Bayesian Methods for Robust and Active Inverse Reinforcement Learning



Ondrej Bajgar

Lady Margaret Hall

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2025

Acknowledgements

Personal

I would like to thank my supervisors, Mike Osborne, Alessandro Abate, and Konstantinos Gatsis for their unwavering support throughout my DPhil journey, for all the inspiring discussions, constructive feedback, as well as their kindness and patience. Thanks certainly belong also to Dylan Hadfield-Menell and Yangchen Pan for kindly serving as my DPhil examiners, and to Wendy Poole, the AIMS CDT Administrator, for her great energy and immense willingness to help. I would also like to thank the Future of Humanity Institute and Trajan House for creating an inspiring environment in which to incubate ideas and work on this project. Finally, I would like to thank my wife Kristyna for her love and support, for following me to Oxford for 6 years and creating a loving environment in which to recover after every difficult research day.

Institutional

I would also like to acknowledge financial support from Deepmind, the Engineering and Physical Sciences Research Council (EPSRC; through grant EP/S024050/1), the Long-Term Future Fund, and the Future of Humanity Institute. I would also like to thank FAR.AI and the Center for Human-Compatible AI at UC Berkeley for hosting me as a research visitor during my DPhil.

Abstract

For AI systems to be both helpful and safe, they require a clear specification of human preferences and values, and as these systems grow more autonomous, such specifications must be increasingly robust to avoid harmful unintended consequences. Reward functions offer one formalism for representing human preferences, but hand-designing them is notoriously difficult. Inferring rewards from human behaviour offers an alternative, though any finite set of demonstrations leaves the true reward underdetermined. Bayesian inverse reinforcement learning (IRL) addresses this by quantifying uncertainty over the inferred reward, enabling robust policies and principled identification of where additional human input would be most valuable.

However, existing Bayesian IRL methods remain largely impractical. Approaches that meaningfully capture posterior uncertainty are computationally intractable beyond small discrete environments, and all methods demand numerous expert demonstrations that are costly to obtain in terms of human labour.

This thesis addresses both barriers. To address the computational bottleneck, we introduce Q-centric inference: performing Bayesian inference over Q-values rather than rewards to avoid the computational burden of repeated forward planning, while exploiting the inverse Bellman equation to also recover the implied reward posterior. ValueWalk implements this via Hamiltonian Monte Carlo, while Q-based Variational IRL (QVIRL) uses variational inference to scale to higher-dimensional continuous domains including learning from raw-pixel observations in ATARI games. To address the scarcity of human demonstrations, we develop information-theoretic active IRL methods that strategically select maximally informative demonstration scenarios, providing the first probably-approximately-correct (PAC) guarantees for active IRL with imperfect demonstrations.

Contents

List of Figures **viii**

1 Introduction **1**
- 1.1 Why Bayesian IRL? . . . 3
- 1.2 Challenges . . . 4
- 1.3 Contributions of My Doctoral Project . . . 5
- 1.4 Limitations . . . 8
- 1.5 Structure of the Thesis . . . 10

2 Background **12**
- 2.1 The Specification Problem . . . 12
  - 2.1.1 Rewards as Representations of Preferences . . . 13
  - 2.1.2 Risks of Reward Misspecification . . . 14
  - 2.1.3 Approaches to the Specification Problem . . . 17
- 2.2 Imitation Learning . . . 19
- 2.3 Inverse Reinforcement Learning . . . 20
  - 2.3.1 Underdeterminacy . . . 21
  - 2.3.2 Computational Challenges . . . 22
  - 2.3.3 Extensions of IRL . . . 23
  - 2.3.4 Methods with Bayesian Elements . . . 24
- 2.4 Bayesian Inverse Reinforcement Learning . . . 25

3 Walking the Values in Bayesian Inverse Reinforcement Learning **28**
- 3.1 Introduction . . . 30
- 3.2 Background . . . 32
  - 3.2.1 Bayesian Inverse Reinforcement Learning . . . 32
  - 3.2.2 Markov Chain Monte Carlo (MCMC) . . . 33
  - 3.2.3 Related Work . . . 34
- 3.3 Method . . . 35
  - 3.3.1 Finite State and Action Spaces . . . 36
  - 3.3.2 Continuous State Representations . . . 39
  - 3.3.3 Continuous Actions . . . 41

3.4 Experiments . . . 41
3.4.1 Gridworld . . . 41
3.4.2 Classic Control Environments . . . 44
3.5 Discussion . . . 46

4 Q-based Variational Inverse Reinforcement Learning 48
4.1 Introduction . . . 50
4.2 Problem Formulation . . . 51
4.3 Related Work . . . 53
4.4 Method . . . 54
4.4.1 Variational Posterior over Optimal Q-Values . . . 55
4.4.2 From Optimal Q-Values to Rewards . . . 56
4.4.3 Likelihood . . . 58
4.4.4 Training Objective . . . 58
4.5 Experiments . . . 59
4.5.1 Evaluation of Posterior Approximation Quality . . . 60
4.5.2 Apprenticeship Learning . . . 61
4.5.3 Active Learning . . . 65
4.6 Discussion . . . 67
4.7 Conclusion . . . 67

5 PAC Apprenticeship Learning with Active IRL 69
5.1 Introduction . . . 71
5.2 Task Formulation . . . 72
5.3 Related Work . . . 76
5.3.1 Active Learning with Single Action Annotations . . . 76
5.3.2 Active Learning with Full Trajectories . . . 77
5.3.3 Other Settings . . . 78
5.3.4 Connection to General Active Learning . . . 78
5.4 Method . . . 79
5.4.1 PAC-EIG: Information Gain for Reliable Policies . . . 79
5.4.2 Computing PAC-EIG . . . 81
5.4.3 Reward EIG: When Learning the Reward is the Goal . . . 81
5.5 Producing a PAC Policy . . . 82
5.6 Experiments . . . 84
5.6.1 Results . . . 85
5.7 Discussion and Conclusion . . . 86

6 Conclusion 89

Appendices

A Appendix to Chapter 3 (ValueWalk) 93
A.1 Unknown Transition Probabilities . . . 93
A.2 Proof of Soundness of the Algorithm . . . 94
A.3 Variations of the Baseline Methods . . . 96
A.3.1 PolicyWalk-HMC . . . 96
A.3.2 Model-Based AVRIL . . . 96
A.4 Experiment Details . . . 97
A.5 Additional Details of Results . . . 98
A.5.1 Gridworld Experiments . . . 98

B Appendix to Chapter 4 (QVIRL) 100
B.1 Method Details . . . 101
B.1.1 Evaluating Expectations over Next States . . . 101
B.1.2 Clark Approximation to the State-Value Distribution . . . 102
B.1.3 KL and the Loss . . . 108
B.1.4 State-Only Reward Variant . . . 109
B.2 Approximations: Further Notes and Illustration . . . 110
B.2.1 Max-Mean Approximation . . . 110
B.2.2 Ablation: State-Value Approximation and KL Term . . . 111
B.2.3 Likelihood Approximation . . . 113
B.2.4 Gaussianity . . . 114
B.3 Additional Discussion on AVRIL . . . 115
B.3.1 Dynamics-Aware AVRIL . . . 115
B.3.2 3×3 Gridworld and a Comparison to AVRIL . . . 116
B.3.3 Limitations of AVRIL . . . 117
B.4 Experiment Details . . . 119
B.4.1 Environment Descriptions . . . 119
B.4.2 Expert Demonstrations . . . 121
B.4.3 QVIRL Training . . . 122
B.4.4 Baselines on ATARI . . . 123
B.4.5 Evaluation on ATARI . . . 124
B.4.6 Active Learning . . . 124
B.4.7 Prior Distribution . . . 125
B.5 Additional Experiments . . . 127
B.5.1 Sensitivity of QVIRL to Rationality Coefficient β . . . 127
B.5.2 Illustrative Gridworld Experiment with a Single Unknown State 128
B.6 Code, Data, and Compute . . . 129
B.6.1 Code and Data . . . 129
B.6.2 Computing Resources Used . . . 130

