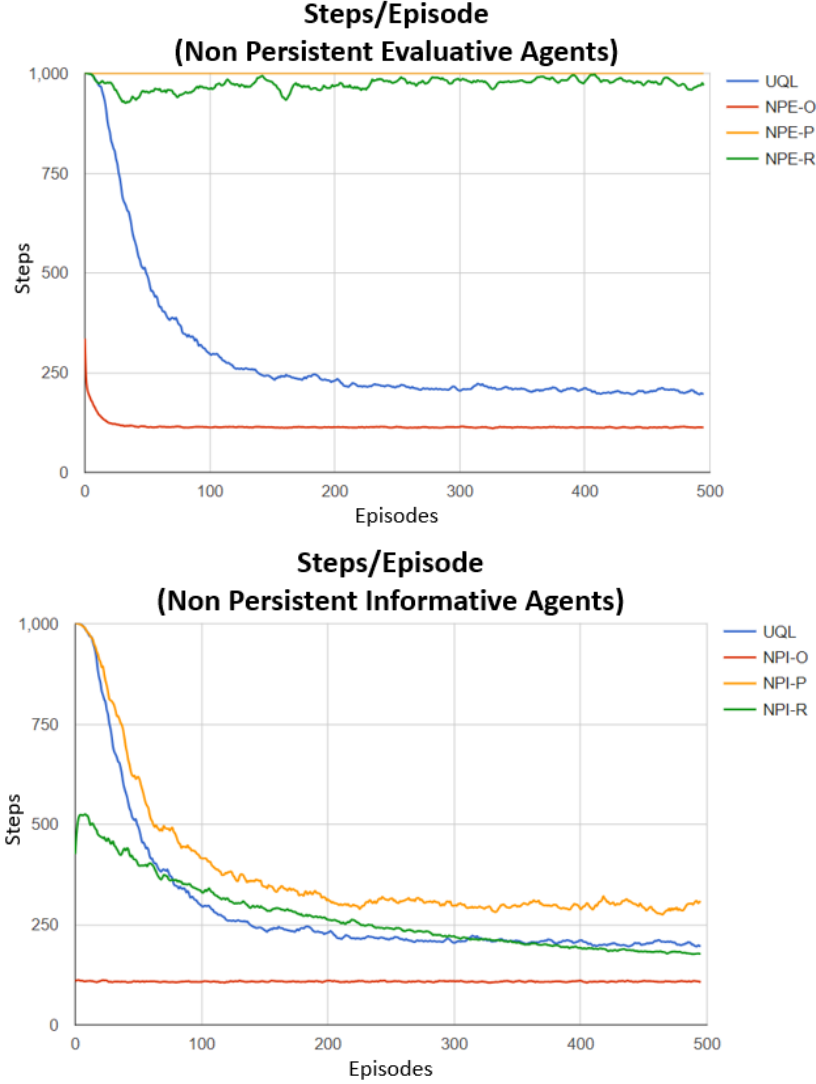


Figure 7.7: Performance of non-persistent evaluative and informative agents on the Mountain Car environment, using different simulated users with varying levels of advice accuracy and availability. The UQL agent is an unassisted q-learning agent for benchmarking. Agents with the suffix -O are using optimistically initialised simulated users, the suffix -R denotes realistic initialised simulated users, and the suffix -P denotes pessimistically initialised simulated users. Figure 7.2 shows the levels of accuracy and availability for each simulated user.

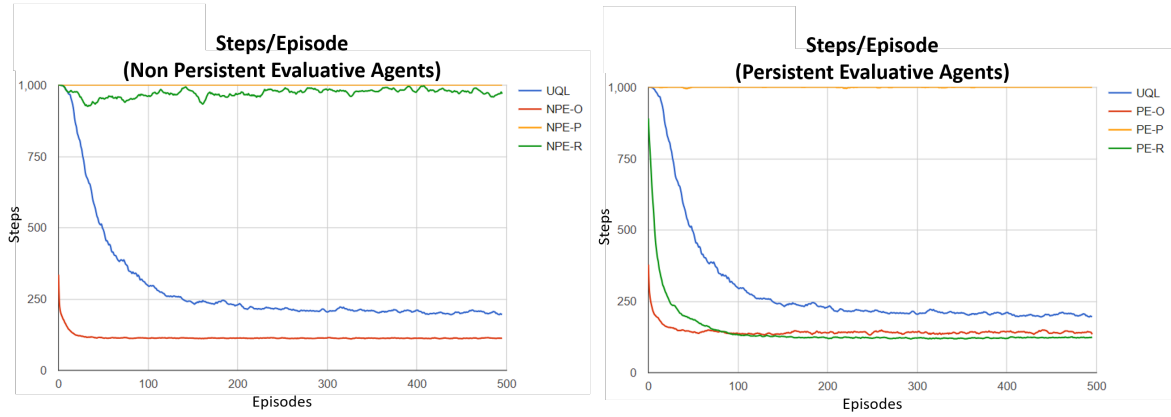


Figure 7.8: Steps per episode for non-persistent (NPE-*) and persistent (PE-*) agents using evaluative advice. The agents are assisted by three different simulated users, initialised with either **O**ptimistic, **R**ealistic, or **P**essimistic values for accuracy and availability. The figure shows that evaluative persistent agents learn in fewer steps than non-persistent evaluative agents.

The simulated users that have been created have a direct correlation between the availability of the user and its accuracy. Simulated users that are highly accurate allow the agent to learn the optimal policy faster, which result in the agent taking fewer steps, and the simulated user having fewer opportunities to provide advice. Simulated users with lower accuracy, such as the pessimistic users, cause the agent to take longer to learn the policy, resulting in the agent taking more steps, and allowing the simulated user more opportunities to provide advice. This is what creates the inverse correlation between the users advice availability and the number of interactions recorded.

Figure 7.8 shows evaluative agents, both non-persistent (PE^*) and persistent (PE^*), using advice from three different users. The persistent agent is using probabilistic policy reuse to manage the trade-off between the advice received from the user, its own learnt policy, and its exploration strategy. The persistent agent is limited to only receiving one interaction from the user per state-action pair. If the agent has already received some advice for the state-action pair in question, then the user is not given the opportunity to provide additional advice. The agent instead relies on the stored advice from the first interaction regardless of its accuracy. Both agents will always utilise advice received directly from the user on the current time step. However, the persistent agent follows PPR, which allows the agent to diminish the probability of using the advice for a state-action pair over time. This results

in the persistent agent receiving one interaction per state-action pair, maximising the utility that interaction, then eventually only relying on its own policy.

The agents being assisted by optimistically initialised users perform almost the same. The optimistically-assisted persistent agent (*PE-O*) takes slightly longer to learn than the non-persistent counterpart (*NPE-O*), because the advice it receives is only listened to 80% (diminishing over time) after the initial interaction with the user, compared to the non-persistent agent whose user will continually interact with the agent and the agent will always follow the advice.

The agents being assisted by realistically initialised users differ greatly in performance. The non-persistent evaluative agent (*NPE-R*) using a realistically initialised user, while able to solve the Mountain Car problem in less steps than the 1000 cut-off limit, was not able to find the optimal solution. However, the persistent evaluative agent (*PE-R*) was not only able to solve the problem, but also learn the solution faster than the benchmark unassisted agent (*UQL*). Just like the *NPE-O* and *PE-O* agents, the difference in performance is not only due to the persistent agent remembering the advice, but also because the persistent agent can eventually disregard incorrect advice as the likelihood that the PPR algorithm will choose to take the recommended action diminishes over time, while the agent's value estimation of the recommended action remains the same. What is particularly notable from these results is that the persistent agent (*PE-R*), being advised by the simulated user models from real human trials (Chapter 5), still outperformed unassisted Q-learning despite more than half of all interactions giving the incorrect advice.

Regardless of whether the agent is persistent or not, neither agent (*NPE-P*, *PE-P*) that was advised by a pessimistically-initialised user was able to solve the Mountain Car problem. This is likely due to the accuracy of the pessimistic user being less than 25%.

Figure 7.9 shows the performance of informative agents, both non-persistent (*NPI-**) and persistent (*PI-**), using advice from three users with different levels of advice accuracy and availability. These agents can receive informative advice from a user. The advice that they receive is an action recommendation, informing them of which action to take in the current state. When either agents, persistent or non-persistent, receive an action recommendation directly from the user on the current time step that action will be taken by the agent. The

persistent agent will remember that action for the state it was received in, and use the PPR algorithm to continue to take that action in the future. Once the persistent agent has received an action recommendation from the user for a particular state, the user will not interact with the agent for that state in the future.

Figure 7.9 shows that the informative agents ($NPI-O$, $PI-O$), regardless of persistence, learnt the solution in the same amount of time when being advised by an optimistically initialised user. This is not surprising as the agent is receiving 100% accurate advice for every time-step, making this essentially a supervised learning task at great effort of the user. A difference in the time required to find a solution can be seen in the agents that are assisted by a realistically initialised agent ($NPI-R$, $PI-R$). While the non-persistent agent ($NPI-R$) agent does learn faster than an unassisted agent (UQL), the persistent agent learns the solution almost immediately, much like the optimistically-assisted persistent agent ($PI-O$).

This difference in learning speed is likely due to the agent retaining and reusing advice. The $NPI-R$) and $PI-R$) agents are being assisted by a simulated user with realistic values for accuracy and availability. The realistic simulated user has a 48% change of interacting with an agent on any particular time step. The non-persistent agent does not retain advice from the user, so it will always have a 48% change of receiving advice for any particular state. However, the persistent agent will retain and reuse advice with an 80% (diminishing over time from PPR) probability for any state that it has received advice on in the past. As long as the retained advice is sufficiently accurate, the persistent agent will learn faster because it utilises that human sourced advice more often.

The last two agents are assisted by a pessimistically initialised user. The non-persistent agent outperformed the persistent agent in this experiment. This is due to the same principle as the realistically-assisted informative agents. The pessimistically-assisted agent performed the recommended action more often than the non-persistent agent. Both agents have a 23.6% chance of receiving advice from the pessimistic user, however, the persistent agent retains and reuses this advice, and will take the recommended action 80% of the time for states it has been advised on. This results in the $PI-R$ agent taking the advised incorrect action far more often than the $NPI-R$ agent.

Table 7.5 shows the number and percentage of interactions that occurred on on average

for each agent/user combination. The number of interactions is a measure of the total number of interactions that occurred, on average, each experiment. This measurement is not suitable to compare agents, as agents that benefit from advice will take less steps, giving the users less opportunities to provide advice, despite perhaps requiring more attention from the user per episode. What this measurement does indicate is the total number of interactions required to achieve and maintain the observed performance.

The second measurement, shown in parenthesis, is the percentage total steps taken that the agent was directly advised on. For non-persistent agents, this interaction percentage is equal to the advising users availability. This measurement is suitable for comparing agents against each other, as it is a function of the interaction requirements, rather than a direct measurement of the number of interactions.

A clear observation from Table 7.5 is that persistent agents require substantially less interactions than non-persistent agents. All persistent agents measured less than 1% of steps had a direct interaction with a user. Assuming a direct correlation between the number of interactions and the time required to perform said interactions, the use of persistence offers a large time reduction for assisting users. This significant drop in required interactions, coupled with the previous observation of large performance gains shown by the majority of persistent

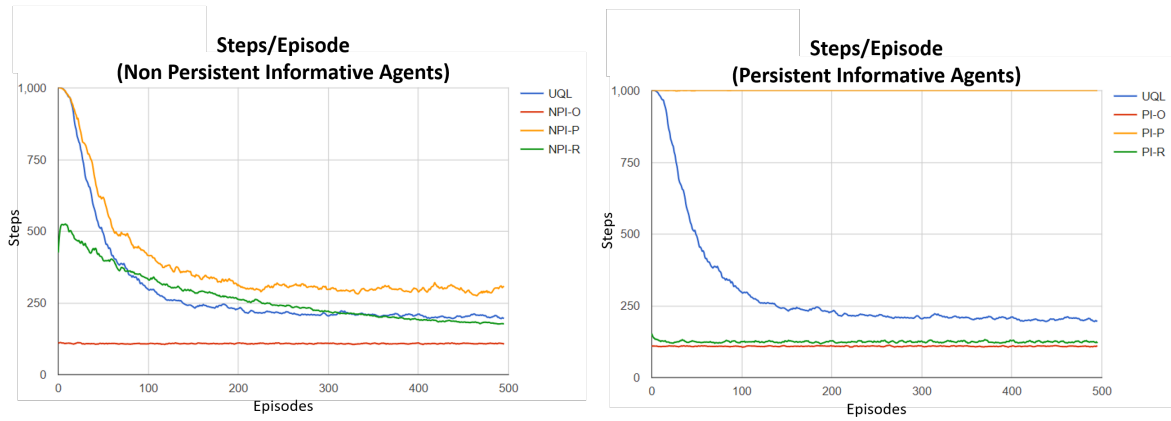


Figure 7.9: Steps per episode for non-persistent (NPI-*) and persistent (PI-*) agents using informative advice. The agents are assisted by three different simulated users, initialised with either **O**ptimistic, **R**ealistic, or **P**essimistic values for accuracy and availability. The figure shows that informative persistent agents learn in fewer steps than non-persistent informative agents when assisted by sufficiently accurate users.

RECORDED INTERACTION PERCENTAGE FOR EACH USER/AGENT COMBINATION		
AGENT	INTERACTION (% INTERACTIONS / TOTAL STEPS)	
	NON-PERSISTENT	PERSISTENT
UNASSISTED Q-LEARNING AGENT	0.00%	
EVAL. AGENT / OPTIMISTIC USER	57,855 (100%)	668 (0.91%)
EVAL. AGENT / REALISTIC USER	130,540 (26.86%)	117 (0.01%)
EVAL. AGENT / PESSIMISTIC USER	67,156 (13.43%)	47 (<0.01%)
INFOR. AGENT / OPTIMISTIC USER	53,583 (100%)	253 (0.46%)
INFOR. AGENT / REALISTIC USER	63,431 (47.31%)	255 (0.01%)
INFO. AGENT / PESSIMISTIC USER	45,568 (23.65%)	63 (0.38%)

Table 7.5: Average number of interactions performed per experiment, and the percentage of interactions compared to the steps taken, for each non-persistent and persistent agent/user combination in the Mountain Car environment.

agents, makes a compelling case for the retention and reuse of advice, assuming a suitable level of accuracy of that advice.

Table 7.5 shows the number of interactions, and the percentage of interactions compared to total steps, for each agent/user combinations, both persistent and non-persistent. These results show that the number of interactions required by the user to achieve each agent’s recorded performance is significantly reduced when advice is retained.

7.4 Conclusion

This chapter introduced the concept of *persistence* to Interactive Reinforcement Learning. Current Interactive Reinforcement Learning agents do not retain the advice provided by assisting users. This may be due to the effect that incorrect advice has on an agent’s performance. To mitigate the risk that inaccurate information has on agent learning, probabilistic policy reuse was employed to manage the trade-off between following the advice policy, the learnt policy, and an exploration policy. Figure 7.6 showed that PPR can reduce the impact that inaccurate advice has on agent learning in the Mountain Car environment, up to a certain level of inaccuracy.

Further experiments performed in this chapter found that both evaluative and informative Interactive Reinforcement Learning agents learn faster when retaining the information

provided by an advising user, when the advising user’s accuracy is sufficient. Additionally, persistent agents were shown to require significantly fewer interactions than non-persistent agents, while achieving the same or better learning speeds when advice accuracy was sufficient. Further exploration into the mitigation of the risks that inaccurate information introduces, and identification of the levels of inaccuracy that PPR can mitigate, are both areas for future research.

In the next chapter, rule-based advice is introduced as a method for advice delivery and retention. Rule-based advice potentially offers the same performance benefits as persistent advice, while requiring fewer interactions and a smaller policy for retained advice.

Chapter 8

Rule-Based Interactive Reinforcement Learning

Current Interactive Reinforcement Learning agents allow users to evaluate or recommend actions based only on the current state of the environment (Griffith et al., 2013; Knox & Stone, 2010). This constraint restricts the user to providing advice relevant to the current state and no other, even when such advice may be applicable to multiple states. Restricting the time and utility of interactions only serves to increase the demand on the user’s time, and to withhold potentially useful information from the agent.

Allowing users to provide information in the form of rules, rather than per-state action recommendations, increases the information per interaction, and does not limit the information to the current state. By not constraining the advice to the current state, users can give advice pre-emptively, no longer requiring the current state to match the criteria for the user’s assistance.

Additionally, as rules require the user to set conditions for when a recommendation is provided, persistence of advice is introduced to the agent, allowing the retention and reuse of the information in the future. The contributions introduced so far in this dissertation are brought together in this final chapter. This chapter introduces Rule-Based Interactive Reinforcement Learning. Rule-Based Interactive Reinforcement Learning allows human advisors to provide additional information that is not constrained by the current state of the agent. This more informationally rich interaction method improves performance of the agent compared to existing methods, and reduces the number of interactions between the agent and the advisor.