C Appendix to Chapter 5 (PAC Active IRL) 131
C.1 Failure Modes of Prior Methods . . . 131
C.2 Theoretical Analysis . . . 135
C.2.1 Notes on Possible Improvements . . . 145
C.3 Accounting for Visitation Frequencies . . . 145
C.4 Alternative Acquisition Functions . . . 147
C.4.1 From Apprentice Return to Regret . . . 147
C.4.2 The Challenge of Direct Regret Optimization . . . 147
C.4.3 Information Gain About Regret . . . 148
C.4.4 Immediate Regret EIG . . . 149
C.4.5 Discretization and Connection to PAC-EIG . . . 149
C.4.6 Summary . . . 150
C.5 Experiment Details . . . 150
C.5.1 Basic Parameter Values . . . 150
C.5.2 Environments . . . 151
C.5.3 Bayesian IRL Methods . . . 151
C.5.4 Metrics . . . 152
C.5.5 Implementation . . . 152
C.5.6 Timing . . . 153
C.6 Notation Overview . . . 153

D Negative Human Rights as a Basis for AI Safety and Regulation 155

References 158

List of Figures

3.1 Comparison of our ValueWalk algorithm to AVRIL on an illustrative 3x3 gridworld . . . 41
3.2 Results on classic control environments . . . 44
3.3 Log likelihood and entropy of ValueWalk and AVRIL on classic control environments . . . 45

4.1 Diagram of the QVIRL architecture . . . 55
4.2 Posteriors recovered by ValueWalk, AVRIL, and QVIRL in randomized gridworlds . . . 61
4.3 QVIRL apprenticeship learning: episode return in Lunar Lander and Highway . . . 63
4.4 QVIRL apprenticeship learning: held-out action log-likelihood in Lunar Lander and Highway . . . 64
4.5 Active learning results with QVIRL . . . 66

5.1 Illustration of the active IRL task. . . . 73
5.2 Results on the feature-based jail gridworld. . . . 86
5.3 Results with single state annotations on the 10x10 gridworld. . . . 87
5.4 Results of the experiments with single state annotations (i.e. $|\tau| = 1$) on the 8x8 fully random gridworld. . . . 87
5.5 Results of the experiments with expert trajectories of maximum length $|\tau| = 5$ on the 8x8 fully random gridworld. . . . 88

A.1 Grid of 2D histograms representing pairwise joint posteriors of ValueWalk in a gridworld . . . 99

B.1 Illustration of the effects of our approximation to the maximum of random variables . . . 111
B.2 Ablation of QVIRL in Lunar Lander apprenticeship learning . . . 112
B.3 Illustration of the effects of our likelihood approximation . . . 114
B.4 QVIRL and AVRIL comparison on a 3x3 gridworld . . . 116
B.5 The environments used in the QVIRL experiments . . . 119
B.6 Sensitivity of QVIRL to the rationality coefficient . . . 128

B.7 Illustrative gridworld experiment with a single unknown state . . . 128

C.1 Two-state environment designed to illustrate a failure mode of Lopes et al. [160] . . . 132
C.2 Two state environment to demonstrate a failure mode of Brown et al. [161]. . . . 134

1 Introduction

Contents

1.1 Why Bayesian IRL? 3
1.2 Challenges 4
1.3 Contributions of My Doctoral Project 5
1.4 Limitations 8
1.5 Structure of the Thesis 10

Artificial intelligence (AI) systems capable of autonomous decision-making offer huge potential to enhance human capabilities across domains ranging from everyday tasks to complex scientific research or even governance. A defining characteristic that distinguishes advanced AI from conventional tools is its ability to independently discover novel solutions to achieve specified objectives. However, this autonomy introduces an important challenge: the necessity to precisely specify what constitutes desirable versus undesirable outcomes. This *specification* problem – the translation of human intentions and values into computationally tractable objective functions – represents one of the central challenges in AI alignment.

For some well-structured tasks, reward functions can be specified by hand, enabling direct application of reinforcement learning or automated planning. For instance, games like Chess or Go have clear win/loss conditions that can be formalized; simple robotic control tasks such as pendulum stabilization have

mathematically defined target states; and certain manufacturing processes feature well-defined quality metrics that can serve as rewards. However, in many real-world contexts, appropriate reward functions are not available in closed form. In these cases, one might consider having humans directly provide scalar reward signals as feedback. Content recommendation systems, for example, might rely on explicit user ratings; image generation systems might use human evaluations of aesthetic quality; and household robots might receive human judgment on task completion. Even if, for the moment, we set aside problems like the system intervening in the provision of reward [1], this approach is labour-intensive in terms of human supervision, which limits scalability and practicality for many tasks. Moreover, condensing the entirety of their preferences into a single numerical rating covering all consequences of the AI system's actions is extremely challenging, can feel unnatural to humans, and will generally constitute only an imperfect proxy for their true preferences.

The stakes of this specification problem extend well beyond suboptimal task performance. Misspecified reward functions can lead agents to pursue proxy goals that superficially resemble the intended objective but diverge catastrophically under optimization – a phenomenon sometimes called reward hacking or specification gaming [2, 3]. As AI systems become more capable, this divergence could manifest not merely as amusing failures but as irreversible harms. Russell [4] argues that an advanced AI system optimizing even a slightly incorrect objective might resist correction or shutdown if doing so would interfere with achieving its assigned goal. Similarly, Ngo et al. [5] identify reward misspecification as a key pathway to catastrophic outcomes: a capable system pursuing the wrong objective could acquire resources, resist modification, or take actions whose consequences humans cannot undo. These considerations motivate treating robust reward learning not as a mere engineering convenience but as a safety-critical problem.

These challenges motivate an alternative approach: learning reward functions from demonstrations rather than having humans specify them explicitly. Inverse reinforcement learning (IRL) addresses the specification problem by inferring the underlying reward function that best explains observed behaviour. This approach

aligns with how humans naturally teach and learn from each other – through demonstration rather than explicit numerical feedback. When we observe an expert performing a task, we can often intuitively infer their objectives and can then generalize this understanding to new situations. Similarly, IRL algorithms observe demonstrations of desired behaviour and recover a reward function that makes this behaviour appear optimal or near-optimal. This methodology resonates with Stuart Russell's principles for beneficial AI systems [4] (p.173): the machine's only objective should be to maximize the realization of human preferences; it should be initially uncertain about what these preferences are; and the ultimate source of information about these preferences should be human behaviour.

By recovering the reward function representing the underlying human values and preferences rather than merely imitating behaviour, IRL aims to provide better generalization to new situations and thus a more robust basis for autonomous decision-making. Although achieving this benefit in practice depends on the quality and diversity of the demonstrations and the chosen model class, the hope is for the inferred reward function to serve as a compact, transferable representation of human intentions that can guide AI systems even in contexts not covered by the original demonstrations.

1.1 Why Bayesian IRL?

Most methods for IRL recover a single 'best guess' point estimate of the expert's reward function. However, especially for safety-critical systems, learning just one guess can be risky. Even a slightly inaccurate reward may lead to unintended behaviour that goes against the true reward. Such a risk is particularly high in parts of the state space where the demonstrations offer limited coverage.

Bayesian approaches to IRL aim to directly model uncertainty about the reward. Instead of picking a single function, they maintain a posterior distribution over all plausible reward functions, given prior information and the demonstrations observed so far. In that sense, Bayesian IRL addresses the inherent underdeterminacy of the IRL problem – a key challenge in the field stemming from the fact that any

given set of behaviour trajectories can be explained using many different reward functions, as we will discuss later.

Modelling uncertainty over the reward function offers two main advantages. Firstly, it allows us to produce risk-averse policies. A point estimate of the reward function may be wrong, so an apprentice policy optimizing it may cause harm with respect to the true reward (and the preferences or values it represents). If we maintain a posterior distribution over a range of reward functions compatible with the prior and the available evidence, we may be able to produce a policy robust with respect to this uncertainty, i.e. one that performs reasonably well with high probability under the posterior distribution over rewards. Such a policy may not exist yet, since we may not yet have enough data to determine with confidence that any particular policy is robustly good or safe, but that brings us to the second advantage of modelling uncertainty.

Given a current posterior over rewards, Bayesian *active* IRL enables targeted queries to human experts (for instance in the form of an initial state of the environment in which they should perform their next demonstration), efficiently reducing uncertainty where it most impacts the performance of the apprentice policy or where it allows us to learn the most about the reward. This has the potential to reduce the amount of human labour required for producing demonstrations – which may be an important bottleneck in applying IRL – as well as to obtain targeted information that maximally helps the apprentice to become both helpful and safe.

1.2 Challenges

Nonetheless, Bayesian IRL faces significant computational hurdles. A straightforward implementation often requires repeatedly solving the forward reinforcement learning problem – mapping a reward function to the corresponding optimal policy and the associated Q-values – within the inner loop of inference. In high-dimensional or continuous-state problems, this becomes intractable. As a result, Bayesian IRL methods described in early foundational work were mostly tested on small, discrete environments. Substantial improvements are needed to make Bayesian

IRL applicable to more realistic or larger-scale tasks while maintaining meaningful estimation of posterior uncertainty. This applies to both the risk aversion and active learning applications of Bayesian IRL.

Concerning *active* IRL in particular, while there exist a few acquisition functions for actively gathering demonstrations, none of them provides theoretical results that would guarantee eventually producing a reliable policy. In fact, for some of these prior acquisition functions, we show cases where they can completely prevent learning. Both these hurdles stand in the way of using active IRL to address the higher-level challenge of making IRL more sample-efficient and robust.

1.3 Contributions of My Doctoral Project

While the main focus of this thesis is technical, motivated by the need to equip AI systems with robust safety constraints, I started my doctoral work by examining what values could form the basis of these safety constraints, how they could be learnt, and how they could interact with possible legal protections, without which safety from AI risks is impossible to achieve. This resulted in the paper *Negative Human Rights as a Basis for Long-Term AI Safety and Regulation* [6] published in the *Journal of Artificial Intelligence Research* together with the legal scholar Jan Horenovsky. Since the article's topic relates to the motivation for the rest of the thesis, I provide a short summary of the argument in Appendix D; however, because it is a very different kind of paper from the rest, building on a different body of prior work, I decided to exclude it from the core thesis and provide it only as a supplement for readers who want additional context or are interested in the argument.

In the rest of this thesis, I present three main contributions toward more robust and scalable Bayesian IRL, designed to address the challenges outlined in the previous section and thus push beyond some of the computational or methodological barriers seen in earlier work:

(1) ValueWalk. Most Bayesian IRL methods naturally treat the reward parameters as the primary parameters to be sampled or optimized. However, the likelihood – i.e. how likely the expert trajectories are given a reward – is often expressed in terms of the Q-values, since these more directly express how good an action is for the expert to take. The associated reward-to-Q-value computations – the forward planning problem – can be prohibitively expensive, especially if they need to be done repeatedly for each considered hypothetical reward function. ValueWalk addresses this challenge by shifting focus to Q-values themselves and performing Bayesian sampling primarily in the Q-value space. By mapping Q-values back to rewards (instead of the other way around), ValueWalk avoids repeatedly solving the forward RL problem and thus achieves notable efficiency gains. In addition, this Q-centric view allows the use of gradient-based sampling methods like Hamiltonian Monte Carlo, further improving performance.

This paper was published under the title *Walking the Values in Bayesian Inverse Reinforcement Learning* [7] as a spotlight paper at UAI 2024 and was co-authored with my three supervisors. I authored the main ideas, performed the experimental work and did most of the writing, while my supervisors provided guidance and feedback throughout the process and helped me structure my work and drive it to conclusion.

(2) Q-based Variational IRL (QVIRL). Variational methods provide an appealing way to tackle Bayesian inference in larger or higher-dimensional environments that MCMC methods like ValueWalk struggle with, trading some fidelity in modelling the posterior for speed and scalability. Our Q-based Variational IRL (QVIRL) combines the Q-value focus with a variational inference framework. This yields a posterior distribution over Q-values (and thereby over rewards), but without exhaustive MCMC sampling. It opens the door to applying Bayesian IRL in higher-dimensional continuous environments, while still retaining uncertainty quantification – a key feature for robust downstream decision-making. We demonstrate the potential of this method on randomized gridworlds, Lunar Lander, the Highway environment,

and two ATARI games played from raw pixels, and we show that it can serve as a scalable backbone for active IRL. The split of contributions between me and my supervisors is the same as for the previous paper. In addition, Peter Tisnikar joined at the end of the project, provided feedback on drafts of the manuscript, helped edit it for submission to the *Reinforcement Learning Journal*, and ran the experiments on mismatch in the rationality coefficient. The paper was published in the *Reinforcement Learning Journal* [8] and presented orally at the Reinforcement Learning Conference (RLC) 2026.

(3) Active Bayesian IRL. Finally, I propose an active Bayesian IRL framework designed to reduce the cost and improve the efficiency of acquiring expert demonstrations. Specifically, this includes: (i) introducing novel information-theoretic acquisition functions for actively selecting demonstration queries; (ii) demonstrating empirically that these new strategies significantly outperform existing heuristic approaches; and (iii) establishing theoretical guarantees in a probably-approximately-correct (PAC) sense for the resulting policies, even when demonstrations are imperfect.

A shorter version of this work was published at the *Workshop on Bayesian Decision-making and Uncertainty* at NeurIPS 2024 under the title *Toward Information-theoretic Active Inverse Reinforcement Learning* [9] and the longer version forming the basis of Chapter 5 was published in the *Reinforcement Learning Journal* under the title *PAC Apprenticeship Learning with Active Inverse Reinforcement Learning* [10] and presented orally at RLC 2025. The paper was a collaboration with several other researchers in Oxford, whom I originally started mentoring as part of OxAI Safety Hub Labs, a programme for junior researchers looking to try AI safety research. My contribution consisted of proposing the project and the solution pathway as well as writing most of the code, assembling the proof, and writing most of the manuscript. Dewi S. W. Gould authored the implementation of reward EIG and ran preliminary experiments with it. He also wrote the appendix on the limitations of each of the prior methods, and participated in collaborative sessions

exploring initial ideas for the proof. Jonathon Liu also contributed pieces of code and produced some of the plots in the paper and the associated descriptions. Both participated in collaborative discussions throughout the research process. Alessandro Abate, Konstantinos Gatsis, and Michael Osborne offered feedback and guidance.