8.1 Rules

In computer science, a rule is a statement consisting of a condition and a conclusion. An example of a rule is *'IF p THEN q'*, dictating that if the condition of p is met, then the conclusion is q . Additional qualifiers may supplement rules, allowing for a rules condition or conclusion to be constructed to meet specific demands. Qualifiers such as NOT or ELSE allow a rules conclusion to be reached if conditions are not met. For example, *'IF NOT p THEN u'* or *'IF p THEN q ELSE u'*. This simple structure for inference underpins many of the teaching methods that humans use to convey information.

When teaching or conveying information between people, one form of knowledge transfer is rules. While the syntax of the rule is not as formal as the examples previously given, the relation of condition and conclusion is maintained. For example, the sentence *'Don't touch the stove when hot'* can be represented formally as *'IF stove==hot THEN don't touch'*. Other examples of the use of rules in teaching and guidance are:

- Turn right when you get to the end of the road
- If the car is low on fuel it is time to start looking for a fuel pump
- Wear a jacket when it is cold outside.

For each of the examples above, the condition and conclusion are quickly identifiable by humans. Recent advances in speech-to-text systems have demonstrated the ability to identify the condition and conclusion in human speech. Assistant systems such as Google Assistant and Apple Siri have demonstrated this functionality through conditional reminders such as *'When I am at the supermarket remind me to get milk'* or *'Remind me at 3PM to call the restaurant'*(López, Quesada, & Guerrero, 2017). The ease with which humans can identify rules for knowledge transfer, and ability for machines to translate speech to rules, means that rules are an increasingly viable option for knowledge transfer for non-technical users(Cruz et al., 2015).

Current Interactive Reinforcement Learning methods limit human guidance and evaluation to the current state of the agent, regardless of whether the conditions for the information are shared among multiple states. This constraint requires the advising user to monitor the

current state of the agent and wait until conditions are met that suit the advice they wish to provide. Additionally, the user is required to repeat the advice for each state where the conditions are met. This lack of generalisation increases the number of interactions and the demand on the user.

The advice that a rule contains is paired with the conditions for its applicability. Because the reason why the advice applies is provided, the same rule can be used for every state in the environment. If the conditions are true for any given state, then the advice can be provided. This provides the generalisation that existing methods lack. Additionally, the user is not required to provide the same advice multiple times as long as the conditions for its applicability are the same.

8.1.1 Rules Trees

A user may create multiple rules over the duration of their assistance to an agent. As a result, a single state may have multiple rules, each with conflicting advice for the current state. Which conclusion should be followed for such states?

Binary decision trees offer a method of structuring rules in such a way that only one conclusion is given for each state. A binary decision tree is made up of nodes, each containing either a condition, a conclusion, or both. Each node has a parent, and up to two children. The parent is the node whose condition lead directly to the current node. The decision on which child to evaluate next is dependent on the whether the condition of the current node is met or not. The root node is the only node without a parent and is the entry to the tree. A node with no children is a terminal node and is the exit of the tree.

To navigate a binary tree, the root node is the first evaluated. The attributes of the current state are used to check the conditions of each rule. If the conditions of the current node evaluate as TRUE, then the TRUE child is moved to, otherwise the FALSE child is moved to. The process of evaluating the current node and moving to the next is repeated until a terminal node is reached.

The philosophy of early decision tree systems was that knowledge was static, and if captured in its entirety, would not need updating or correction. Traditionally, the approach to designing a decision tree involved employing an expert and a knowledge engineer. The

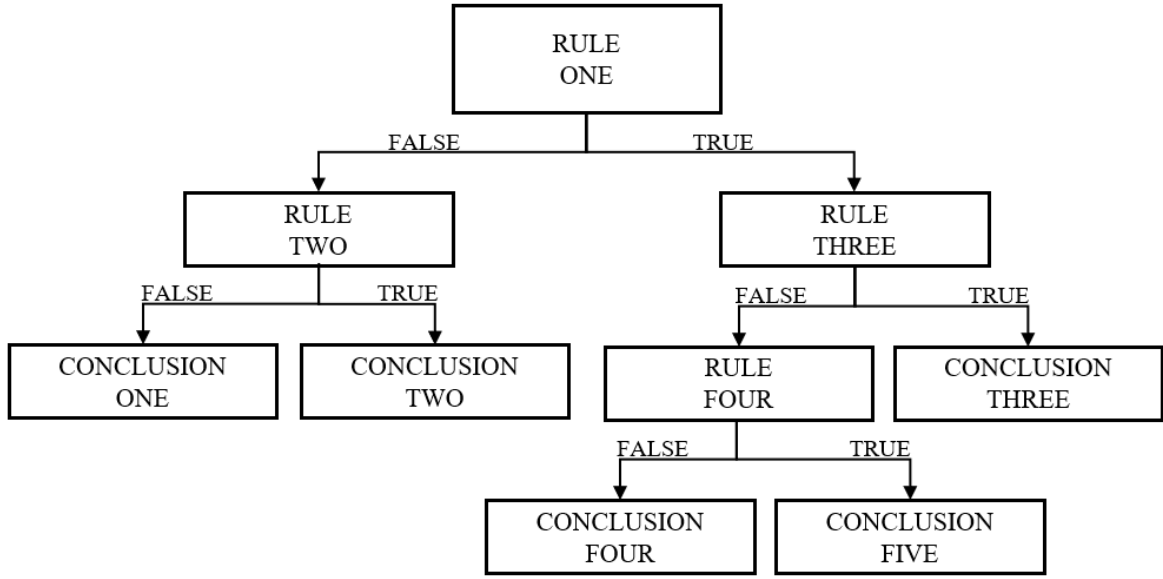


Figure 8.1: An example of a Binary Decision Tree.

expert held the knowledge to be captured in their head (Richards, 2009). Through exhaustive deliberation, the knowledge engineer and expert would transform the knowledge into a decision tree. Algorithms such as ID3 (Quinlan, 1986) and CART (Breiman, 2017) allow this process to be automated, provided that large amounts of labelled data are available. In the absence of labelled data, knowledge engineers are required to build trees by hand. This is often an arduous and time-consuming task, as each new rule needs to be constructed within the context of each preceding rule.

An additional concern with traditional decision tree methodologies is maintenance. Maintenance is the task of modifying an existing tree and is often performed with the intention of expanding or correcting the knowledge it captures. Maintenance is a difficult task, as it requires a deep understanding of the entire tree and the reasons why each rule was added in the past. For trees generating using algorithms such as ID3 and CART, this understanding of the tree is lost, as the rules were generated by the algorithm not by a knowledge engineer. In such cases, it is often easier to update the set of labelled data used to generate the original tree and build a new tree from scratch, an expensive process.

The methods for building decision trees mentioned so far do not meet the constraints set by Interactive Reinforcement Learning. Interactive Reinforcement Learning does not have access to large amounts of labelled data and aims to be within the skill level of non-expert

users, not specialized knowledge engineers. Rule-based Interactive Reinforcement Learning requires a method for generating binary decision trees without the need for expert skills in knowledge engineering, without large amounts of labelled data, and that can be built iteratively without the need for the user to know the full context of the tree.

8.1.2 Ripple-Down Rules

Ripple-Down Rules (RDR) is an iterative technique for building and maintaining binary decision trees (B. Kang et al., 1995; Compton et al., 1991). Introduced by Compton and Jansen (1988), RDR was developed in response to the concerns raised with traditional decision tree knowledge systems. While the philosophy of early decision tree systems was that knowledge was static, Compton and Jansen considered knowledge to be fluid and ever-changing. With such a philosophy in mind, they created a system in which knowledge acquisition was incremental, occurring naturally as domain experts interacted with the system over time.

Ripple-Down Rules is a combination of decision trees and case-based reasoning. A case is a collection of potentially relevant material that the system uses to make a classification and is equivalent to the concept of states in Reinforcement Learning. Each node in an RDR system contains a rule, a classification, and a case. The case paired with each node is referred to as the ‘cornerstone case’ and provides justification for the node’s creation.

The classification of a case is determined by the last node whose rule evaluated as TRUE using the current case. Beginning at the root of the tree, each rule within each node is evaluated using the attributes in the current case, not the cornerstone case. If the rule is evaluated as TRUE, the case is moved to down the TRUE branch, otherwise it is moved down the FALSE branch. This process continues until there are no more nodes for the current case to move to on the current branch. The last node in the branch is referred to as the ‘insertion node’ and the last node that evaluated true along the cases travelled path is referred to as the ‘classification node’. The ‘classification node’ holds the classification of the current case and the ‘insertion node’ is where corrections resulting from the current case will be appended.

The process of building a Ripple-Down Rule tree is incremental. Each tree begins with a root node containing a default classification. Should none of the nodes following the root node classify a case, the root node will return a classification. The root node does not

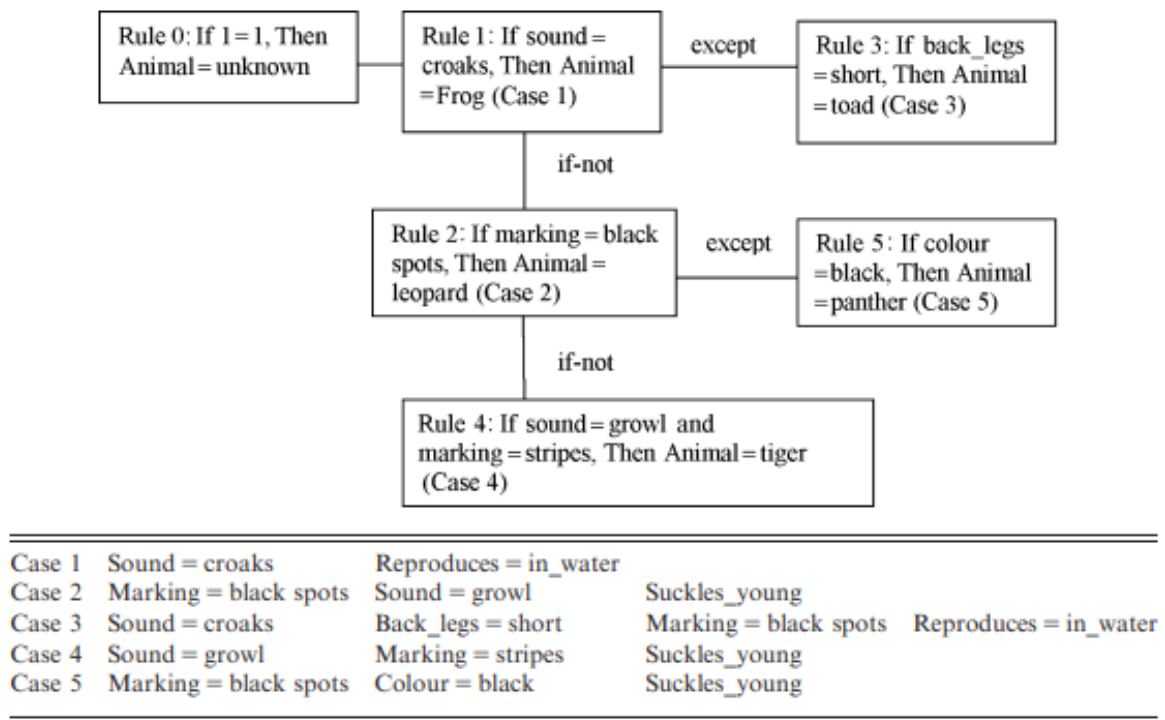


Figure 8.2: An example of a Ripple-Down Rules model, including cornerstone cases (Debbie Richards, 2009). The root node contains a rule that will always equate to true, and if no other rule in the tree equates true, the classification of the root node will be chosen. The tree is navigated down until a terminal node is reached. Once reached, the classification of the last node to equate true is returned.

have a cornerstone case or a rule based on case attributes; the node will always equate TRUE. Users provide cases to the system for classification. When the user disagrees with the systems classification, the cornerstone case assigned to the node that is responsible for the classification is presented to the user. The user identifies the differences between the current case and the cornerstone case and assigns a new classification. The differences highlighted are used by the system to generate a new rule. The new rule, current case, and corrected classification form a new node. The new node is appended to the insertion node, on the branch equal to the evaluation of the insertion node using the current case.

RDR systems require the user to only consider the difference between the current case and the cornerstone case. Using this methodology, the user does not need to know the context of the entire system, or how new rules will impact its structure. There is no need for a knowledge engineer for such a system, as no rules are written by the user and no oversight of the construction of the tree is needed. The iterative nature of Ripple-Down Rules also negates

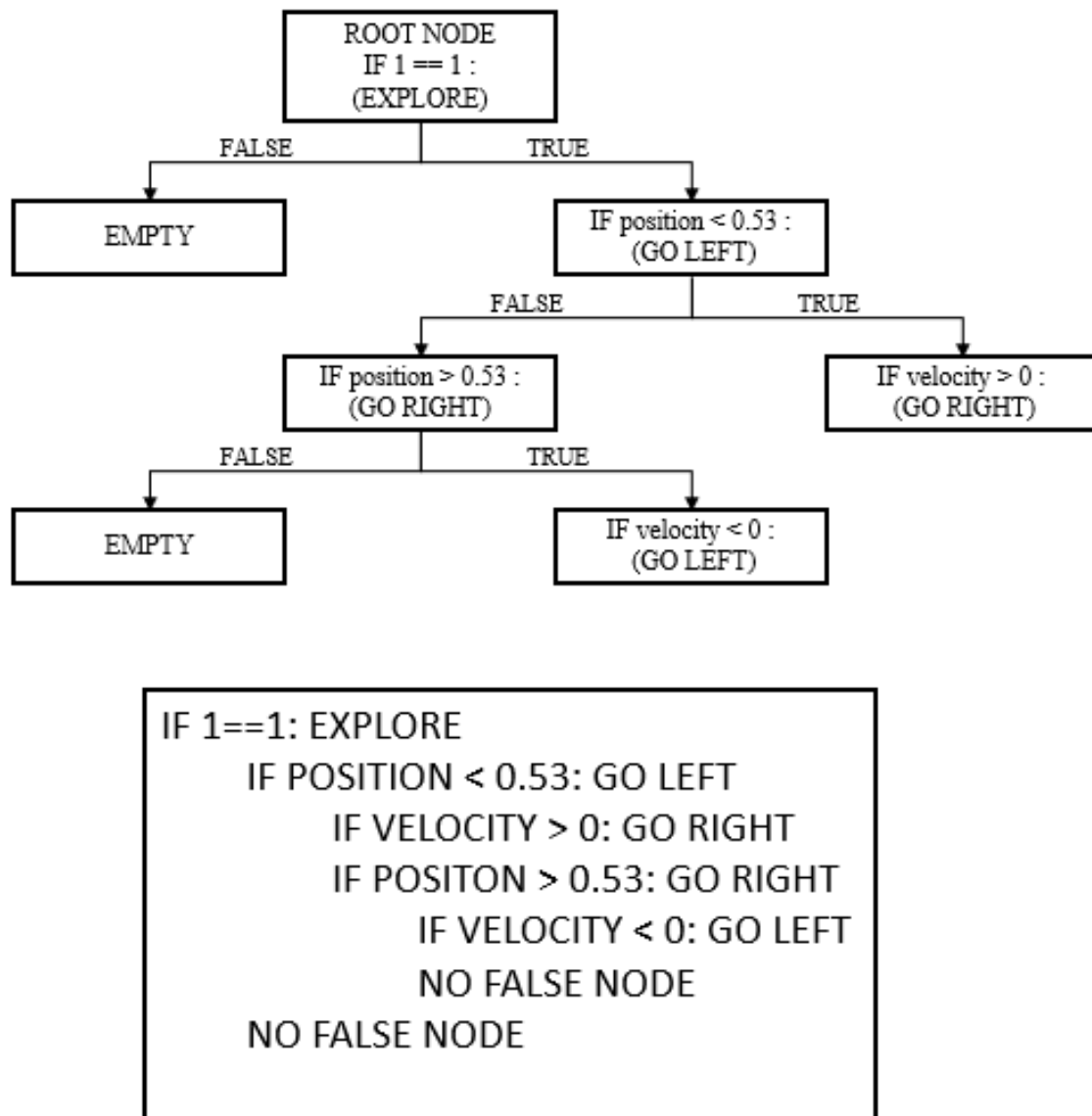


Figure 8.3: An example of a Ripple-Down Rules model, and a text representation of the same model. The example does not have the cornerstone cases listed. The text representation is used in the remainder of the chapter. The root node contains a rule that will always equate to true, and if no other rule in the tree equates true, the classification of the root node will be chosen. The tree is navigated down until a terminal node is reached. Once reached, the classification of the last node to equate true is returned. For example, if position is 0.75, and velocity is -0.2, then the classification returned by the tree will be GO RIGHT.

the need for large amounts of labelled data. Instead, the tree is built using the gradual flow of cases that any decision tree system is subject to. RDR offers many advantages over traditional decision tree systems. These distinguishing features include:

- Case-driven knowledge acquisition
- Exception-based structure
- Iterative development
- Knowledge acquisition and maintenance performed by user not engineer

These features make RDR suitable for rule-based Interactive Reinforcement Learning.