An important point to note is that while our methods centre inference in the space of Q-values, the reward posterior can still be cheaply recovered in most of the considered settings, and the methods thus constitute valid algorithms for classic, reward-focused inference with all the advantages that learning about the reward and the associated uncertainty brings.

By integrating these three strands – more efficient inference via Q-values, scalable approximation via variational methods, and uncertainty-guided active demonstration queries – this thesis takes several steps toward making Bayesian IRL more practical and robust across a range of tasks, including safety-critical ones.

1.4 Limitations

The thesis still presents only partial solutions to the mentioned challenges. Firstly, it works with the assumption that human preferences or values can be modelled using a static reward function in a Markov decision process (MDP), which is certainly simplistic. For instance, Skalse and Abate [11] outline multiple cases that such rewards fail to capture: multi-objective settings (where the agent optimizes for a non-linear combination of multiple reward functions), risk aversion, or what they call *modal tasks*, which work with the notion of *possibility* (e.g. one should always be able to return to the starting state). Also, human preferences can change over time and in particular can be manipulated by the actions of the AI system that aims to learn and fulfil them [12]. In real-world settings, our simplistic model will need refinement; nevertheless, modelling uncertainty will remain crucial even when adopting more sophisticated models of human preferences.

Secondly, while this thesis investigates Bayesian inference techniques within the classical inverse reinforcement learning framework, real-world human-AI interaction often requires a shift in perspective from learning reward functions solely to guide

autonomous agent behaviour towards explicitly assisting humans in achieving the *humans'* goals. Assistance games, or *cooperative inverse reinforcement learning* (CIRL) introduced by Hadfield-Menell et al. [13], capture the scenario where an AI system actively cooperates with a human who may not always behave optimally and whose preferences must be inferred jointly with cooperative actions. A significant conceptual limitation of standard IRL is precisely its assumption of optimal or near-optimal demonstrations from a first-person viewpoint of a single agent – which could translate into teaching a robot to mimic a human drinking coffee – rather than enabling an agent to support or assist a human in their environment according to human preferences – i.e. safely serving the coffee to the human. Nevertheless, since the assistance paradigm also relies on modelling human reward functions, and uncertainty in the inferred reward still needs to be taken into account, the Bayesian inference methods developed in this thesis offer a foundation for extending and improving scalability and robustness within assistance-focused IRL frameworks as well.

Thirdly, the experimental evaluations presented throughout this thesis predominantly use simplified simulated settings, ranging from discrete gridworlds to simulated control and driving tasks and ATARI games played from raw pixels. While these controlled scenarios offer clear benchmarks for methodological development and comparative analysis, they fall short of capturing the full complexity and scale of real-world problems. With our work, we hope to make contributions toward scaling – and we think our methods, especially QVIRL, could be pushed to substantially larger problems with some engineering effort – but the step toward larger scale and real-world problems still needs to be made.

Further limitations related to each constituent paper are discussed therein, and I discuss some directions for mitigating the limitations and extending this work in the final chapter.

1.5 Structure of the Thesis

This is a thesis in an 'integrated format', i.e. one whose main chapters are in the form of full research papers as published. That means there is some duplication in the Introduction and Background chapters of the thesis and papers' introduction and task description sections, so readers are encouraged to skim or skip these parts as feels appropriate. In the next chapter, I introduce some background on which the rest of the thesis builds and review selected related work. Next, in Chapter 3, I introduce the ValueWalk algorithm in detail, describing how its Q-value-centric approach can make MCMC sampling from the reward posterior more efficient. Then I discuss experimental evaluation on gridworlds as well as continuous-state simulated control environments. Subsequently, in Chapter 4, I turn to QVIRL, showing how variational inference can help scale Bayesian IRL to higher-dimensional environments than what is possible with ValueWalk. Empirical results illustrate the quality of QVIRL's posterior in randomized gridworlds, its apprenticeship learning performance on tasks up to ATARI games played from raw pixels, and its use as a backbone for active IRL. In Chapter 5, I present my work on active Bayesian IRL, detailing information-theoretic query-selection methods and offering experimental results on how these strategies outperform prior heuristic approaches. I also outline the first PAC-style result for a Bayesian active IRL algorithm that can incorporate imperfect demonstrations. The final chapter concludes and offers a brief discussion of possible future work in this area. This is followed by the appendices to the three main chapters, which contain proofs of theoretical results in each paper, a discussion of some of the assumptions made, extensions to more general settings, and other additional details. The final appendix summarizes my work concerning negative human rights as values on which robust safety specification and regulation could be based – work which I performed during my doctoral studies, but which is quite distinct from the other three papers, so its full inclusion may have disrupted the thesis' coherence.

Overall, by combining methodological innovation and empirical analysis, this thesis aims to demonstrate how Bayesian IRL can be made more tractable and

effective. Its potential to capture and quantify uncertainty in learnt reward functions, particularly using limited and targeted demonstrations, may have value for designing more trustworthy AI systems.

If we use, to achieve our purposes, a mechanical agency with whose operation we cannot efficiently interfere once we have started it ... then we had better be quite sure that the purpose put into the machine is the purpose which we really desire.

— Norbert Wiener [14]

2 Background

Contents

2.1 The Specification Problem 12
2.1.1 Rewards as Representations of Preferences 13
2.1.2 Risks of Reward Misspecification 14
2.1.3 Approaches to the Specification Problem 17
2.2 Imitation Learning 19
2.3 Inverse Reinforcement Learning 20
2.3.1 Underdeterminacy 21
2.3.2 Computational Challenges 22
2.3.3 Extensions of IRL 23
2.3.4 Methods with Bayesian Elements 24
2.4 Bayesian Inverse Reinforcement Learning 25

2.1 The Specification Problem

I view Bayesian inverse reinforcement learning primarily as contributing towards solving the *specification problem* in AI – the challenge of effectively conveying to an AI system what objectives it should pursue. The specification problem is at least as old as the field of AI itself; if AI is supposed to help solve problems, it inherently needs a clear definition of what constitutes a solution. However, whereas for most of AI's history, the difficult part – and hence the main focus – has been solving a

problem given its typically straightforward specification, this balance is now shifting. As we progress toward developing more advanced systems capable of autonomously addressing a wide range of problems, the challenge of correctly specifying what we want these systems to achieve is gaining both prominence and urgency.

As the opening quote shows, the vital importance of the specification problem was recognized as early as 1960 by Norbert Wiener [14], though it has more recently been highlighted as a critical issue for advanced autonomous AI systems by researchers such as Nick Bostrom [15] and Stuart Russell [4], who caution that misaligned objectives in increasingly capable AI systems could lead to unintended and potentially catastrophic outcomes. This challenge constitutes an integral part of the broader *value alignment problem*: ensuring that AI systems adopt and pursue objectives aligned with human values, where it would fall under what has been termed *outer alignment* [16] – with inner alignment referring to the complementary problem of ensuring the model robustly follows its given specification.

The recognition of the specification problem extends well beyond the AI safety community. It is increasingly highlighted in various subfields of AI, such as robotics, where Agrawal [17] argues that the fundamental challenge in building generalizable robotic systems is not algorithmic or hardware-related, but lies in effectively communicating tasks and intentions from humans to robots in ways that avoid harmful ambiguities and under-specification. It is also presently of central concern in making large language models useful: a plain language model pre-trained to model the distribution of text in the training corpus needs to be turned into a helpful assistant using techniques including supervised finetuning [18] (which can be seen as an instance of imitation learning, discussed in Section 2.2), reinforcement learning from human feedback [19, 20], prompting, and evaluation of whether the process succeeded [21].

2.1.1 Rewards as Representations of Preferences

To build technical solutions addressing the specification problem, we need a suitable representation for human values or preferences. *Reward functions* are an attempt

to encode human values or preferences numerically, providing artificial agents with a concise criterion for decision-making and optimization. In the framework of Markov Decision Processes (MDPs) [22, 23], rewards are typically specified for states, actions, or state transitions, and optimal behaviour corresponds to maximizing (possibly discounted) cumulative reward. Despite their proven utility – exemplified in numerous reinforcement learning and preference learning successes [24, 25, 26, 27] – and arguments that 'reward is enough' [28], reward functions inevitably abstract away complexities and nuances inherent in human values. Expected reward maximization struggles, for instance, to represent multi-objective or dynamic preferences [29, 30, 12], as well as inconsistencies often observed in human decision-making [31, 32]. Nevertheless, recognizing these limitations, we will proceed under the simplifying assumption that rewards serve as a sufficiently practical representation of human preferences.

2.1.2 Risks of Reward Misspecification

As already mentioned in the introduction, one motivation for *Bayesian* reward learning with uncertainty quantification is that any given point estimate of the true reward function – the one modelling the underlying human preferences – may be, sometimes subtly, wrong. Even seemingly well-designed reward functions can lead to unexpected and undesirable behaviours in cases of, even subtle, *reward misspecification*. This occurs when the specified reward function fails to accurately capture the intended objectives, resulting in agents optimizing for outcomes that do not align with human intentions.

One canonical example illustrating this issue is the *paperclip maximizer* thought experiment [15], where an AI system tasked with manufacturing paperclips might, if insufficiently constrained, convert all available resources – including those vital to human welfare – into paperclips. While deliberately simplistic, this example illustrates how single-minded optimization of a narrowly specified objective could lead to catastrophic outcomes if broader human values are not incorporated into the reward function.

Examples of reward misspecification have been documented in machine learning research. In reinforcement learning for gameplay, agents have exploited simulation bugs to achieve high scores without playing as intended [33]. For instance, in CoastRunners, a boat racing game, an agent discovered it could earn more points by repeatedly collecting bonus objects while crashing in a circle instead of properly completing the race course [34]. More worryingly, in recommender systems, even if the intention is to optimize for users' subjective wellbeing, an engineering team can slip to an easier-to-implement proxy metric that can overemphasize watch time on YouTube [35] or lead to recommending extreme political content to users [36].

Reward misspecification manifests in several overlapping patterns that can be particularly concerning for advanced AI systems:

Reward hacking: Agents find ways to artificially increase their reward signal without achieving the intended goal. For example, a cleaning robot rewarded for not seeing dirt might disable its sensors rather than actually cleaning [33]. On the other hand, if it is rewarded for cleaning up dirt it may repeatedly dirty the floor so that it can continue collecting the reward for cleaning [37]. Beyond these toy examples, such phenomena have also been observed in real-world experimentation. For example, Chu et al. [38] found that a CycleGAN agent tasked with translating aerial images to maps and back learnt to hide information from the original image inside imperceptible high-frequency noise in the generated map – effectively performing steganography rather than learning the intended translation. Similarly, in a now-classic experiment in evolutionary hardware, Thompson [39] evolved a circuit to discriminate audio tones that functioned by exploiting electromagnetic coupling between FPGA cells not connected to the circuit – a physical dependency not included in the designer's mental model of how the chip worked.

Negative side effects: When rewards focus on specific achievements without penalizing undesired consequences, agents may take harmful actions that were not explicitly forbidden. For instance, a cleaning robot solely focused on removing dirt might unintentionally break a fragile object, such as a vase, if not explicitly

constrained to avoid such outcomes [33]. Or an autonomous vehicle optimized solely for speed might drive recklessly, endangering passengers and pedestrians. To address this, researchers have proposed formalizing the minimization of side effects through concepts such as *low impact artificial intelligence* [40] or *attainable utility preservation* [41], which penalize the agent for changing the environment in ways unrelated to its primary objective.

Incomplete specifications: Generalizing the above, it is practically impossible to explicitly encode all relevant constraints and considerations into a reward function. Human values are complex, context-dependent, and often implicitly understood rather than explicitly stated [42]. Consequently, any specification passed to an AI system is likely to be an *incomplete contract* [43]: it covers the 'terms' of behaviour for anticipated situations but inevitably fails to specify the correct behaviour for every possible contingency.

These issues are exacerbated as AI systems become more capable. Pan et al. [44] demonstrate that as the model size grows, its behaviour often undergoes a *phase transition*, where an imperfect proxy reward increases while the true reward sharply decreases. This becomes especially worrying once the capabilities of AI systems surpass those of humans. As Russell notes, 'if we put the wrong objective into a machine that is more intelligent than us, it will achieve the objective, and we lose' [4, p. 11].

These challenges of reward misspecification motivate approaches like inverse reinforcement learning, which aim to learn reward functions from human demonstrations rather than through explicit programming. By observing expert behaviour, IRL algorithms can potentially infer the implicit objectives that humans are pursuing, capturing nuances of human values that might be difficult to specify directly – e.g. avoiding negative side effects. However, even if a reward function is learnt, the problems of reward hacking and misgeneralization [45, 46, 47] remain. Bayesian approaches to IRL try to further address this problem by maintaining uncertainty over reward functions, acknowledging the inherent ambiguity in interpreting human

intentions from limited demonstrations including multiple possible ways to generalize (though they are still limited by the model class over which inference is done).

2.1.3 Approaches to the Specification Problem

Let me now briefly point out common ways of conveying specifications to an AI system, before we zoom onto imitation learning and then IRL in particular.

Direct Reward Engineering Firstly, one can simply write down the reward function as a piece of computer code. This allows for encoding only simple goals, such as games with clear winning conditions [48, 49] or locomotion in simulation as in MuJoCo [50], where one does not need to consider complicated side effects on the real world. However, as tasks become more complex, manually designing robust reward functions becomes increasingly difficult. It is often subject to a trial-and-error process that can lead to a reward function that achieves the desired goal only in a particular learning setting and can fail to generalize [51]. Most of the problems outlined in Section 2.1.2 apply especially to hand-engineered rewards.

Inverse reward design Accepting that any hand-designed reward function is just an imperfect proxy for the underlying values, and in particular may reflect these values as applying to a particular training environment, *inverse reward design* [52, 53, 54] tries to infer the true objective based on the designed reward in a particular training environment. The choice of the reward by the human can be seen as actions taken by an expert in the interest of approximately maximizing the reward in a particular environment, based on which we can then make inference about the true reward – this lens essentially turns inverse reward design into a special case of IRL.