8.2 Rule-Based Interaction Reinforcement Learning

Existing Interactive Reinforcement Learning systems are limited to interactions that inform the current state only. While these systems have demonstrated substantial improvements to the performance of Reinforcement Learning agents, they take little consideration of the demand on the user advising the system. Additionally, existing interactive systems do not retain the advice the user provides, reducing the potential performance improvements and increasing the demand for user input and supervision.

The final contribution of this dissertation is a Rule-Based Interactive Reinforcement Learning agent. Rules as an advice delivery mechanism offers advantages over current methods. Rule-structured advice allow information to be generalised over multiple states. This reduces the interactions required with the human advisor while simultaneously increasing the potential benefit each interaction has on the agent's behaviour. The generalisation occurs because the user can specify the conditions in which the information is applicable, allowing the advice to be generalised beyond the current state. The agent can then check each state it encounters to see if the conditions are met, at which point the recommendation or evaluation can be utilised.

To mitigate issues of conflicting and incorrect advice, a method for managing and correcting retained information is required. Ripple-Down Rules offers a methodology for iteratively building knowledge-based systems without the need for engineering skills. RDR has an exception-driven and case-based approach to model design. This approach makes the addition and correction of information to the system simple and quick. The ability for ordinary users to be able to build a knowledge-based system makes RDR well-suited for Interactive Reinforcement Learning.

8.2.1 RDR-RL Agent

This section details the Rule-Based Reinforcement Learning agent contributed by this thesis. To assist with the readers comprehension, the agent is referred to as RDR-RL from this point on. While current Interactive Reinforcement Learning agents accept advice pertaining to the current state only, RDR-RL accepts rule-based advice that can apply to multiple states. Each interaction contains the recommendation/evaluation from the user and the conditions for its application. For example, the user may provide the following rule to an agent learning to drive a car. E.g. *“IF obstacle_on left==TRUE THEN action=turn_right”*. In this example, the advice is to turn right, and the condition for its use is that there is an obstacle on the left-hand side of the car. While Rule-Based Interactive Reinforcement Learning assumes that all interactions contain a rule, this rule does not have to be sourced directly from the user. The method in which the user interacts with the agent can be by any means, as long as the advice collected results in a set of conditions and the recommendation. The user may provide the set of conditions for the applicability of the advice directly, or optionally, the conditions may be discovered using assistive technologies such as case-based reasoning or speech-to-text.

The RDR-RL agent has three aspects to its construction, each of which are described below. These aspects are advice modelling, advice gathering, and advice utility.

8.3 Advice Gathering

The RDR-RL agent has the same foundation as any Reinforcement Learning agent. The ability to retain and use advice provided by the user is an addition to the Reinforcement Learning agent, built around the existing algorithm. Like existing Interactive Reinforcement Learning agents, when no advice has been provided to the agent, it will operate to the exact same as a standard Reinforcement Learning agent. Performance changes to the agent are not observed until advice has been provided to the agent.

In Chapter 5, the difference between evaluative and informative advice was highlighted. Informative advice, advice that recommends an action to perform, was found to be both more beneficial to the agent’s performance and preferred by the advising user. For this reason, the RDR-RL agent described in this chapter has been built for and testing with only informative

advice. However, there is not technical limitation preventing the system from not utilising both methods of advice simultaneously.

At any point during the agent’s learning, a user may assist the agent by providing recommending an action to take. When the user begins an interaction, they are provided with the agent’s current state, and if available, the current recommendation. Further details about the model of human advice and the cornerstone case are discussed in the next section. If the user agrees with the recommendation the agent presented, or if the user is no longer available, the agent continues learning on its own.

If the user disagrees with the action the agent is proposing, or if there is no action proposed, then the interaction continues. The user is provided with a cornerstone case. The cornerstone case is the state in which the user recommended the action that the agent is intending to take. The differences between the cornerstone case and the current state are presented to the user. If there is no cornerstone case, for example, when it is the first time the user is providing advice to the agent, then only the current state is provided. The user recommends an action for the agent to take, and creates a rule that distinguishes the two cases, setting the conditions for their recommended action.

Once the recommended action has been provided, and the rule setting the conditions for its use determined, they are passed to the agent. The agent then uses the rule and recommendation to update its model of human advice.

8.4 Advice Modelling

Advice modelling is the process of storing the information received from the user. The agent receives a rule and a recommendation from the user each time an interaction occurs. The rule dictates the conditions that must be met for the recommendation to be provided to the agent.

Chapter 7 introduced persistence for state-based Interactive Reinforcement Learning. In the experiments demonstrated in that chapter, the agents maintained a lookup table for each state and the corresponding recommendation/evaluation that had been provided. This simple method for advice modelling demonstrated improved performance compared to agents that did not retain advice. However, this lookup model did not generalise advice across multiple

states and had difficulty with incorrect advice.

For rule-based advice, a Ripple-Down Rules decision tree is used to model the advice provided by the user. This system allows a model of the human advice to be iteratively built over time, as the user provides more information to the agent. The RDR model is part of the Reinforcement Learning agent but is independent of the q-value policy. It is used to assist in action selection.

When an interaction with the user occurs, the agent is provided with an action recommendation, and a rule governing its use. To update the model of human advice, the agent provides the current state as a case to the RDR system. The system returns a classification node and an insertion node. The classification node contains the recommended action based on the human advice collected prior to the current state; the recommendation that the user disagrees with given the current state. The insertion node is the last node in the branch of the RDR tree that evaluated the current state and is the point at which the new rule will be inserted. A new node is created using the rule and recommendation from the user, along with the current state as the cornerstone case. If the rule in the insertion node evaluated TRUE using the information in the current state, then the new node will be inserted as a TRUE child, otherwise it will be inserted as a FALSE child.

The last aspect of the agent's construction details when the advice gathered from the user is used by the agent. In chapter 7, the concept of persistence was discussed. That chapter identified an issue in which performance of an agent was decreased if incorrect advice was provided, or recommended actions were always followed and neglecting exploration. To mitigate this issue, probabilistic policy reuse was proposed. PPR uses probabilistic bias to determine which exploration policy to use when multiple options are available, the goal of which is to balance random exploration, use of the guidance policy, and use of the current policy, derived from the agent's Q-values.

For the RDR-RL agent, the guidance policy is the model of human advice. The trade-off between exploration and the exploitation of the learned expected-rewards policy continues to be managed by whichever action selection method is preferred by the agent designer. An e-greedy action selection method is used for the experiments in this chapter. PPR manages switching between the action recommended by the advice model and the e-greedy action

selection method.

At each time step, the advising user has a chance to interact with the agent. If an interaction occurs, the model is updated. When a user first recommends an action, it is expected that the agent will perform it. For this reason, the recommended action is always performed on the time step at which it was recommended, regardless of the probabilities currently set by PPR.

When an agent is selecting an action in a time step where the user has not recommended an action, then PPR is used. First, the agent's model of advice is checked to see if any advice pertains to the current state. If the model recommends an action, then that action is taken with a probability determined by the PPR selection policy. If no action is recommended, then the agent's default action selection policy is used; e-greedy for example.

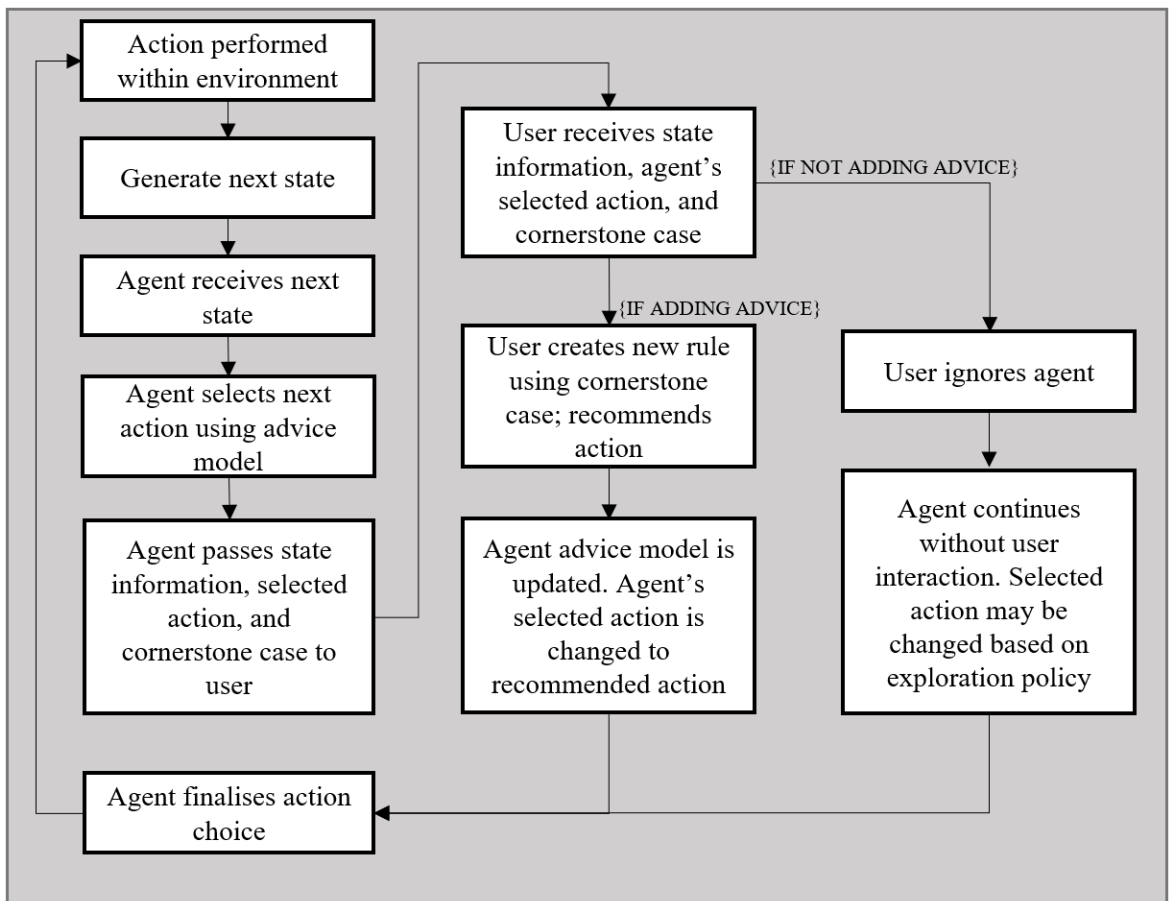


Figure 8.4: Process flow of a Rule-Based Interactive Reinforcement Learning agent.

8.5 Experiments and Methodology

In this chapter, state-based and rule-based advice delivery styles, and their respective impacts on agent performance and human interaction requirements, are compared. To compare agent performance and human interaction, metrics for agent steps, agent reward, and human interactions are recorded. A number of different agents and simulated users have been designed and applied to the mountain car, self-driving car, and Super Mario environments. Simulated users have been chosen over actual human trials, as they allow rapid and controlled experiments. When employing simulated users, interaction characteristics such as knowledge level, accuracy, and availability can be set to specified and measurable levels. To read more about simulated users and a methodology of their use, refer to Chapter 6.

8.5.1 Agents

For the following experiments, three agents have been designed, which include unassisted Q-Learning, persistent informative Q-Learning, and RDR Assisted Q-Learning. No evaluative assisted agents are tested in this chapter, as they cannot be suitably compared to the rule assisted agent which is using informative advice. Each agent used in this chapter is described below, with more specific details about their hyper-parameters and action selection strategies provided in the environment section later in this chapter.

- (i) **Benchmark Q-Learning Agent** A Q-learning agent used for capturing a baseline for performance on each environment. This agent is unassisted, receiving no guidance or evaluation from the human. The agent will represent each environment as described in the relevant sections in Chapter 4 unless otherwise specified. The expected-reward values have been initialized to zero. This agent uses e-greedy action selection.
- (ii) **Persistent Informative Advice Q-Learning Agent** This agent is assisted by a user. The user may recommend an action each time step for the agent to perform. If an action is recommended by the user, the agent will take it on that time step and retain the recommendation for use when it visits the same state in the future. When the agent visits a state in which it was previously recommended an action, it will take that action with the probability defined by the probability policy reuse (PPR, Section 7.5)

action selection strategy. The persistent informative agent uses the same parameter settings as the unassisted Q-Learning agent for each environment.

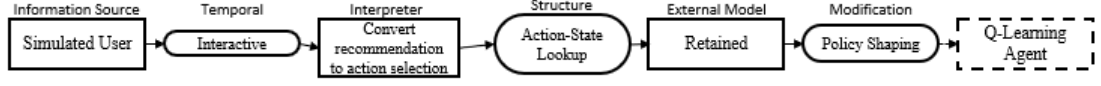


Figure 8.5: Definition of the persistent informative assisted experimental agent using the Assisted Reinforcement Learning framework.

- (iii) **Rule-Assisted Q-Learning Agent** This agent is assisted by a user. The user may provide a rule and recommended action at each time step. The rule-assisted Reinforcement Learning agent uses Ripple-Down Rules to model the advice received by the agent. Section 8.1.2 details how rules are retained and selected from the RDR model. If the user provides advice, and the rule provided equates to true for the current state, then the agent will take the recommended action during that time step. If the provided rule equates to false, then the agent will use its default action selection strategy. When a rule is provided the agent will retain the rule for use in future states. Each time the agent visits a state, it will query its retained model of rules. If a rule is found that equates to true for the current state, then that action is taken with a probability defined by the agent's probability policy reuse (PPR, Section 7.5) action selection strategy. All rule assisted agents used in this chapter begin with an 80% chance of taking the action recommended by its advice model. This 80% chance is decayed each episode, until the point at which the agent is relying solely on its secondary action selection strategy. The agent's secondary action selection strategy is the same strategy used by the unassisted Q-Learning agent, which is an e-greedy approach with parameters unique to each environment. The rule-assisted agent uses the same parameter settings as the unassisted Q-Learning agent for each environment.

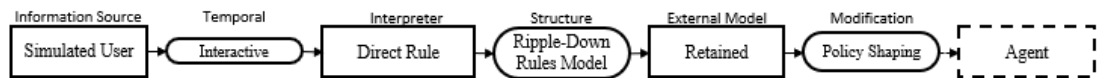


Figure 8.6: Definition of the persistent rule-based assisted experimental agent using the Assisted Reinforcement Learning framework.

8.5.2 Simulated Users

To allow quick, bias-reduced, repeatable testing of the agents in this chapter, simulated users are used in place of human trials. As demonstrated in Chapter 6, simulated users offer a method for performing indicative evaluations of Reinforcement Learning agents that require human input, with controlled parameters. There are two types of simulated users required for the following experiments, one must provide state-based advice, and the other must provide rule-based advice. Both types of simulated users will provide the same, and same amount, information. The level of knowledge for each state-based user is shown on the next page.

STATE-BASED SIMULATED USERS		
ENVIRONMENT	USER NAME	LIMITS
MOUNTAIN CAR	MC-FULL	USER WILL PROVIDE ADVICE FOR ALL STATES.
MOUNTAIN CAR	MC-HALF	USER WILL ONLY PROVIDE ADVICE FOR STATE IN WHICH THE AGENT IS ON THE LEFT SLOPE OF THE VALLEY. (IF POSITION < -0.53)
MOUNTAIN CAR	MC-QUARTER	USER WILL ONLY PROVIDE ADVICE FOR STATE IN WHICH THE AGENT IS ON THE BOTTOM HALF OF THE LEFT SLOPE OF THE VALLEY. (IF POSITION < -0.53 AND POSITION > -0.865)
MOUNTAIN CAR	MC-MIDDLE	USER WILL ONLY PROVIDE ADVICE FOR THE FEW STATES AT THE BOTTOM OF THE VALLEY. (IF POSITION < -0.43 AND POSITION > -0.63)
SELF-DRIVING CAR	SC-AVOID	USER WILL ONLY PROVIDE ADVICE FOR STATES WHERE THE AGENT HAS AN OBSTACLE ON THE LEFT SIDE OR THE RIGHT SIDE, BUT NOT BOTH. (IF RIGHT = TRUE OR RIGHT-FRONT-CLOSE = TRUE) OR (IF LEFT = TRUE OR LEFT-FRONT-CLOSE = TRUE)
MARIO	NONE	INFORMATIVE STATE-BASED ADVICE IS NOT TESTED FOR THE MARIO ENVIRONMENT

Table 8.1: State-based simulated user knowledge bases for the Mountain Car and Self-Driving Car environments.

The first type, an informative state-based advice user, is the same user employed for the previous chapter. This user may provide a recommended action on each time step. The agent that the user is assisting will retain any recommendations provided by the user, and will not give the user an opportunity to provide advice for a state for which advice has already been received, capping the number of interactions at the number of states. In this previous chapter, each informative state-based user had an accuracy and availability score. Accuracy is the probability that the advice that the user is providing is optimal for the current state. Availability was the probability that the user would provide advice for any given opportunity, and is a method for simulating the attention or engagement of

a user. For this chapter, all state-based advice users will have 100% accuracy and 100% availability. However, the states that the agent can provide advice may be limited to parts of the environment, simulating a limited or incomplete knowledge of the environment. Table 8.1 shows the knowledge limitations of the various state-based users built for the following experiments.