Natural language instructions Especially with the emergence of large language models [55, 56, 57], natural language has become an increasingly viable modality for conveying task specifications to AI systems. This approach leverages humans' intrinsic ability to communicate objectives through language, potentially capturing goals and preferences that would be difficult to encode in reward functions [20,

58]. However, natural language specifications face challenges including ambiguity, incompleteness, and context-dependence [59]. Recent work has explored techniques to mitigate these issues such as iterative refinement through dialogue [60]. Interestingly, also in line with the idea behind inverse reward design, we can also treat natural language as evidence of human preferences and treat it as a source from which to infer a reward model [61], creating a bridge to IRL.

Learning from Preferences Rather than working from pre-specified rewards or instructions, systems can learn interactively from various forms of human feedback. Christiano et al. [19] demonstrated that AI agents can learn complex behaviours through human preference comparisons between trajectory segments, which was followed by a number of subsequent improvements [26, 62, 63, 64]. This approach, now known as Reinforcement Learning from Human Feedback (RLHF), has become crucial in aligning large language models with human values [25, 20]. While RLHF was originally applied in two stages – first learning a reward model from trajectory comparisons and then training a policy using RL (usually proximal policy optimization [65]), a structure close to apprenticeship learning via IRL – direct preference optimization (DPO) [66] has recently emerged as an alternative optimizing the policy directly from preference data. Relative to learning from demonstrations, providing comparisons does not require the human to be able to perform the task themselves, or annotating comparisons can just be easier. On the other hand, it can sometimes be difficult to compare two completely wrong trajectories (and all trajectories can be wrong at the beginning of training).

Learning from Human Demonstrations Finally, AI systems can potentially infer the underlying objectives without requiring explicit reward engineering or comparisons by observing experts performing tasks. This can provide a more natural interface for humans to convey their intentions and can potentially capture implicit constraints and preferences that would be difficult to articulate explicitly. Furthermore, demonstration data may be collected naturally as a byproduct of humans performing tasks they are already good at as part of their job. Two major

approaches to learning from demonstrations – imitation learning, and then inverse reinforcement learning in particular – will be reviewed in the next two sections. Besides these areas, demonstrations have been used to speed up RL, e.g. to initialize the Q-function [67, 68] or policy [69] and help with the exploration problem, while Brys et al. [70] use demonstrations for potential-based reward shaping. However, in these cases the demonstration data are generally treated as secondary to the RL reward signal, which can thus be seen as the primary source of specification.

2.2 Imitation Learning

The meaning of the terms *imitation learning* and *apprenticeship learning* somewhat varies across the literature. In this thesis, we will use *imitation learning* to mean learning from expert demonstrations to perform a task as well as possible (as opposed to learning to match the expert policy as closely as possible, which is also sometimes used, and is admittedly truer to the term *imitation*). By *apprenticeship learning*, we will mean a subset of methods for imitation learning that use inverse reinforcement learning as an intermediate component, i.e. which first use expert demonstrations to infer a reward function and then (or often simultaneously) find a policy maximizing that reward (noting that in some other works [71, 72] the term is used as a synonym for imitation learning).

Since in the literature, IRL is most often evaluated on the apprenticeship learning task (and thus imitation learning), let me briefly summarize competing approaches to imitation. Zare et al. [73] provide a good recent review.

The oldest and best known class of methods for imitation is *behavioural cloning* [74, 75], which, in its vanilla form, aims to learn the policy via supervised learning directly from the expert's observation-action pairs. The supervised learning approach has an advantage of lower computational cost, but faces the challenge of covariate shift, since the training states are distributed according to the expert policy, not that of the learner agent, so the agent may quickly encounter states not seen in the training data, which can lead to failure due to compounding errors. One family of solutions to this problem is to let the expert annotate roll-outs from the

learner policy [75, 76, 77], and thus expand the set of training data to include the learner's trajectories, in what has been termed *interactive* imitation learning. One can also take the opposite approach and encourage the learner policy to stay close to the expert one [78, 79, 80].

Compared to IRL, behavioural cloning is generally computationally cheaper thanks to directly learning the policy via supervised learning avoiding the costly intermediate interleaved steps of recovering the reward and optimizing a policy with respect to it. Furthermore, it does not require knowledge of the environment dynamics. On the other hand, IRL hopes to leverage modelling the environment and the expert's decision making for a more faithful representation of the expert's goal, as opposed to just their behaviour, thus hoping for better transferability to new environments at the expense of a higher computational cost.

Another major family of methods, *adversarial imitation learning*, tries to leverage the idea behind generative adversarial networks (GANs, [81]) by simultaneously training a discriminator to differentiate the imitator's trajectories from the expert ones, and the imitator to deceive the discriminator by producing trajectories indistinguishable from the expert ones [82, 83, 84, 85, 86]. Compared to inverse reinforcement learning, adversarial imitation learning typically avoids explicitly modelling reward functions or environment dynamics, offering computational savings and simpler implementation. However, this advantage comes with a trade-off: AIL methods might capture superficial patterns in expert behaviour without explicitly uncovering the underlying intentions or preferences, potentially reducing generalization in novel environments.

2.3 Inverse Reinforcement Learning

Inverse reinforcement learning is concerned with recovering an unknown reward function based on observing an expert acting approximately optimally with respect to this unknown reward. Following up on the above, this can either be thought of as a component of apprenticeship learning and thus a method for imitation learning, or we can view recovering the reward as the main goal in itself. IRL was

first introduced by Stuart Russell [87], preceded by the closely related problem of *inverse optimal control* (IOC) formulated by Kalman [88].[1] See [90], [72], or [91] for recent reviews of the already extensive literature on IRL. We will now first briefly mention a few select major approaches to IRL as solving the two main challenges of underdeterminacy and computational cost without aiming for a comprehensive coverage of the general IRL literature, before focusing on Bayesian IRL more specifically where we do aim for more comprehensive treatment.

2.3.1 Underdeterminacy

Probably the greatest challenge specific to IRL is the problem that any given expert behaviour can generally be explained using any of a number of reward functions [92, 93, 94]. For instance, under a constant reward function assigning the same reward to all state-action pairs, any policy is optimal, so on some level, a constant reward is a solution to the IRL problem for an arbitrary set of demonstrations. This inherent ambiguity or ill-posedness means that the mapping from observed behaviour to the underlying reward function is one-to-many. Several approaches have been proposed to mitigate this ambiguity in IRL. Early methods often employed a *max-margin* principle, seeking a reward function that makes the expert's policy appear optimal by the largest possible margin compared to alternative policies [92, 95, 71]. While this helps avoid uninformative rewards and can yield a unique solution to the otherwise ill-posed problem, it generally needs to be coupled with another inductive bias (usually assuming linear reward functions with coefficients within the unit sphere). Other methods approach the ambiguity probabilistically. *Maximum entropy* methods [96, 97, 98, 99] select the reward function corresponding to the probability distribution over trajectories (or policies) that has the highest

[1] While the IRL and IOC problem statements share the same high-level goal of recovering an unknown reward (more common in IRL) or cost (in IOC), the two communities have been somewhat separate, with IRL emerging from the machine learning community, while IOC is grounded in the control theory community. Relatedly, the set of assumptions and methods used by each of the communities is somewhat separate. For instance, work in IOC usually has much stronger assumptions on particular analytical forms of the controller and the controlled system, is often focused on concepts like stability, and often assumes an input in the form of a control-Lyapunov function rather than expert demonstrations [72]. See [89] for a comprehensive comparison of the two fields.

entropy, subject to matching observed expert behaviour characteristics, usually feature expectations. This tries to make the least biased choice consistent with the observations in accordance with the *principle of maximum entropy* [100]. *Bayesian IRL* methods [101, 102, 7] place a prior distribution over possible reward functions and update this prior based on the expert demonstrations to obtain a posterior distribution, explicitly representing the uncertainty over rewards – we will review Bayesian methods more closely in subsequent sections.