The advice that the state-based simulated users provide for the Mountain Car environment is optimal. However, the same may not be true for the simulated car environment. The reward function for the simulated car environment reinforces behaviour that avoids collisions and maximises speed. The advice that the simulated user for the self-driving car environment only attempts to avoid collisions. While this advice should be optimal, there may situations where the agent will want to stay close to an obstacle to maximise speed. In these situations, the advice provided would be considered incorrect, and the agent will need to learn to ignore it to learn the true optimal behaviour. The Mario environment is not tested with state-based advice, as the results from the first two environments are sufficient to compare rule-based advice against state-based advice.

The second type of simulated user required are rule-based advice giving users. These simulated users will return a rule and a recommended action for each interaction with the user. Simulated users are a common methodology for the creation and evaluation of Ripple-Down Rule system in research (Dazeley & Kang, 2004b; B. Kang et al., 1995; B. H. Kang et al., 1998; Compton et al., 1995). The simulated users employed for the following experiments have been built with their own Ripple-Down Rules model, and populated with a set of rules that they will, over time, provide to the agent. In reality, users do not have their own rule model, rather they would generate rules themselves through thought and observation. The rule model is required for the simulated user as a means to replicate the interaction process of a real user.

The agent begins each experiment with an empty model of human advice, and the simulated user begins with a full model. Over time, the agent will provide the human an opportunity to provide advice. When an opportunity occurs, the agent provides the current state observation, the current action it will take, and details about how it chose that action (either from the retained user model, or from an exploration strategy). This information is

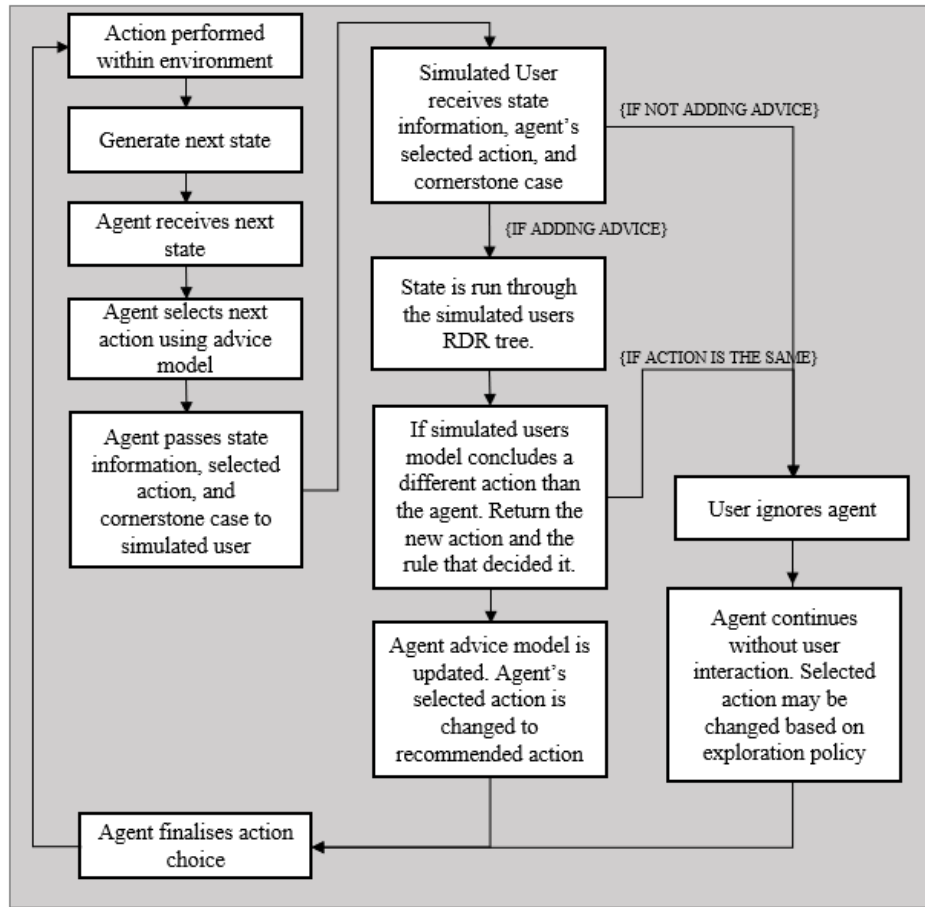


Figure 8.7: Process flow for a rule-based simulated user using a Ripple-Down Rules model assisting a rule-based reinforcement learning agent.

the same information that would be made available to an actual human advisor. Now that the simulated user has this information, it may choose to respond and what advice it will respond with. The simulated user will respond if it has a rule that applies to the current state and it disagrees with the agent's choice of action. In reality, human users can provide new rules whenever the agent provides the opportunity, regardless of if the new rule applies to the current state or if they disagree with the user. For these experiments, the simulated user is constrained to these conditions, simply because it is likely that actual humans will only provide new advice in response to disagreements with the agent. The simulated user will continue to provide advice for as long as it is given opportunities, that it has new rules to provide, and that the new rules disagree with the agent's current behaviour.

Multiple rule-based simulated users have been created to provide a range of different knowledge levels for the various environments. Table 8.2 and Table 8.3 describes and pro-

vides the knowledge bases in use for each of the environments. A short description of each knowledge base is provided.

RULE-BASED SIMULATED USERS		
ENVIRONMENT	USER NAME	LIMITS
MOUNTAIN CAR	MC-FULL	<p>USER WILL PROVIDE ADVICE FOR ALL STATES.</p> <p>IF 1==1 : EXPLORE IF VELOCITY > 0: GO RIGHT NO TRUE NODE IF VELOCITY <=0 GO LEFT</p>
MOUNTAIN CAR	MC-HALF	<p>USER WILL ONLY PROVIDE ADVICE FOR STATES IN WHICH THE AGENT IS ON THE LEFT SLOPE OF THE VALLEY.</p> <p>IF 1==1 : EXPLORE IF POSITION < -0.53: GO RIGHT IF VELOCITY >= 0: GO RIGHT IF VELOCITY < 0: GO LEFT NO FALSE NODE</p>
MOUNTAIN CAR	MC-QUARTER	<p>USER WILL ONLY PROVIDE ADVICE FOR STATES IN WHICH THE AGENT IS ON THE BOTTOM HALF OF THE LEFT SLOPE OF THE VALLEY.</p> <p>IF 1==1 : EXPLORE IF POSITION < -0.53 AND POSITION > -0.86: GO RIGHT IF VELOCITY >= 0: GO RIGHT IF VELOCITY < 0: GO LEFT NO FALSE NODE</p>
MOUNTAIN CAR	MC-MIDDLE	<p>USER WILL ONLY PROVIDED ADVICE FOR THE FEW STATES AT THE BOTTOM OF THE VALLEY.</p> <p>IF 1==1 : EXPLORE IF POSITION < -0.43 AND POSITION > -0.63: GO RIGHT IF VELOCITY >= 0: GO RIGHT IF VELOCITY < 0: GO LEFT NO FALSE NODE</p>
SELF-DRIVING CAR	SC-AVOID	<p>USER WILL ONLY PROVIDE ADVICE FOR STATES WHERE THE AGENT HAS AN OBSTACLE ON THE LEFT SIDE OR THE RIGHT SIDE, BUT NOT BOTH.</p> <p>IF 1==1 : EXPLORE IF RIGHT OR RIGHT-FRONT-CLOSE: TURN LEFT NO TRUE NODE IF LEFT OR LEFT-FRONT-CLOSE: TURN RIGHT NO FALSE NODE</p>

Table 8.2: Rule-based simulated user knowledge bases for the mountain car and self-driving car environments.

RULE-BASED SIMULATED USERS	
ENVIRONMENT	LIMITS
MARIO	<p>USER WILL PROVIDE THE AGENT RULES FOR A SIMPLE STRATEGY FOR PLAYING THE SUPER MARIO GAME. THE RULES ENCOURAGE THE AGENT TO JUMP RIGHT UNLESS THERE IS AN ENEMY NEARBY. IF THERE IS AN ENEMY, THE AGENT IS PROVIDED NO ADVICE, AND WILL RESORT TO ITS OWN EXPLORATION POLICY TO LEARN HOW TO HANDLE ENEMIES. IF THERE IS NO ENEMY NEARBY, THE AGENT WILL LOOK FOR ITEMS WHILE MOVING RIGHT. IF THERE IS AN ITEM NEARBY, THE AGENT WILL MOVE TOWARDS IT, UNLESS THE ITEM IS MORE THAN 2 TILES ABOVE IT.</p> <p>THIS ADVICE IS INCOMPLETE IN RESPECT TO THE OPTIMAL POLICY. THE ADVICE IS ALSO PARTIALLY INACCURATE AS THERE WILL BE TIMES WHEN THE AGENT SHOULD IGNORE NEARBY ITEMS TO PRIORITISE OTHER REWARDS. ADDITIONALLY, THE ADVICE ENCOURAGES THE AGENT TO MOVE RIGHT, WHEN THERE ARE TIMES WHEN MOVING LEFT IS OPTIMAL, TO REACH AN ITEM OR PLATFORM FOR EXAMPLE.</p> <p>IF 1==1 : EXPLORE IF CLOSESTENEMYXDISTANCE > 10 OR CLOSESTENEMYXDISTANCE < 2: JUMP RIGHT IF ONGROUND AND CLOSESTITEMXDISTANCE > 2: RIGHT OR JUMP RIGHT IF CLOSESTITEMXDISTANCE < -2: LEFT OR JUMP LEFT IF CLOSESTITEMYDISTANCE > : EXPLORE NO FALSE NODE NO FALSE NODE</p>

Table 8.3: Rule-based simulated user knowledge base for the Super Mario environment.

8.5.3 Environments and Experiments

The mountain car environment is a common benchmark problem for the Reinforcement Learning field because of its small state and action space, and simple dynamics. While the dynamics of the environment are simple, the environment requires that the agent perform the optimal action consistently if it is make it to the goal state. The mountain car environment is a good candidate for Rule-Based Interactive Reinforcement Learning as the optimal solution can be captured in very few rules, while still remaining understandable by humans. A detailed specification of the mountain car environment is provided in Section 4.1.2. The rule-based and state-based agents are tested against the mountain car environment, employing simulated users with varying levels of knowledge of the environment. The aim is to compare the performance of the agents, and the number of interactions performed to achieve that performance. The mountain car agents are given a learning rate of 0.25, a discounting of 0.9, and used an e-greedy action selection strategy with an epsilon of 0.05.

The self-driving car environment has the agent take control of a car and navigate an environment. The goal of the agent is to learn a behaviour that maximises the cars velocity

while avoiding collisions. The state and action spaces for this environment is larger than the mountain car environment, but still remain understandable by human observers. The self-driving car agents are given a learning rate of 0.1, a discounting of 0.999, and used an e-greedy action selection strategy with an epsilon of 0.01.

The requirements of the reward function, to avoid collisions and to maximise velocity, make the creation of optimal rules much more difficult. For the self-driving car environment, it is easy to provide rules that will help achieve greater performance in parts of the environment, maximising speed OR when to turn for example. However, it is much more difficult to provide rules that meet both requirements optimally, for example, when to turn the car and by how much to maintain the highest possible velocity while not crashing. The characteristic of being able to easily creating performance improving yet non-optimal rules, is what makes the self-driving car environment an interesting benchmark for Rule-Based Interactive Reinforcement Learning. A detailed specification of the self-driving car environment is provided in Section 4.1.3. The rule-based and state-based agents are tested against the self-driving car environment, employing simulated users with varying levels of knowledge of the environment. The aim is to compare the performance of the agents, and the number of interactions performed to achieve that performance. The difference between this environment and the mountain car environment is that this environment will test a larger state and feature space, and consist of advice that, while beneficial, is not optimal.

The final environment is the Super Mario game environment. The dynamics and reward function of the Mario environment are complex, and are not intuitive to human observers. For example, it is not easily apparent to the advising user whether the optimal behaviour is to rush to the end of each level, or to attempt to maximise the score by collecting items and killing enemies. Because of this reason, the advising human may believe they are providing rules for optimal behaviour, when in reality they are doing the opposite. There are two implementations of the Mario environment used in the following experiments. A summary of each implementation is provided below. For a detailed specification of the Mario environment, and each of the state-feature implementations, refer to 4.1.4. The Mario agents are given a learning rate of 0.001, a discounting of 0.9, and used an e-greedy action selection strategy with an epsilon of 0.05.

The first Mario state feature implementation, named the Littman implementation, has a very large state space which is not easily interpretable by the observing human (Goschin et al., 2013). The Littman implementation has a constantly changing number of state features. These state features list the exact position, velocity, and accelerations of every visible entity as continuous values, as well as a representation of each of the visible 352 tiles. Due to the detail and size of the the Littman implementation, typical advice givers may have difficulty creating rules using the implementation.

The second implementation of the Mario environment, named the Brys+ implementation, is much simpler and more understandable compared to the Littman implementation (Brys, 2016; Harutyunyan, Brys, et al., 2015). This simplification is due to a large amount of abstraction and discarding of state features. Rather than showing the exact position, velocity, and acceleration of all entities on the screen, the Brys+ implementation only shows the number of tiles away the nearest entity is. Furthermore, the Brys+ implementation is entirely discrete, and has a static number of state features. This abstraction and reduction of state features allows humans to more easily create rules for providing advice, as well as speed up learning of the agent.

It stands to reason that while an agent’s learning may be faster using the reduced state space of the Brys+ implementation, the agent should learn a better solution using the more detailed and finer-grained information provided by the Littman implementation. However, it may be more difficult for an advising user to create rules for the Littman implementation, so for the following experiments the user will only provide advice in the context of the Brys+ implementation. Because the information required to represent a state in Brys+ format can be sourced entirely from the information from a Littman state, it allows both the user and the agent to use the implementation best suited for them.

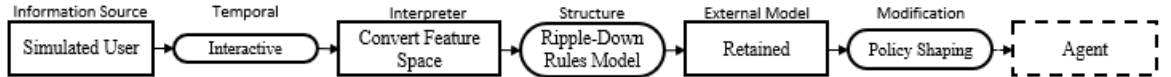


Figure 8.8: Definition of the persistent rules-based assisted experimental agent with feature set interpretation using the Assisted Reinforcement Learning framework.

For the following experiments, the agent will receive state information provided in the Littman format but transform the state to Brys+ format before asking the user for assistance or checking its model for suitable recommendations. This way, the human does not need to know the representation the agent is using, and can use the more user-friendly format, while the agent can continue to use the more detailed implementation that should lead to a more optimal behaviour. Having the agent learn from one feature space, while receiving advice in terms of another set of features is likely to be useful for environments with large feature sets, or features that are difficult for users to comprehend, such as image/pixel based environments.

To test the use of two feature sets, one for the agent and one for the user, there are a few experiments to run. The first is to compare the learning speed and end behaviour for each of the standalone unassisted implementations. The second is to test the achievable performance and learning speed when the agent and user are both using the Brys+ implementation. The final experiment will test the achievable performance and learning speed of the agent when it is using the Littman implementation and assisted using the Brys+ implementation.

8.6 Results

8.6.1 Mountain Car

Figure 8.9 shows the number of steps each agent performed each episode for the Mountain Car environment. The graph on the left shows the results for the rule-based agents, and the graph on the right shows the state-based agents. A comparison of the two graphs shows that the agents performed the same, regardless of the advice delivery method. This was expected, as the method in which the agent uses the advice and the amount of advice in total that the agent receives does not differ between the two types of agents, because the user's availability was 100%. The agents using minimal advice end up learning a worse behavior than the unassisted Q-Learning agent. This is likely an indication that the decay rate for the PPR action selection method is too low, and that the agent has not yet learned to ignore the human advice after its initial benefit and focus on its own learning.