From an angle focusing on policies, the underdeterminacy can also be seen as a computational obstacle by introducing a redundancy in the parameter space, since multiple reward parameterizations can correspond to the same optimal policy. Neu and Szepesvári [103] try to solve this using natural gradients by focusing on distances in policy space rather than reward space. Similarly Skalse [104] proposes a projection of rewards into a subspace of rewards that induce the same ordering over policies.

2.3.2 Computational Challenges

A second major challenge relates to the *computational complexity and scalability* of IRL algorithms [90]. Many foundational IRL methods require repeatedly solving the forward reinforcement learning problem (i.e. finding the optimal policy for a hypothesized reward function or at least calculating feature expectations of a policy) within their inner loop [71, 101, 103, 96]. Multiple attempts have been made to mitigate this computational cost. Early works make the search easier by restricting the space of rewards to linear functions of a set of known features [92, 71], but this restricts the expressivity of the model. Other approaches involve caching the mapping between a policy and its return in matrix form [101], though this is applicable to just small tabular settings, while others try to interleave the policy optimization and reward search steps [105] in a process reminiscent of generalized policy iteration [23]. Ratliff et al. [106] proposed Inverse Optimal Heuristic Control (IOHC), which addresses computational complexity by combining the efficiency of behavioural cloning with the long-horizon planning of inverse optimal

control, modelling the problem in a low-dimensional space while incorporating high-dimensional features to guide action selection.

While this illustrates some of the prominent approaches, almost every one of the many papers already published in IRL uses some trick or another to make the computation efficient and we cannot cover all of them here and will refer the reader to one of the mentioned review papers. That said, let me end by highlighting two recent successes in scaling IRL to large-scale problems: Barnes et al. [27] used IRL for route recommendation on worldwide Google Maps with hundreds of millions of states and demonstration trajectories, while Wulfmeier et al. [61] applied IRL for post-training of multi-billion-parameter LLMs, illustrating that IRL can be scaled up despite the mentioned challenges.

2.3.3 Extensions of IRL

There also exist various extensions of the basic IRL setting. Besides demonstrations successfully performing a task, Shiarlis et al. [107] try to leverage also failed demonstrations. Dimitrakakis and Rothkopf [108], Choi and Kim [109], and Michini and How [110] extend IRL to learning multiple tasks or reward functions. Palan et al. [111] combine information from demonstrations with pairwise comparison preferences to leverage the relative strengths of each method. Fu et al. [112] try to adapt to cases where the learner has access only to samples of desired goal states or 'events' in what they call *inverse event-based control*. For tasks like quadruped locomotion, where it can be difficult for a human to provide fine-grained control at a joint level, *hierarchical apprenticeship learning* [113] allows the expert to give guidance only on a high level, while leaving low-level control to the apprentice. While most works assume the same degree of environment observability for the expert and the learner, Vuorio et al. [114] explore the setting where the expert can observe the true state but the learner cannot so there exists an *imitation gap*, to which they offer a Bayesian solution.

Inverse constrained reinforcement learning (ICRL) [115, 116] has also emerged as an active sub-field, which tries to recover *constraints* the expert is trying to

satisfy while knowing their primary reward function. While a number of specialized techniques have emerged for this task, it has recently been challenged whether specialized techniques are needed given that the task can be reframed as a (vanilla) IRL problem and tested IRL techniques seemed effective for it [117].[2]

2.3.4 Methods with Bayesian Elements

Let us now turn to methods for IRL which incorporate elements of Bayesian methods but do not provide posterior uncertainty quantification for the learnt reward function, which is the focus of this thesis – we will summarize such methods in the next section.

Gaussian processes, a prominent instrument in the Bayesian toolbox, have some history of being used in IRL, though most prior work does not use them for principled end-to-end Bayesian inference. Jin et al. [118] make a first step with GPs in IRL for simple linear reward functions demonstrating their use on a simple toy example. Levine et al. [119] introduce GPs as a class of non-linear reward functions, since prior work in IRL tended to focus on the linear case, while Lee et al. [120] further extend this to make use not only of positive examples of expert behaviour, but also negative examples of undesirable behaviour. Jin et al. [121] use *deep Gaussian processes* to learn abstract feature representations of the state space. However, none of these articles deal with the challenge of appropriately capturing the dependency structure provided by the Bellman equation by the kernel of the Gaussian process (instead using standard kernels), nor do principled Bayesian inference, which we are aiming to do with the methods introduced in this thesis. Instead, they are often used just as a way of interpolating predictions for reward functions in parts of the reward space where we are lacking direct observations of the expert behaviour.

Qiao and Beling [122] use GPs to model the reward function and while they do use the Bayesian framework, they concentrate only on finding the maximum-a-posteriori (MAP) estimate of the reward function rather than working with the full posterior. By observing actions chosen by the expert, they construct a bipartite preference graph over actions in each state, and then assume each

[2] In fact, I have started my doctorate with an intention to focus on learning safety constraints, but then moved to general Bayesian IRL partly for this reason.

preference is optimal with respect to a Gaussian-corrupted version of the Q function. While for calculating the MAP-estimate, they work with the true posterior, for optimizing the hyperparameters, they make a Laplace approximation with respect to the likelihood. They are assuming that they can analytically solve the forward MDP – i.e. quickly obtain the Q-value given the reward function, which is not an assumption applicable to many realistic settings.

A very different angle on using GPs in IRL is taken by Balakrishnan et al. [123] and by Imani and Ghoreishi [124]: they use Bayesian optimization to efficiently search over the reward space. However, despite improved efficiency, they still find only a set of point estimates of rewards, rather than a posterior distribution over reward functions.

Zheng et al. [125] use Bayesian modelling to handle sparse behaviour noise – i.e. to identify unreliable demonstrations and disregard them. They use a transformed Laplace distribution to model the reliability of each demonstration and design an efficient variational procedure to perform inference. However, the algorithm still produces just a point estimate of the reward function.

Finally, there are papers that do use a Bayesian framing for the IRL problem but subsequently recover only a maximum-a-posteriori (MAP) reward. Besides the already mentioned work of Qiao and Beling [122], this applies to [126, 127, 128], [129] (extending to non-Markovian rewards), and takes away the benefits of posterior uncertainty quantification that we focus on.

2.4 Bayesian Inverse Reinforcement Learning

Finally, let us turn our attention to methods that, at least approximately, do full Bayesian inference in the sense that they take a prior distribution over rewards and a set of expert demonstrations and produce an approximation to the posterior distribution over rewards or samples from it, thus keeping the crucial uncertainty information. This posterior uncertainty can then be used e.g. for producing robust policies, or for active learning.