Table 8.4 shows the number of interactions, and the percentage of interactions over opportunities for interactions, for each agent. These results show that the number of interactions

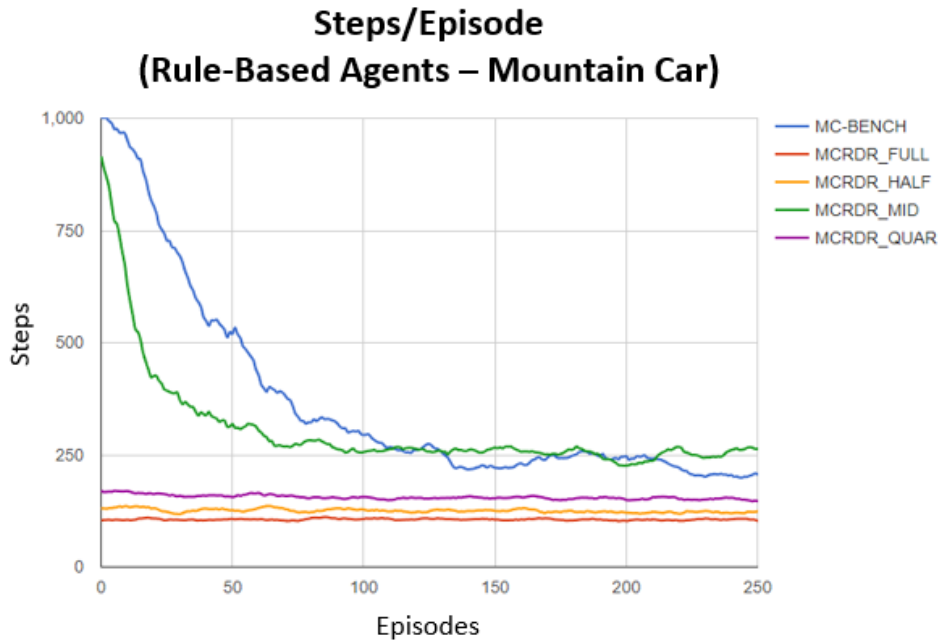
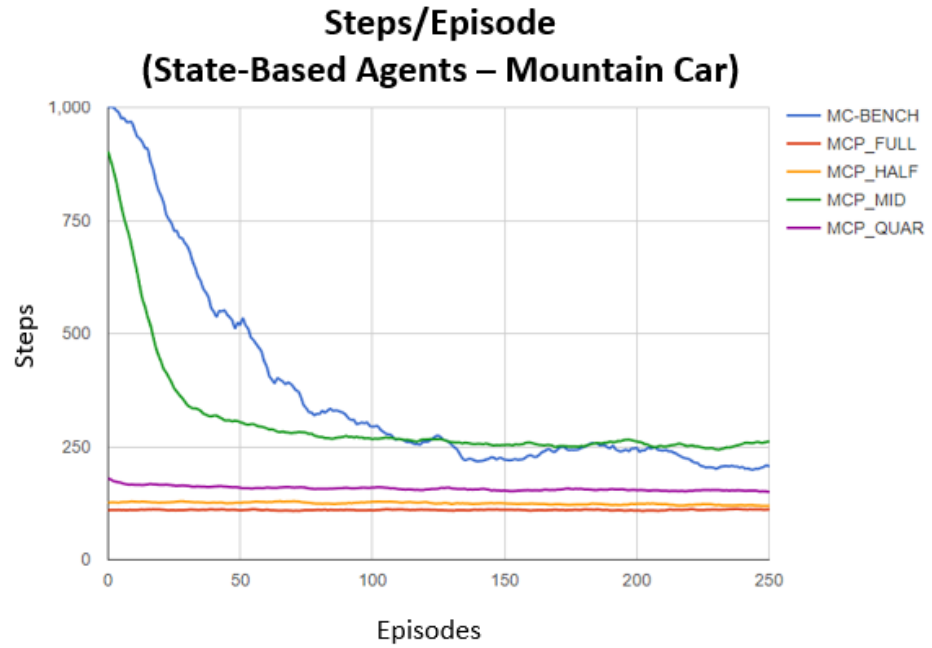


Figure 8.9: Step performance for rule-based and state-based Interactive Reinforcement Learning agents for the mountain car environment. Results show no significant difference in performance between the two types of agents when advisor knowledge, accuracy, and availability are the same.

is much less for the rule-based agents compared to the state-based agents. In past chapters, the number of interactions was not a useful measure to compare agents against each other. This is because the advice provided to the agent's effects the number of steps the agent takes, which results in fewer opportunities for interactions. However, Figure 8.9 shows that

INTERACTION PERCENTAGE RULES AND STATE BASED AGENTS FOR MOUNTAIN CAR ENVIRONMENT	
AGENT	#INTERACTION (% INTERACTIONS / TOTAL STEPS)
UNASSISTED Q-LEARNING BENCHMARK	0.00%
STATE-BASED AGENT, FULL ADVICE. (MCP-FULL)	254 (<0.01%)
STATE-BASED AGENT, HALF ADVICE. (MCP-HALF)	227 (<0.01%)
STATE-BASED AGENT, QUARTER ADVICE. (MCP-QUAR)	139 (<0.01%)
STATE-BASED AGENT, TINY ADVICE. (MCP-MID)	45 (<0.01%)
RULE-BASED AGENT, FULL ADVICE. (MCRDR-FULL)	2 (<0.01%)
RULE-BASED AGENT, HALF ADVICE. (MCRDR-HALF)	3 (<0.01%)
RULE-BASED AGENT, QUARTER ADVICE. (MCRDR-QUAR)	3 (<0.01%)
RULE-BASED AGENT, TINY ADVICE. (MCRDR-MID)	3 (<0.01%)

Table 8.4: Average number of interactions performed per experiment, and the percentage of interactions compared to the steps taken, for each state-based/rule-based agent/user combination in the Mountain Car environment.

the performance the agents that use the same simulated user are the same, regardless of the advice type. This means that the number of interactions is a useful measure for comparing the corresponding state-based and rule-based agents in Table 8.4.

8.6.2 Self-Driving Car

As with Figure 8.9, Figures 8.10 and 8.11 show no discernible difference in the performance of the rule-based agent compared to the state-based agent. The aim of the agent in the self-driving car environment is to avoid collisions and maximise speed. Both agents outperformed the unassisted Q-Learning agent, both achieving a higher step count and reward. The agent is forcibly terminated when it reaches 3000 steps. The figures show that the agents never quite reach the 3000 step limit, this is because the agents are given a random starting position and velocity at the beginning of each episode, some of which result in scenarios where the agent cannot avoid a crash.

Figure 8.5 shows the number of interactions, and the percentage of interactions compared to the number of opportunities for interactions (equal to steps), for each agent. These results show that the number of interactions is much less for the rule-based agents compared to the state-based agents.

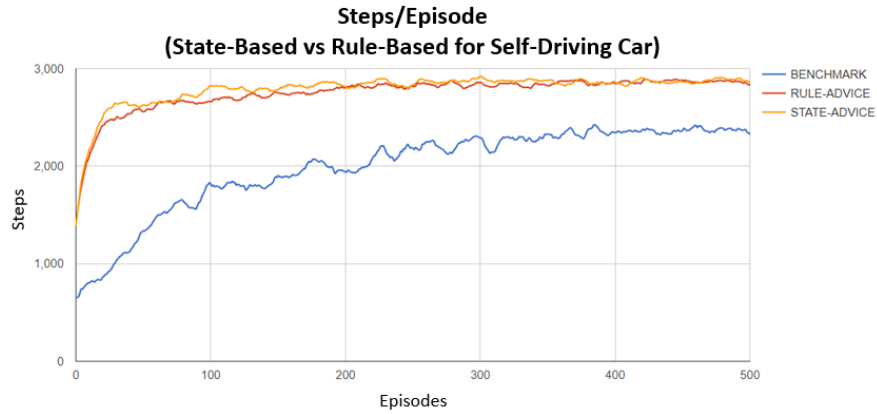


Figure 8.10: Step performance for rule-based and state-based Interactive Reinforcement Learning agents for the self-driving car environment. Results show no significant difference in performance between the two types of agents when advisor knowledge, accuracy, and availability are the same.

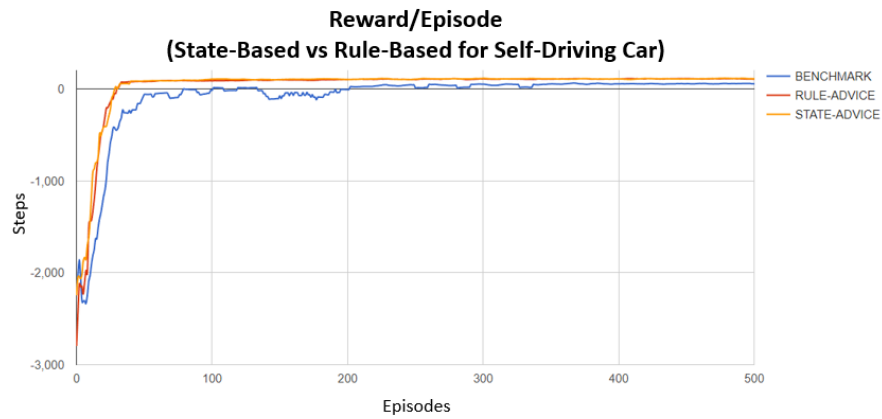


Figure 8.11: Step performance for rule-based and state-based Interactive Reinforcement Learning agents for the self-driving car environment. Results show no significant difference in performance between the two types of agents when advisor knowledge, accuracy, and availability are the same.

INTERACTION PERCENTAGE RULES AND STATE BASED AGENTS FOR SELF-DRIVING CAR ENVIRONMENT	
AGENT	#INTERACTION (% INTERACTIONS / TOTAL STEPS)
UNASSISTED Q-LEARNING BENCHMARK	0.00%
STATE-BASED SELF-DRIVING CAR AGENT(STATE-ADVICE)	232 (<0.01%)
RULE-BASED SELF-DRIVING CAR AGENT(RULE-ADVICE)	2 (<0.01%)

Table 8.5: Average number of interactions performed per experiment, and the percentage of interactions compared to the steps taken, for each state-based/rule-based agent/user combination in the Self-Driving Car environment.

8.6.3 Super Mario Brothers

As discussed in this chapter, and in Chapter 4, the Super Mario experiments performed in this thesis use two different feature sets. The first is the Littman implementation (Goschin et al., 2013), which has at least 359 state features, and increases with the number of enemies and items in the environment. The Littman feature set is very detailed, containing multiple continuous features, as well as a full representation of the environment. The second implementation is named Brys+, and is a small extension to the original implementation used by Tim Brys in his research (Brys, 2016; Harutyunyan, Brys, et al., 2015) (See Chapter 4 (Table 4.2) for a list of modifications. The Brys+ feature set has a constant 31 features, all discrete. By comparison, the Brys+ implementation is much simpler than the Littman implementation, but contains much less information about the environment.

Figure 8.12 show the benchmark performance of an unassisted Q-Learning agent for the Littman and Brys+ implementations of the Super Mario environment. These results show that while the agent using the Brys+ implementation initially learnt more quickly than the

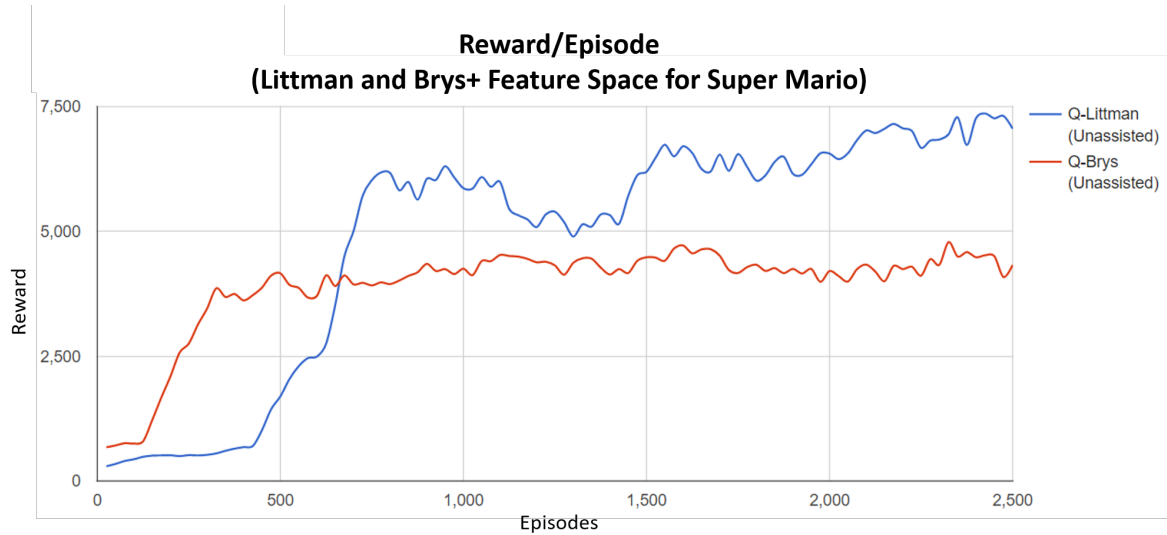


Figure 8.12: Unassisted Q-Learning benchmark performance on Super Mario using Littman and Brys+ feature spaces. These results show that an unassisted Q-Learning agent will learn faster using the Brys+ implementation. However, when using a Littman implementation, the agent can learn a better solution despite taking longer to learn.

Littman agent, it was unable to learn a better solution. This is expected, as the Littman implementation contains more information, resulting in a slower learning speed but a better fitting solution.

The Super Mario environment is complex and quite large, making the ability for a user to provide a full and accurate advice model difficult. Additionally, it is not easily identifiable whether the strategy to maximise reward is to collect all the items and kill all the enemies, or to race to the end of the environment to collect the time bonus (See Chapter 4 for more information). The simulated user designed for the following experiments attempts to provide advice that considers both of the strategies. The user will provide advice that encourages the user to move right, to the end of the environment, but also collect items that are easily reachable along the way. To provide advice, the user is using the Brys+ feature set, as it is short and discrete, making the process of constructing and providing rules easier than for the Littman implementation.

Figure 8.13 shows the results of a rule-assisted interactive reinforcement learning agent,

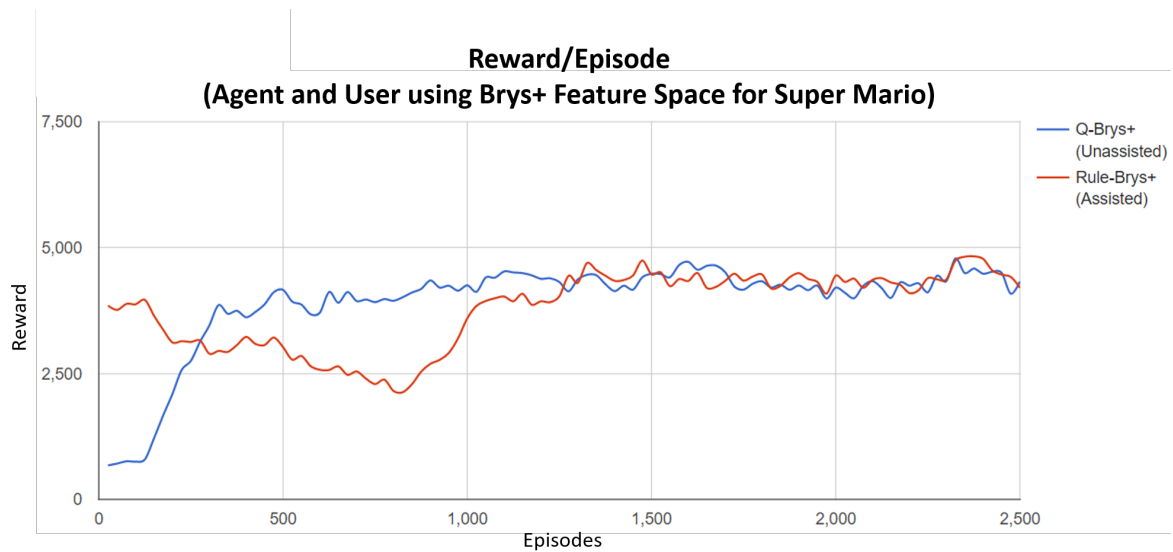


Figure 8.13: Rule-Assisted Interactive Reinforcement Learning on Super Mario using a Brys+ advice and state feature set. The assisted agent initially has a better performance. However, as the agent begins to ignore the human advice and search for the optimal behaviour, the unassisted agent outperforms the assisted agent. The benefit of advice in this situation is debatable. While both agents have roughly equal total reward, the assisted agent had a higher minimal performance.

where the agent and the advisor used the Brys+ feature set. There results show that the assisted agent initially learnt much faster than the unassisted Brys+ agent. However, the performance of the assisted agent dropped over time, as the began to ignore the advice it received, and rely on it's own policy and exploration strategy. The benefit of advice in this situation is debatable. The cumulative performance of the two agents are the same, within error margin. However, the reward from the worst performing episodes for the unassisted agent is substantially lower than the worst episodes for the assisted agent.

The results from Figure 8.12 shows that an agent can learn a better policy when using the Littman feature set. However, it is easier for the human to provide advice using the Brys+ feature set. As discussed earlier in this Chapter, all the information required to construct a Brys+ observation can be derived from a Littman observation. This allows the agent to learn a policy for the Littman observation, but accept and use advice provided with the Brys+ feature set. The agent interprets the Brys+ advice, and applies it to the current Littman observation for each timestep.

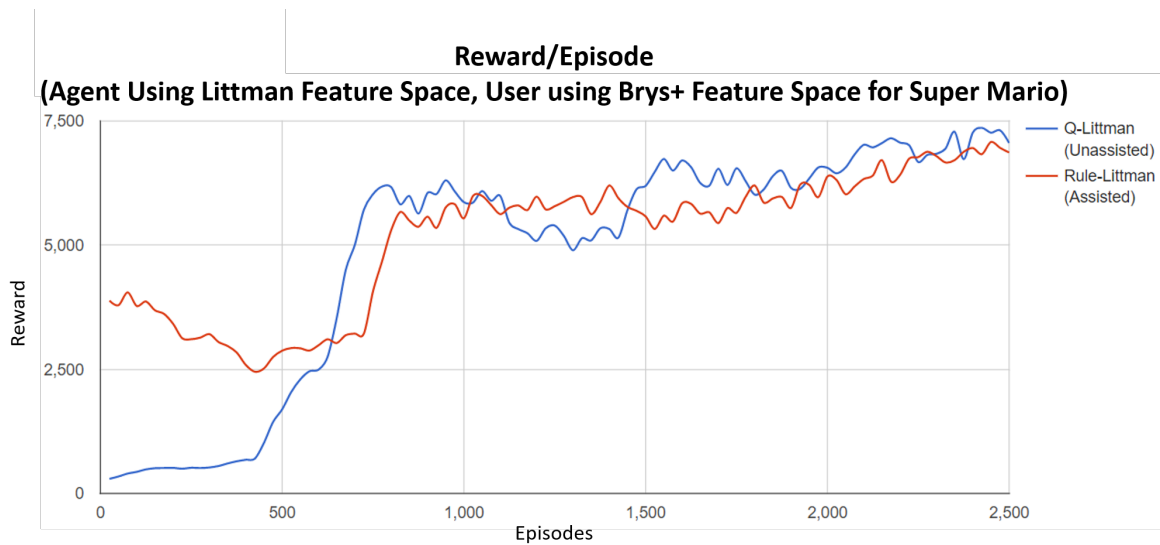


Figure 8.14: Rule-Assisted Interactive Reinforcement Learning on Super Mario using a Brys+ advice feature set and a Littman state feature set. The user provided advice in the context of a Brys+ implementation and the agent learnt a policy for the Littman implementation. The assisted agent has a greater performance initially, but degrades as it begins to disregard user advice in favour of its own exploration policy. Both agents end up learning the same behaviour, with the assisted agent slightly outperforming the unassisted agent overall.

Figure 8.14 shows the results of a rule-assisted agent that is learning a policy for the Littman feature set, but receives advice for the Brys+ feature set. These results show that the assisted agent had an immediate performance gain, but performance dropped as the agent incorporated the advice and began its own exploration. However, the assisted agent was able to recover and match the performance of the unassisted agent with little delay.

The number of interactions performed by the user for each of these experiment was equal to the number of rules the user could provide. The simulated user built for Mario has 4 rules, so each experiment only recorded 4 interactions.

8.7 Conclusion

This chapter introduced rule-based Interactive Reinforcement Learning, a method for users to assisted agents through the use of rule-structured advice and retention. Three environments were tested to investigate the impact that rule-based advice has on performance and the number of interactions performed to achieve the measured performance. Two of the environments also tested the use of rules that, while not optimal for the reward function, would still provide beneficial advice to the agent. These tests found that the agent can use this advice to improve learning speed, and still learn to ignore the incorrect/non-optimal advice later to achieve the optimal behaviour. Compared to state-based advice for Interactive Reinforcement Learning, rule-based advice was able to achieve the same level of performance with substantially fewer interactions between the agent and the user.

This chapter did not investigate the time and cognitive requirements for users to construct state-based and rule-based advice. It is likely that rule-based advice will require more time and thought to construct. However, existing research has shown that decision trees built with Ripple-Down Rules are easier for users to construct (Gaines & Compton, 1995; Compton et al., 1991; Compton, Peters, Edwards, & Lavers, 2006). Future work is required to test if this will justify the benefits that rules provide over state-based advice, in terms of the number of interactions.

Chapter 9

Conclusion and Future Work

This chapter summarises the contributions made by this thesis, and highlights directions for future research.

9.1 Contributions

9.1.1 Assisted Reinforcement Learning

The first contribution made was a taxonomy and framework for describing and classifying Reinforcement Learning agents that utilise external information to leverage the learning process and supplement the environment and reward functions. The taxonomy and framework has been named Assisted Reinforcement Learning. The Assisted Reinforcement Learning framework has been designed to promote collaboration between the sub-fields of Reinforcement Learning, and to help in describing and comparing the methods they introduce.

9.1.2 Evaluative versus Informative Advice

The second contribution was a comparison of evaluative and informative advice giving styles, where the accuracy, availability, and number of interactions of the two advice delivery styles were measured for the Mountain Car environment. A human trial was performed which found that informative advice givers were more accurate, had higher engagement, and better understood the behaviour of the agent than the evaluative advice-giving users. The findings of this trial should inform future Interactive Reinforcement Learning development, as it has in this thesis, with a greater emphasis on the engagement of the human and the informative advice they provide.

9.1.3 Simulated Users in Interactive Reinforcement Learning

The third contribution was the introduction of simulated users into Interactive Reinforcement Learning. While simulated users have been used in the field before, they have not been formally acknowledged, and a methodology for their design and use has not been published. This thesis showed that simulated users present an effective method for providing indicative evaluations of Interactive Reinforcement Learning agents for comparison and development.

The fourth and fifth contributions of this thesis were a list of characteristics of human interactions used for designing simulated users, and a set of principles for the evaluation of simulated users that were adopted from the spoken dialogue systems field.

9.1.4 Persistent Advice

The sixth contribution of this thesis is the introduction of a method for the retention and reuse of human-sourced advice, named persistence. The use of persistent advice was found to substantially improve the performance of the agent while reducing the number of interactions required of the human. To handle the risk that incorrect advice introduces, and to manage the exploration-exploitation trade-off, probabilistic policy reuse was introduced. PPR was found to be a viable method to balance the advantages and disadvantages that retained advice provides.

9.1.5 Rules-Based Reinforcement Learning

The final contribution of this thesis was Rules-Based Interactive Reinforcement Learning. Rules as an advice delivery method was shown to provided the same performance impact as state-based advice, but with a substantially reduced interaction count. Rules allow advice to be provided that generalises over multiple states. Coupled with the exception-driven decision tree generation algorithm Ripple-Down Rules, conflicting rules are managed and a rule model can be built interactively. Additionally, Ripple-Down rules has previously been shown to assist users in defining rules(Gaines & Compton, 1995; Compton et al., 1991, 2006).

9.2 Future Work

This section discusses possible future directions for the research presented in this thesis. In general, all the techniques and concepts contributed by this thesis should be tested and validated in more complex settings such as higher dimensional and continuous state spaces, and in real-world scenarios. Additionally, these contributions should be tested in combination with other technologies such as function approximators, deep learning, and AI safety, to demonstrate its performance on the leading edge of Reinforcement Learning. The remainder of this section lists some more specific research directions for simulated users, Interactive Reinforcement Learning, and human-agent interaction.

9.2.1 Effect of Latency on Accuracy

Chapter 5 performed a human trial that compared evaluative and informative advice delivery, measuring human accuracy, availability, and engagement. Humans providing evaluative advice were observed to have considerably worse accuracy than informative advice-giving users. A potential reason for this is disparity in latency. If the humans were late in giving their advice, accuracy would suffer more for evaluative advice givers than informative advice givers on the Mountain Car environment. A more detailed study that compared latency for the environment, and between advice delivery methods, should be a future direction of research.

9.2.2 Simulated Users

This dissertation introduced the use of simulated users into Reinforcement Learning for the purpose of indicative evaluation and development of RL technologies. In chapter 4, a list of characteristics for human interactions was introduced. This list included accuracy, availability, and knowledge level, all of which were extensively used in this research. The other interactions included concept drift, reward and cognitive bias, and latency. Future research is required to investigate methods for simulating these characteristics for different types of advice delivery methods and environments. Additionally, a larger and more comprehensive trial is required to compare how well simulated users can replicate the interaction behaviour

of real humans, and to investigate how well the evaluations provided by simulated users reflect the actual behaviour presented by humans.

9.2.3 Human/Agent Interfaces

The Interactive Reinforcement Learning agents demonstrated in this thesis used two types of advice, either state-based or rule-based advice. Directly providing state-based or rule-based advice may not be the most user-friendly method for humans to provide assistance. Improving the user experience for the human when interacting with the agent may improve engagement and amount of advice the user provides. While the agent needs to receive advice in a specific format, as long as an interpreter is used to transform the input, the human may provide assistance in any form.

Research and development into methods for humans to provide advice that are user-friendly and allow interactions with a high informational payload should be a priority. These improved interaction interfaces should improve engagement, decrease the interactions required to convey a lesson, and improve the learning speed of the agent.

9.2.4 Closing the Loop

In Interactive Reinforcement Learning, the focus is on the advice that the human is providing to the agent. However, in human teaching, it is well established that teaching is a two-way communication task, with the teachers and students informing each other. Interactive Reinforcement Learning differs slightly from this teacher / student model, as the RL agent can learn a better solution than the teacher initially demonstrates. Research is needed into methods for conveying behaviour learnt by the agent back to the human, so that the agent can teach the user the better behaviour. Ideally, this transfer of behaviour between the agent and the human would occur repeatedly and in both directions, making use of the agent's rapid trial-and-error learning and the humans problem solving and pattern recognition abilities. This tandem learning may help in finding optimal behaviours in shorter periods of time, and in teaching the human the optimal behaviour once found. An example of Reinforcement Learning agents teaching humans can be seen in Tesauro's TDGammon paper Tesauro explains that the agent's style of play frequently differs from traditional human

strategies, and in some cases this has led to major revisions in the positional thinking of top human players (Tesauro, 1994).

9.2.5 Multiple Users

A direction for future research is the extension of interaction Reinforcement Learning to support multiple advice-giving users. This extension would allow groups of users, each with their own areas of expertise and accuracy, to assist the agent in learning and decision making. The possibility to receive advice from a few users or a few thousand (C. Zhang & Liu, 2015; Haque, 2014) can allow the agent to rapidly receive massive amounts of advice and would address issues of limited availability of individual users. The support of multiple users introduces challenges such as the management of large amounts of conflicting advice, inaccurate advice, malicious users, and optimal advisor discovery.

9.2.6 Incorrect Advice Identification and Mitigation

Regardless of the intentions and accuracy of the advising user, at some point inaccurate advice is provided to the agent. Inaccurate advice may come directly from the user because of misunderstanding of the reward function, concept drift, or it may come simply from noise in the communication path. Experiments performed in Chapter 7 (Figure 7.6) demonstrated the effect that incorrect advice can have on an agent’s performance. While probabilistic policy reuse was found to reduce the impact of incorrect advice, research into other technologies and preprocessing should be explored.

9.3 Final Words

This thesis has identified the area of Reinforcement Learning as a valuable field of research, drawing together ideas from largely separate areas of RL research such as shaping, transfer learning, and spoken dialogue systems.

The experiments reported have shown that methods such as persistence and rule-based models can maximise the benefits of user advice, while minimising the demands on user’s time. Probabilistic Policy Reuse (PPR) has also been shown to aid in dealing with incorrect

advice. It has also been demonstrated that simulated users offer a valuable and time-saving approach to testing and developing Assisted Reinforcement Learning algorithms.

Future research in Rule-Based Interactive Reinforcement Learning aims to extend support to multiple users, and improved incorrect advice management, and to continue work on closing the loop between the human and the agent.

Bibliography

- Abbeel, P., & Ng, A. Y. (2004a). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine learning* (p. 1). ACM.
- Abbeel, P., & Ng, A. Y. (2004b). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine learning* (p. 1).
- Abdallah, S., & Kaisers, M. (2016). Addressing environment non-stationarity by repeating Q-learning updates. *The Journal of Machine Learning Research*, 17(1), 1582–1612.
- Akila, V., & Zayaraz, G. (2015). A brief survey on concept drift. In L. C. Jain, S. Patnaik, & N. Ichalkaranje (Eds.), *Intelligent computing, communication and devices: Proceedings of iccd 2014, volume 1* (pp. 293–302). New Delhi: Springer India.
- Amershi, S., Cakmak, M., Knox, W. B., & Kulesza, T. (2014). Power to the people: The role of humans in interactive machine learning. American Association for Artificial Intelligence.
- Amir, O. (2016). *Interactive teaching strategies for agent training*. New York City, USA.
- Ammar, H. B., Eaton, E., Ruvolo, P., & Taylor, M. E. (2015, January). Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*.
- Argall, B., Browning, B., & Veloso, M. (2007). Learning by demonstration with critique from a human teacher. In *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction* (p. 57-64). ACM.
- Argall, B. D., Browning, B., & Veloso, M. (2009). Automatic weight learning for multiple data sources when learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (pp. 226–231).
- Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5), 469–483.

- Banerjee, B. (2007). General game learning using knowledge transfer. *IJCAI*, 672-677.
- Barreto, A., Precup, D., & Pineau, J. (2016). Practical kernel-based reinforcement learning. *The Journal of Machine Learning Research*, 17(1), 2372–2441.
- Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 679–684.
- Bellman, R. (2013). *Dynamic programming*. Courier Corporation.
- Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 41–48). New York, NY, USA: ACM.
- Breiman, L. (2017). *Classification and regression trees*. Routledge.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.
- Brys, T. (2016). *Reinforcement learning with heuristic information* (Unpublished doctoral dissertation). Washington State University.
- Brys, T., Harutyunyan, A., Suay, H. B., Chernova, S., Taylor, M. E., & Nowé, A. (2015). Reinforcement learning from demonstration through shaping. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)* (p. 26).
- Brys, T., Nowé, A., Kudenko, D., & Taylor, M. E. (2014). Combining multiple correlated reward and shaping signals by measuring confidence. In *AAAI* (pp. 1687–1693).
- Cakmak, M., Chao, C., & Thomaz, A. L. (2010). Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2), 108–118.
- Cakmak, M., & Thomaz, A. L. (2010). Optimality of human teachers for robot learners. In *Development and Learning (ICDL), 2010 IEEE 9th International Conference on* (pp. 64–69).
- Cao, T. M., & Compton, P. (2005). A simulation framework for knowledge acquisition evaluation. In *Proceedings of the Twenty-eighth Australasian Conference on Computer Science-Volume 38* (pp. 353–360).
- Cassandra, A. R., & Kaelbling, L. P. (2016). Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings 1995: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July*

9-12 1995 (p. 362).

- Celiberto Jr, L. A., Ribeiro, C. H., Costa, A. H., & Bianchi, R. A. (2007). Heuristic reinforcement learning applied to robocup simulation agents. In *RoboCup 2007: Robot Soccer World Cup XI* (p. 220-227). Springer.
- Chen, Z., & Liu, B. (2016). *Lifelong machine learning*. Morgan & Claypool Publishers.
- Compton, P., Edwards, G., Kang, B., Lazarus, L., Malor, R., Menzies, T., ... Sammut, C. (1991). Ripple down rules: possibilities and limitations. In *Proceedings of the Sixth AAAI Knowledge Acquisition for Knowledge-Based Systems Workshop, Calgary, Canada, University of Calgary* (pp. 6–1).
- Compton, P., Peters, L., Edwards, G., & Lavers, T. G. (2006). Experience with ripple-down rules. In *Applications and Innovations in Intelligent Systems XIII* (pp. 109–121). Springer.
- Compton, P., Preston, P., & Kang, B. (1995). The use of simulated experts in evaluating knowledge acquisition. *University of Calgary*.
- Cruz, F., Twiefel, J., Magg, S., Weber, C., & Wermter, S. (2015). Interactive reinforcement learning through speech guidance in a domestic scenario. In *Neural Networks (IJCNN), 2015 International Joint Conference on* (pp. 1–8).
- Cuayáhuitl, H., Renals, S., Lemon, O., & Shimodaira, H. (2006). Reinforcement learning of dialogue strategies with hierarchical abstract machines. In *Spoken Language Technology Workshop, 2006. IEEE* (pp. 182–185).
- Dazeley, R., & Kang, B. (2004b). An online classification and prediction hybrid system for knowledge discovery in databases. In *AISAT 2004 The 2nd International Conference on Artificial Intelligence in Science and Technology* (Vol. 1, pp. 114–119).
- Dazeley, R., & Kang, B.-H. (2004a). Detecting the knowledge frontier: an error predicting knowledge based system. *Pacific Knowledge Acquisition Workshop*.
- Devlin, S., & Kudenko, D. (2011). Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1* (pp. 225–232).
- Devlin, S., & Kudenko, D. (2012). Dynamic potential-based reward shaping. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-*

Volume 1 (pp. 433–440).

- Dorigo, M., & Gambardella, L. (2014). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of ML-95, Twelfth International Conference on Machine Learning* (p. 252-260).
- Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural computation*, 12(1), 219–245.
- Efthymiadis, K., Devlin, S., & Kudenko, D. (2013). Overcoming erroneous domain knowledge in plan-based reward shaping. In *Proceedings of the 2013 International Conference on Autonomous agents and Multi-Agent Systems* (p. 1245-1246). International Foundation for Autonomous Agents and Multiagent Systems.
- Erez, T., & Smart, W. D. (2008). What does shaping mean for computational reinforcement learning? In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on* (p. 215-219). IEEE.
- Fernández, F., & Veloso, M. (2006). Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (pp. 720–727).
- Fürnkranz, J., Hüllermeier, E., Cheng, W., & Park, S.-H. (2012). Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89(1-2), 123–156.
- Gaines, B. R., & Compton, P. (1995). Induction of ripple-down rules applied to modeling large databases. *Journal of Intelligent Information Systems*, 5(3), 211–228.
- Georgila, K., Henderson, J., & Lemon, O. (2005). Learning user simulations for information state update dialogue systems. In *Ninth European Conference on Speech Communication and Technology*.
- Georgila, K., Henderson, J., & Lemon, O. (2006). User simulation for spoken dialogue systems: Learning and evaluation. In *Ninth International Conference on Spoken Language Processing*.
- Giannoccaro, I., & Pontrandolfo, P. (2002). Inventory management in supply chains: a reinforcement learning approach. *International Journal of Production Economics*, 78(2), 153–161.

- Gillespie, L. E., Gonzalez, G. R., & Schrum, J. (2017). Comparing direct and indirect encodings using both raw and hand-designed features in tetris. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 179–186).
- Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). MIT press Cambridge.
- Goschin, S., Weinstein, A., Littman, M. L., & Chastain, E. (2013). Planning in reward-rich domains via PAC bandits. In *European Workshop on Reinforcement Learning* (pp. 25–42).
- Griffith, S., Subramanian, K., Scholz, J., Isbell, C., & Thomaz, A. L. (2013). Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in Neural Information Processing Systems* (p. 2625-2633).
- Guillory, A., & Bilmes, J. A. (2011a). Online submodular set cover, ranking, and repeated active learning. In *Advances in Neural Information Processing Systems* (pp. 1107–1115).
- Guillory, A., & Bilmes, J. A. (2011b). Simultaneous learning and covering with adversarial noise. In *ICML* (Vol. 11, pp. 369–376).
- Haque, A. (2014). Twitch plays Pokemon, machine learns Twitch: unsupervised context-aware anomaly detection for identifying trolls in streaming data. *University of Texas at Austin*.
- Harutyunyan, A., Brys, T., Vrancx, P., & Nowé, A. (2015). Shaping Mario with human advice. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems* (pp. 1913–1914).
- Harutyunyan, A., Devlin, S., Vrancx, P., & Nowé, A. (2015). Expressing arbitrary reward functions as potential-based advice. In *AAAI* (pp. 2652–2658).
- Hernandez-Leal, P., Zhan, Y., Taylor, M. E., Sucar, L. E., & Munoz de Cote, E. (2016, November). Efficiently detecting switches against non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, 1–23.
- Hofmann, K., Schuth, A., Whiteson, S., & de Rijke, M. (2013). Reusing historical interaction data for faster online learning to rank for IR. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining* (pp. 183–192).

- Horowitz, S., Koepnick, B., Martin, R., Tymieniecki, A., Winburn, A. A., Cooper, S., ... others (2016). Determining crystal structures through crowdsourcing and coursework. *Nature communications*, 7, 12549.
- Howard, R. A. (1964). Dynamic programming and Markov processes.
- Isbell, C. L., Kearns, M., Kormann, D., Singh, S., & Stone, P. (2000). Cobot in LambdaMOO: A social statistics agent. In *AAAI/IAAI* (pp. 36–41).
- Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 237–285.
- Kamar, E., Hacker, S., & Horvitz, E. (2012). Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1* (pp. 467–474).
- Kang, B., Compton, P., & Preston, P. (1995). Multiple classification ripple down rules: evaluation and possibilities. In *Proceedings 9th Banff Knowledge Acquisition for Knowledge Based Systems Workshop* (Vol. 1, pp. 17–1).
- Kang, B. H., Preston, P., & Compton, P. (1998). Simulated expert evaluation of multiple classification ripple down rules. In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management*.
- Kaplan, F., Oudeyer, P.-Y., Kubinyi, E., & Miklósi, A. (2002). Robotic clicker training. *Robotics and Autonomous Systems*, 38(3), 197–206.
- Karlsson, J. (2014). *Learning to play games from multiple imperfect teachers* (Unpublished doctoral dissertation). Chalmers University of Technology.
- Khatib, F., DiMaio, F., Cooper, S., Kazmierczyk, M., Gilski, M., Krzywda, S., ... others (2011). Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology*, 18(10), 1175–1177.
- Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., & Matsubara, H. (1997). RoboCup: A challenge problem for AI. *AI magazine*, 18(1), 73.
- Knox, W. B., & Stone, P. (2008). TAMER: Training an agent manually via evaluative reinforcement. In *Development and Learning, 2008. ICDL 2008. 7th IEEE International Conference on* (pp. 292–297).
- Knox, W. B., & Stone, P. (2009). Interactively shaping agents via human reinforcement: The

- TAMER framework. In *Proceedings of the Fifth International Conference on Knowledge Capture* (pp. 9–16).
- Knox, W. B., & Stone, P. (2010). Combining manual feedback with subsequent MDP reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1* (pp. 5–12).
- Knox, W. B., & Stone, P. (2013). Learning non-myopically from human-generated reward. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces* (pp. 191–202).
- Kormushev, P., Calinon, S., & Caldwell, D. G. (2010). Robot motor skill coordination with EM-based reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on* (pp. 3232–3237).
- Krening, S., Harrison, B., Feigh, K. M., Isbell, C. L., Riedl, M., & Thomaz, A. (2017). Learning from explanations using sentiment and advice in RL. *IEEE Transactions on Cognitive and Developmental Systems*, 9(1), 44–55.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436.
- Li, G., Hung, H., Whiteson, S., & Knox, W. B. (2013). Using informative behavior to increase engagement in the TAMER framework. In *Proceedings of the 2013 International Conference on Autonomous agents and Multi-Agent Systems* (pp. 909–916).
- Liang, X., Balasingham, I., & Byun, S.-S. (2008). A reinforcement learning based routing protocol with QoS support for biomedical sensor networks. In *Applied Sciences on Biomedical and Communication Technologies, 2008. ISABEL’08. First International Symposium on* (pp. 1–5).
- Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., . . . Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994* (pp. 157–163). Elsevier.
- López, G., Quesada, L., & Guerrero, L. A. (2017). Alexa vs. Siri vs. Cortana vs. Google Assistant: a comparison of speech-based natural user interfaces. In *International Conference on Applied Human Factors and Ergonomics* (pp. 241–250).
- MacGlashan, J. (2015). *Brown-UMBC reinforcement learning and planning (BURLAP)*.

Retrieved from <http://burlap.cs.brown.edu/>

- Mannion, P., Duggan, J., & Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems* (pp. 47–66). Springer.
- Mason, K., Mannion, P., Duggan, J., & Howley, E. (2016). Applying multi-agent reinforcement learning to watershed management. In *Proceedings of the Adaptive and Learning Agents Workshop*.
- Michels, J., Saxena, A., & Ng, A. Y. (2005). High speed obstacle avoidance using monocular vision and reinforcement learning. In *Proceedings of the 22nd International Conference on Machine learning* (pp. 593–600).
- Misu, T., Georgila, K., Leuski, A., & Traum, D. (2012). Reinforcement learning of question-answering dialogue policies for virtual museum guides. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue* (pp. 84–93).
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529–533.
- Moore, A. W. (1994). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Advances in Neural Information Processing Systems* (pp. 711–718).
- Moore, A. W., Birnbaum, L., & Collins, G. (1991). Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In *Proceedings of the Eighth International Conference on Machine Learning* (pp. 333–337).
- Narvekar, S., Sinapov, J., & Stone, P. (2017, August). Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*.
- Ng, A. Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., ... Liang, E. (2006). Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics*

IX (pp. 363–372). Springer.

- Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML* (Vol. 99, p. 278-287).
- Nowé, A., Verbeeck, K., & Peeters, M. (2006). Learning automata as a basis for multi agent reinforcement learning. In *Learning and adaption in multi-agent systems* (pp. 71–85). Springer.
- Nunes, L., & Oliveira, E. (2003). Exchanging advice and learning to trust. *Cooperative Information Agents VII*, 250–265.
- Papaioannou, I., & Lemon, O. (2017). Combining chat and task-based multimodal dialogue for more engaging HRI: A scalable method using reinforcement learning. In *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction* (pp. 365–366).
- Pareigis, S. (1998). Adaptive choice of grid and time in reinforcement learning. In *Advances in Neural Information Processing Systems* (pp. 1036–1042).
- Parisotto, E., Ba, J. L., & Salakhutdinov, R. (2015). Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*.
- Peng, B., MacGlashan, J., Loftin, R., Littman, M. L., Roberts, D. L., & Taylor, M. E. (2017, May). Curriculum design for machine learners in sequential decision tasks (extended abstract). In *Proceedings of the 2017 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.
- Price, B., & Boutilier, C. (2003). Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19, 569–629.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1), 81–106.
- Randløv, J., & Alstrøm, P. (1998). Learning to drive a bicycle using reinforcement learning and shaping. In *ICML* (Vol. 98, pp. 463–471).
- Richards, D. (2009). Two decades of ripple down rules research. *The Knowledge Engineering Review*, 24(2), 159–184.
- Rieser, V., & Lemon, O. (2006). Cluster-based user simulations for learning dialogue strategies. In *Ninth International Conference on Spoken Language Processing*.
- Rojers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A survey of multi-objective

- sequential decision-making. *Journal of Artificial Intelligence Research*, 48, 67–113.
- Schaal, S. (1997). Learning from demonstration. *Advances in Neural Information Processing Systems*, 9, 1040–1046.
- Schatzmann, J., Weilhammer, K., Stuttle, M., & Young, S. (2006). A survey of statistical user simulation techniques for reinforcement learning of dialogue management strategies. *The knowledge engineering review*, 21(2), 97–126.
- Scheffler, K., & Young, S. (2001). Corpus-based dialogue simulation for automatic strategy learning and evaluation. In *Proc. NAACL Workshop on Adaptation in Dialogue Systems* (pp. 64–70).
- Scheffler, K., & Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the Second International Conference on Human Language Technology Research* (pp. 12–19).
- Sharma, M., Holmes, M. P., Santamaría, J. C., Irani, A., Isbell Jr, C. L., & Ram, A. (2007). Transfer learning in real-time strategy games using hybrid CBR/RL. In *IJCAI* (Vol. 7, pp. 1041–1046).
- Singh, S. P., & Bertsekas, D. P. (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. In *Advances in Neural Information Processing Systems* (pp. 974–980).
- Singh, S. P., & Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Recent Advances in Reinforcement Learning*, 123–158.
- Skinner, B. F. (1990). *The behavior of organisms: An experimental analysis*. BF Skinner Foundation.
- Smart, W. D., & Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on* (Vol. 4, pp. 3404–3410).
- Stone, P., Sutton, R. S., & Singh, S. (2000). Reinforcement learning for 3 versus 2 keepaway. In *Robot Soccer World Cup* (pp. 249–258).
- Suay, H. B., Brys, T., Taylor, M. E., & Chernova, S. (2016). Learning from demonstration for shaping through inverse reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (pp. 429–437).

- Suay, H. B., & Chernova, S. (2011). Effect of human guidance and state space size on interactive reinforcement learning. In *RO-MAN, 2011 IEEE* (pp. 1–6).
- Subramanian, K., Isbell Jr, C. L., & Thomaz, A. L. (2016). Exploration from demonstration for interactive reinforcement learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (pp. 447–456).
- Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems* (pp. 1038–1044).
- Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT press.
- Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems* (pp. 1057–1063).
- Sutton, R. S., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2), 181–211.
- Talvitie, E., & Singh, S. P. (2007). An experts algorithm for transfer learning. In *IJCAI* (pp. 1065–1070).
- Tan, M. (1993). Multi-agent reinforcement learning: Independent versus cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning* (pp. 330–337).
- Tanner, B., & White, A. (2009). RL-Glue: Language-independent software for reinforcement-learning experiments. *Journal of Machine Learning Research*, 10(Sep), 2133–2136.
- Tanwani, A. K., & Billard, A. (2013). Transfer in inverse reinforcement learning for multiple strategies. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on* (pp. 3244–3250).
- Taylor, M. E. (2009, March). Assisting transfer-enabled machine learning algorithms: Leveraging human knowledge for curriculum design. In *The AAAI 2009 Spring Symposium on Agents that Learn from Human Teachers*.
- Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I., & Torrey, L. (2014). Reinforcement learning agents providing advice in complex video games. *Connection Science*, 26(1),

- Taylor, M. E., Kuhlmann, G., & Stone, P. (2008). Autonomous transfer for reinforcement learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems-Volume 1* (pp. 283–290).
- Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul), 1633–1685.
- Taylor, M. E., Stone, P., & Liu, Y. (2007). Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(Sep), 2125–2167.
- Taylor, M. E., Suay, H. B., & Chernova, S. (2011). Integrating reinforcement learning with human demonstrations of varying ability. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2* (pp. 617–624).
- Taylor, M. E., Whiteson, S., & Stone, P. (2007, May). Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 156–163).
- Tenorio-Gonzalez, A. C., Morales, E. F., & Villaseñor-Pineda, L. (2010). Dynamic reward shaping: training a robot by voice. In *Advances in Artificial Intelligence-IBERAMIA 2010* (pp. 483–492). Springer.
- Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, 6(2), 215–219.
- Thimmesh, C. (2006). *Team moon*. Houghton Mifflin Company.
- Thomaz, A. L., & Breazeal, C. (2007). Asymmetric interpretations of positive and negative human feedback for a social learning agent. In *Robot and Human Interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on* (pp. 720–725).
- Thomaz, A. L., Hoffman, G., & Breazeal, C. (2005). Real-time interactive reinforcement learning for robots. In *AAAI 2005 Workshop on Human Comprehensible Machine Learning*.
- Torrey, L., & Taylor, M. E. (2013, May). Teaching on a budget: Agents advising agents in reinforcement learning. In *International Conference on Autonomous Agents and*

- Vamplew, P., Dazeley, R., Berry, A., Issabekov, R., & Dekker, E. (2011). Empirical evaluation methods for multi-objective reinforcement learning algorithms. *Machine learning*, 84(1), 51–80.
- Vlassis, N., Ghavamzadeh, M., Mannor, S., & Poupart, P. (2012). Bayesian reinforcement learning. *Reinforcement Learning*, 359–386.
- Wiering, M. A., & Van Hasselt, H. (2008). Ensemble algorithms in reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4), 930–936.
- Wiewiora, E., Cottrell, G., & Elkan, C. (2003). Principled methods for advising reinforcement learning agents. In *ICML* (pp. 792–799).
- Zhan, Y., Ammar, H. B., & Taylor, M. E. (2016, July). Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. In *Proceedings of the 25th International Conference on Artificial Intelligence (IJCAI)*.
- Zhang, C., & Liu, J. (2015). On crowdsourced interactive live streaming: a Twitch.tv-based measurement study. In *Proceedings of the 25th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video* (pp. 55–60).
- Zhang, W., & Dietterich, T. G. (1995). A reinforcement learning approach to job-shop scheduling. In *IJCAI* (Vol. 95, pp. 1114–1120).

Chapter 10

Appendix

A Ethics Application

This section contains documents submitted as part of Federation University Australia's Ethics Application B117-17A. The files attached include:

- Plain Language Information Statement
- Ethics Application Approval
- Consent Form
- Final Report
- Questionnaire

A.1 Plain English Statement

Plain Language Information Statement



SCHOOL OF SCIENCE, INFORMATION TECHNOLOGY, AND ENGINEERING FACULTY OF SCIENCE

PROJECT TITLE:	Benchmarking Human Performance on Assisting Machine Learning Agents
PRINCIPAL RESEARCHER:	Doctor Richard Dazeley
OTHER/STUDENT RESEARCHERS:	Adam Bignold, Doctor Peter Vamplew, Doctor Cameron Foale

This document is an invitation to participate in a study that aims to benchmark human performance regarding their ability to provide assistance to a machine learning agent. The document also outlines all relevant information about the study, including what data is collected, the risks involved, and the participant's rights for withdrawal.

This study is part of a larger project developing methods for people to teach machine learning agents in an interactive and iterative approach. To assist in the development of this project, the researchers aim to create 'simulated users', computer programs that can replicate the human and their interactions. The information collected in this study will be used to create the simulated users.

Human assistance and the interactions between the human and the machine learning agent have some variables that may be recorded. These include the accuracy of the advice, the type of advice provided, the frequency of interactions, the change in interactions over time, the time it takes for each interaction, and whether the advice is positive or negative. This study will record this data and all interactions between the participant and the agent.

Participation in this study will require the completion of a short experiment and questionnaire. The experiment requires the participant to watch a machine learning agent attempt to solve a simple control problem, during which the participant is given the opportunity to provide advice to the agent. The problem requires the agent to perform a series of actions in the correct order. The participant can provide assistance to the agent to help it to find a solution. The aim of the questionnaire is to assess the participant's initial knowledge about the control problem and gauge their thoughts about how interaction with the agent went. The questionnaire does not collect any personal or private information, and participants are free to choose not to answer questions on the questionnaire. It is estimated that the total involvement of the participant will take 40 minutes. At the completion of the experiment, all participants will be awarded a small confectionary treat.

This study will not collect any personally identifiable data including the participant's name, contact details, or location. Participants have the opportunity to request to preview results and questionnaire answers and to withdraw or amend (if appropriate) any data during or at the end of the participation. Participants may withdraw their consent at any point in time. Once the participant has performed the experiment and the data collected has been stored there will be no method for the participant to withdraw their data from the study as the data they provided cannot be identified. A random identifier will be placed on the participant's data so it can be known that they are about the same data subject, although the person's identity remains unknown. Arrangements have been made to keep all data collected securely. The data will be stored in a database hosted locally, will be password protected, and stored indefinitely.

CRICOS Provider No: 00103D

Benchmarking Human Advice Giving Study – PLIS 2017

Plain Language Information Statement



The data recorded from the human trials detailed in this document are intended to be used by the researchers listed in this study to design simulated users that perform the same tasks as the participants of this study and perform at a similar level. The data recorded as part of this study may be used in future publications to justify the use of the simulated users, until such a time that a publication record has been established that can act as its justification going forward. Be aware that in participating in this research, your de-identified data may be used to inform future research. This data may be subject to legal limitations (e.g., subpoena, freedom of information claim, or mandatory reporting in some professions). In the case of such requests, the confidentiality will be maintained as no names or personally identifiable information are collected.

Participants have been recruited by the researchers, by a snowball method (participants suggesting other participants), and by a widespread email to the faculty. A participant's involvement, or lack thereof, in the study will not affect any ongoing assessment, grades, employment, or management.

Occupation Overuse Syndrome (OOS), Repetitive Stress Syndrome (RSS), and eye strain can accompany computer use. Precautions will be taken to reduce the risk of such injuries. All participants will be provided with a height and back support adjustable office chair, a desk of appropriate height, a monitor that can be adjusted to eye level, and a detachable keyboard and mouse, all of which is to be situated in an office space with a low noise level and appropriate lighting and temperature. The experiment is designed to be short. If an experiment is expected to extend over an hour the participant will be required to take a break for at least 10 minutes of every hour.

All electrical equipment will be checked to ensure it has been tagged and tested for safety. Any equipment found not to be tested will not be used. All cables, cords, and equipment will be positioned as to avoid tripping hazards and to keep a clear and safe floor space.

If a participant suffers a medical injury/event, including stress, anxiety, workplace accident, or OOS/RSS, the participant will be taken to the onsite medical facility, or the relevant next party, e.g., Lifeline, partner, counsellor. At an appropriate time, a follow-up with the participant will be made, and the participant will be debriefed. If the participant completed the required data collection, then a follow-up will be made to confirm that they want their data to remain part of the project. If the participant wants to withdraw or did not complete the project, then their data will be destroyed. If a participant wishes to withdraw from the project during or after participation, their data will be destroyed, and the participant will be debriefed.

Any computer that is used for the experiment will first be checked to ensure it meets University computer security guidelines. Any computer program not required for the experiment will be closed. These measures are performed to reduce the risk of the participant interacting with any malicious software or giving out any identifiable information.

Participation in the study may only occur during the operating hours of the on-site medical facility. The opening hours of Federation University Australia's medical facility are 9:00 AM – 4:00 PM Monday through Thursday, and 9:00 AM – 1:00 PM Friday during the semester. Participation will not occur outside of these hours. Contact details for medical services, including LifeLine and Student Support, can be found at the end of this document.

Plain Language Information Statement



Participants are invited to contact the researchers if they require any further information or explanation regarding the study. The researcher's contact information can be found at the end of this document.

This study has received clearance from Federation University Australia's Human Research Ethics Committee (HREC). If participants wish to make a complaint regarding the conduct of this research they should direct these to the Ethics Officer for attention. The contact information for the Ethics Officer can be found at the end of this document.

Researchers	
Doctor Richard Dazeley (Principal)	5327 9769
<i>PhD Candidate</i> Adam Bignold	5327 9301
Doctor Peter Vamplew p.vamplew@federation.edu.au	5327 9616 p.vamplew@federation.edu.au
Doctor Cameron Foale c.foale@federation.edu.au	5327 9442 c.foale@federation.edu.au
Medical / Support	
FedUni (Mt Helen) Onsite Medical Facility	5327 9477
Ballarat Health Services/Base Hospital	5320 4000 or 5320 3718
St John of God Hospital (Ballarat)	5320 2111
East Grampians Health Service (Ararat)	5352 2221
Wimmera Health Care Group Horsham	5381 9111
Stawell District Hospital	5358 8555
LifeLine	13 11 14
FedUni Student Support (Counselling)	5327 9470 counselling@federation.edu.au
Ethics Officer	5327 9765 or 5122 6446 research.ethics@federation.edu.au

If you have any questions, or you would like further information regarding the project titled **'Benchmarking Human Performance on Assisting Machine Learning'**, please contact the Principal Researcher, Richard Dazeley of the School of Science, Information Technology, and Engineering:

EMAIL: r.dazeley@federation.edu.au

PH: 53279769

Should you (i.e. the participant) have any concerns about the ethical conduct of this research project, please contact the Federation University Ethics Officers, Research Services, Federation University Australia,
P O Box 663 Mt Helen Vic 3353 or Northways Rd, Churchill Vic 3842.
Telephone: (03) 5327 9765, (03) 5122 6446
Email: research.ethics@federation.edu.au

CRICOS Provider Number 00103D

A.2 Ethics Approval

Approval

Human Research Ethics Committee



Principal Researcher:	Dr Richard Dazeley
Other/Student Researcher/s:	Adam Bignold Dr Peter Vamplew Dr Cameron Foale
School/Section:	School of Science, Information Technology and Engineering
Project Number:	B17-117
Project Title:	Benchmarking human performance on assisting machine learning agents.
For the period:	31/08/2017 to 31/03/2018

Quote the Project No: B17-117 in all correspondence regarding this application.

Approval has been granted to undertake this project in accordance with the proposal submitted for the period listed above.

Please note: It is the responsibility of the Principal Researcher to ensure the Ethics Office is contacted immediately regarding any proposed change or any serious or unexpected adverse effect on participants during the life of this project.

In Addition: Maintaining Ethics Approval is contingent upon adherence to all Standard Conditions of Approval as listed on the final page of this notification

COMPLIANCE REPORTING DATES TO HREC:

Final project report:
30 April 2018

The combined annual/final report template is available at:

<http://federation.edu.au/research-and-innovation/research-support/ethics/human-ethics/human-ethics3>

A handwritten signature in black ink, appearing to read "Fiona Koop".

Fiona Koop
Ethics Officer
31 August 2017

Please note the standard conditions of approval on Page 2:

Approval

Human Research Ethics Committee



STANDARD CONDITIONS OF APPROVAL

1. Conduct the project strictly in accordance with the proposal submitted and granted ethics approval, including any amendments made to the proposal required by the HREC.
2. Advise (email: research.ethics@federation.edu.au) immediately of any complaints or other issues in relation to the project which may warrant review of the ethical approval of the project.
3. Where approval has been given subject to the submission of copies of documents such as letters of support or approvals from third parties, these are to be provided to the Ethics Office prior to research commencing at each relevant location.
4. Submission for approval of amendments to the approved project before implementing such changes. A combined amendment template covering the following is available on the HRE website:
<http://federation.edu.au/research/research-support/ethics/human-ethics/human-ethics3>
 - Request for Amendments
 - Request for Extension. Note: Extensions cannot be granted retrospectively.
 - Changes to Personnel
5. Annual Progress reports on the anniversary of the approval date and a Final report within a month of completion of the project are to be submitted by the due date each year for the project to have continuing approval.
6. If, for any reason, the project does not proceed or is discontinued, advise the committee by completing the Final report form.
7. Notify the Ethics Office of any changes in contact details including address, phone number and email address for any member of the research team.
8. The HREC may conduct random audits and / or require additional reports concerning the research project as part of the requirements for monitoring, as set out in the National statement on Ethical Conduct in Human Research.

Failure to comply with the *National Statement on Ethical Conduct in Human Research (2007)* and with the conditions of approval will result in suspension or withdrawal of approval.

A.3 Participant Consent Form

Consent Form

Human Research Ethics Committee



PROJECT TITLE:	Benchmarking human performance on assisting machine learning agents.
RESEARCHERS:	Richard Dazeley, Adam Bignold, Peter Vamplew, Cameron Foale,

Code number allocated to the participant:	
---	--

Consent – Please complete the following information:

I _____ of _____ +

hereby consent to participate as a subject in the above research study.

The research program in which I am being asked to participate has been explained fully to me, verbally and in writing, and any matters on which I have sought information have been answered to my satisfaction.

I understand that: all information I provide (including questionnaires) will be treated with the strictest confidence and data will be stored separately from any listing that includes my name and address.

- Aggregated results will be used for research purposes and may be reported in scientific and academic journals.
- I am free to withdraw my consent at any time during the study in which event my participation in the research study will immediately cease and information/data obtained from it will not be used.
- I understand the exception to this is if I withdraw after information has been aggregated - it is unable to be individually identified - so from this point it is not possible to withdraw my information/data, although I may still withdraw my consent to participate.
- I understand that by participating in this research project the de-identified data provide may be used to inform future research.

SIGNATURE: _____ **DATE:** _____

A.4 Ethics Final Report

Annual/Final Project Report

Human Research Ethics Committee



Please indicate the type of report	<input type="checkbox"/> Annual Report (Omit 3b & 5b) <input checked="" type="checkbox"/> Final Report
Project No:	B17-117 A
Project Name:	Benchmarking human performance on assisting machine learning agents.
Principal Researcher:	Dr Richard Dazeley
Other Researchers:	Adam Bignold, Dr Peter Vamplew, Dr Cameron Foale
Date of Original Approval:	
School / Section:	School of Science, Information Technology, and Engineering
Phone:	03 5327 9000
Email:	a.bignold@federation.edu.au

Please note: For HDR candidates, this Ethics annual report is a separate requirement, in addition to your HDR Candidature annual report, which is submitted mid-year to research.degrees@federation.edu.au.

1) Please indicate the current status of the project:				
1a) Yet to start			<input type="checkbox"/>	
1b) Continuing			<input type="checkbox"/>	
1c) Data collection completed			<input checked="" type="checkbox"/>	
1d) Abandoned / Withdrawn:			<input type="checkbox"/>	
1e) If the approval was subject to certain conditions, have these conditions been met? (If not, please give details in the comments box below)			<input type="checkbox"/> Yes	<input checked="" type="checkbox"/> No
Comments:				
1f) Data Analysis		<input type="checkbox"/> Not yet commenced	<input checked="" type="checkbox"/> Proceeding	<input type="checkbox"/> Complete <input type="checkbox"/> None
1g) Have ethical problems been encountered in any of the following areas:				
Study Design			<input type="checkbox"/> Yes	<input checked="" type="checkbox"/> No
Recruitment of Subjects			<input type="checkbox"/> Yes	<input checked="" type="checkbox"/> No

Annual/Final Project Report

Human Research Ethics Committee



Finance Facilities, Equipment (If yes, please give details in the comments box below)	<input type="checkbox"/> Yes <input type="checkbox"/> Yes	<input checked="" type="checkbox"/> No <input checked="" type="checkbox"/> No
Comments: 		

2a) Have amendments been made to the originally approved project?	
<input checked="" type="checkbox"/> No	<input type="checkbox"/> Yes
2b) If yes, was HREC approval granted for these changes?	
<input type="checkbox"/> Yes	Provide detail: <input type="checkbox"/> Yes Application for Amendment to an Existing Project <input type="checkbox"/> Yes Change of Personnel <input type="checkbox"/> Yes Extension Request
<input type="checkbox"/> No	If you have made changes, but not had HREC approval, provide detail as to why this has not yet occurred:
2c) Do you need to submit any amendments now?	
<input checked="" type="checkbox"/> No	<input type="checkbox"/> Yes Application for Amendment to an Existing Project <input type="checkbox"/> Yes Change of Personnel <input type="checkbox"/> Yes Extension Request * NB: If 'Yes', download & submit the appropriate request to the HREC for approval: Please note: Extensions will not be granted retrospectively. Apply well prior to the project end date, to ensure continuity of HRE approval.

3a) Please indicate where you are storing the data collected during the course of this project: (Australian code for the Responsible conduct of Research Ch 2.2.2, 2.5 – 2.7)
The data is stored in password-protected compressed files, and in a password protected database. The database can only be accessed from the computer that the database is hosted on. The database and the compressed files are stored on a password-protected computer, with encryption. The computer is kept primarily on university premises, behind locked doors. When the computer is taken of university premises, the computer remains password protected, encrypted, and stored behind locked doors.
3b) Final Reports: Advise when & how stored data will be destroyed (Australian code for the Responsible conduct of Research Ch 2.1.1)

Annual/Final Project Report

Human Research Ethics Committee



The stored data, which is wholly non-identifiable, will be kept until the day in which it serves no further purpose to the current research topic, or until the findings drawn from the data have been peer-reviewed and published.

The data, and all copies and backups, will be destroyed 5 years after the final publication using the data has been published, or until all challenges to the published research have concluded, whichever comes last. In the meantime, the data will remain password-protected and non-identifiable and will continue to follow the universities security guidelines.

The data will be destroyed by removing all files, copies, and backups, from the machines it is stored on. Any physical devices containing the data, (USB, CD, Paper) will either have the data removed (where possible), or be shredded using the university facilities and disposal services.

4) Have there been any events that might have had an adverse effect on the research participants OR unforeseen events that might affect continued ethical acceptability of the project?

☒ No

☐ Yes * NB: If 'yes', please provide details in the comments box below:

Comments:

5a) Please provide a short summary of results of the project so far (no attachments please):

Collection and processing of the data has been completed.

Participants performed one of two experiments. The experiments involved assisting an agent in completing a task by providing small discrete advice. The two experiments differed on the type of advice given to the agent, either providing a critique of the agents past action (Right or Wrong), or telling the agent which action to take next (Left or Right). We label these experiments/advice methods evaluative and informative respectively.

Preliminary results show that the participants who provided informative advice were more accurate in the advice they gave and provided advice for a longer period then the participants who gave evaluative advice. It was also found that agents that were provided informative advice performed better than agents provided evaluative advice.

5b) Final Reports: Provide details about how the aims of the project, as stated in the application for approval, were achieved (or not achieved). (Australian code for the Responsible conduct of Research 4.4.1)

Annual/Final Project Report

Human Research Ethics Committee



Aims of the experiment:

- **Gather data about human interactions, accuracy, availability, and latency, to be used to build simulated humans.**

Successful. Data regarding human interactions, accuracy, availability, and latency was collected. This data clearly shows the differences between the two advice giving methods tested. Using the data gathered, a simulation of the average user for each of the experiments was created, and the performance of the simulated users was closely matched to that of the average of the real users.

- Gather data used to compare different reinforcement learning methods that utilise human advice

Successful. Sufficient data for both reinforcement learning / advice giving methods was collected to make an accurate comparison between the two. For a summary of the findings see section 5a.


6) Publications: Provide details of research dissemination outcomes for the previous year resulting from this project: eg: Community seminars; Conference attendance; Government reports and/or research publications

The data collected in this research project are to be used for Adam Bignold's PhD and papers to be published alongside the PhD. While drafts of papers have been completed, none have been published. The date of publication/submission for the PhD will be in early June.

7) The HREC welcomes any feedback on:

- Difficulties experienced with carrying out the research project; or
- Appropriate suggestions which might lead to improvements in ethical clearance and monitoring of research.




8) Signatures

Principal Researcher:	 Print name: Richard Dazeley	Date:	31/03/2018
------------------------------	---	--------------	------------

Annual/Final Project Report

Human Research Ethics Committee



Other/Student Researchers:	 Print name: Peter Vamplew	Date:	31/03/2018
	 Print name: Cameron Foale	Date:	31/03/2018
 Print name: Adam Bignold	Date:	31/03/2018	

Submit to the Ethics Officer, Mt Helen campus, by the due date:
research.ethics@federation.edu.au

A.5 Questionnaire

Questionnaire Form



Participant Code: _____

No identifying information is collected. The participant code is used to match your questionnaire responses to your experiment responses. After completing, neither your name nor any identifying information will be kept. See the Plain Language Information Statement for more details.

Have you participated in a machine learning study in the past?

On a scale of 0 – 10, how would you rate your level of knowledge about the Mountain Car experiment?

0 (Nothing) 1 2 3 4 5 6 7 8 9
10 (Expert)

Stop the questionnaire now and perform the experiment. After completing the experiment, turn over the page and complete the questions.

Questionnaire Form



Do not complete this side until you have completed the experiment.

Now that you have completed the experiment, on a scale of 1 – 10, how would you rate your level of knowledge about the Mountain Car experiment?

0 (Nothing) 1 2 3 4 5 6 7 8 9 10 (Expert)

How do you feel about the level of engagement you had with the agent?

- A. I could have spent more time interacting with the agent.
- B. I'm happy with how much time I interacted with the agent.
- C. I spent too much time interacting with the agent.

How accurate do you feel your advice was to the agent?

- A. Always Incorrect
- B. Mostly Incorrect
- C. Sometimes Incorrect
- D. Sometimes Correct
- E. Mostly Correct
- F. Always Correct.

How well do you think the agent followed your advice?

0 (Never) 1 2 3 4 5 6 7 8 9 10 (Always)

Is there any other information you want to provide?

”Persistence is very important. You should not give up unless you are forced to give up.”

– Elon Musk