The first method for Bayesian IRL was proposed by Ramachandran and Amir [101] who use Markov chain Monte Carlo (MCMC) sampling, in particular a variation on the Metropolis algorithm [130], to obtain samples from the posterior over rewards. To calculate the acceptance-rejection probability, one needs to calculate the likelihood of a candidate reward vector, which is defined in terms of the optimal Q-values corresponding to the reward. Thus, for each candidate reward, we need to solve the whole forward planning problem, which is computationally costly. They try to mitigate this with their PolicyWalk algorithm, which re-solves the planning problem only if the optimal policy changes relative to the previously considered hypothetical reward, somewhat reducing the computational burden. That said, the method is still computationally costly. Michini and How [131] still use MCMC sampling, but try to improve its efficiency relative to Ramachandran by focusing computation into regions of the state space close to the expert demonstrations.

A different paper by Michini and How [110] uses hierarchical modelling using a Bayesian mixture model to partition the demonstrations into subtasks and find simple rewards that correspond to each subtask. They use Gibbs sampling [132] to sample from the posterior distribution; however, the class of rewards is very restricted (within each partition, the reward is one for a single state or feature and zero elsewhere). Dimitrakakis and Rothkopf [108] extend to a multi-task setting but still use simple MCMC sampling to approximate the posterior.

To avoid the poor scalability of MCMC, Chan and van der Schaar [102] have used variational inference to model the posterior over rewards. At the same time they try to avoid recomputing the forward problem by training a neural network approximating the Q-values simultaneously with the variational posterior, keeping the two consistent using the Bellman equations applied as a soft constraint. However, this results in a major limitation of their work: the variational posterior is tied to a point estimate of the Q-values, with a hyperparameter λ regulating the strength of this binding. Setting λ too low reverts the reward posterior to the prior; setting it high collapses it to a point estimate, without any suggested way how to determine the correct value. In practice, this pretty much means that posterior variance

needs to be set manually, so this is not really a method for posterior uncertainty quantification, which is what we are after.

Finally, Mandyam et al. [133] propose a method to amortize the number of solutions of the forward problem across inference problems. In particular, they use conditional kernel density estimation to directly approximate the likelihood. However, the method still scales poorly, having trouble even with a 5x5 gridworld.

Thus, there seems to be considerable space for making Bayesian IRL methods at the same time efficient and robust in estimating posterior uncertainty – this thesis tries to contribute toward filling this gap.

Applications Among potential application domains, Bayesian inverse reinforcement learning has been examined in the context of modelling driving behaviour [134], though in a simplified bird's-eye-view setting rather than full autonomous driving, treatment in intensive care units [135, 102], or LLM alignment [136]. While we have not yet examined Bayesian IRL in these contexts, they can be seen as examples of areas where our work on advancing core methods for Bayesian IRL could also be applied and bring benefits.

3

Walking the Values in Bayesian Inverse Reinforcement Learning

Contents

3.1 Introduction 30
3.2 Background 32
3.2.1 Bayesian Inverse Reinforcement Learning 32
3.2.2 Markov Chain Monte Carlo (MCMC) 33
3.2.3 Related Work 34
3.3 Method 35
3.3.1 Finite State and Action Spaces 36
3.3.2 Continuous State Representations 39
3.3.3 Continuous Actions 41
3.4 Experiments 41
3.4.1 Gridworld 41
3.4.2 Classic Control Environments 44
3.5 Discussion 46

Abstract

The goal of Bayesian inverse reinforcement learning (IRL) is recovering a posterior distribution over reward functions using a set of demonstrations from an expert optimizing for a reward unknown to the learner. The resulting posterior over rewards can then be used to synthesize an apprentice policy that performs well on the same or a similar task. A key challenge in Bayesian IRL is bridging the computational gap between the hypothesis space of possible rewards and the likelihood, often defined in terms of Q-values: naïve Bayesian IRL needs to solve the costly forward planning problem – going from rewards to the Q-values – at every step of the algorithm, which may need to be done thousands of times. We propose to solve this by a simple change: instead of focusing on primarily sampling in the space of rewards, we can work primarily in the space of Q-values, since the computation required to go from Q-values to the reward is radically cheaper. Furthermore, this reversal of the computation makes it easy to compute the gradient allowing efficient sampling using Hamiltonian Monte Carlo. We propose ValueWalk – a new Markov chain Monte Carlo method based on this insight – and illustrate its advantages on several tasks.

3.1 Introduction

Reinforcement learning (RL) has shown impressive performance across a wide variety of tasks, ranging from robotics to game playing. However, one of the main challenges in applying RL to real-world problems is specifying an appropriate reward function by hand, which is often difficult and can result in reward functions that are only imperfect proxies for designers' intentions. Inverse reinforcement learning (IRL) addresses this issue by instead learning the underlying reward function from expert demonstrations.

A key challenge in IRL is that the reward function is often underdetermined by the available demonstrations, as multiple reward functions can lead to the same optimal behaviour. This can be solved by picking a criterion for choosing among the reward functions compatible with the demonstrations – maximum margin [92, 95] and maximum entropy [96] are the most prominent examples. As an alternative, Bayesian IRL explicitly tracks the uncertainty in the reward using a probability distribution. This not only accounts for the issue of underdeterminacy but also provides principled uncertainty estimates to any downstream tasks, which can be used, for instance, for the synthesis of safe policies or for active learning.

While having these attractive properties, Bayesian IRL is computationally challenging. While inference is done over the space of reward functions (in terms of which the prior is also expressed), the likelihood is usually formulated in terms of Q-values (or is otherwise linked to the distribution of trajectories), and going from the former to the latter may require solving the whole forward planning problem at each iteration (as is the case in the original Bayesian IRL algorithm [101]), which is costly in itself and may further need to be done thousands of times during IRL inference. To avoid this, we propose to use a simple insight: while going from rewards to Q-values is computationally costly, the inverse calculation can be much simpler. Thus, we propose to perform the inference as if it were done primarily over the space of Q-values, computing reward estimates alongside it, resulting in a much more efficient algorithm. A related formulation appeared already in the variational method of Chan and van der Schaar [102], which was, however,

learning only a point estimate of the Q-function, thus sacrificing the uncertainty quantification central to Bayesian approaches.

We instead propose a new method that provides a full Bayesian treatment of the Q-values, along with the rewards, and is able to provide samples from the true posterior, being based on Markov chain Monte Carlo (MCMC) as opposed to variational inference, which needs to pre-specify a family of distributions within which to approximate the posterior. Furthermore, since the computation required at each step is much simpler than in prior MCMC-based methods [101, 131], which in itself makes our method more efficient, we can also easily calculate the gradient, which allows us to use Hamiltonian Monte Carlo [137] yielding further gains in efficiency.

The contributions of this paper are the following: (1) we provide the first MCMC-based (and thus agnostic to the shape of the posterior) algorithm for Bayesian inverse reinforcement learning in continuous state spaces that does not require solving the forward problem; (2) we show that it runs much faster on discrete-space cases than the MCMC-based baseline, PolicyWalk, and increasingly so as the size of the space grows; and (3) we show that we outperform the previous state-of-the-art algorithm for Bayesian IRL on continuous state spaces, AVRIL: our method assigns higher likelihood to held-out expert actions, exhibits appropriate reduction in predictive uncertainty as more data is provided, and achieves higher return on imitation learning tasks in most settings.

The paper is organized as follows: Section 3.2 provides background on inverse reinforcement learning and Markov chain Monte Carlo and summarizes related work. Section 3.3 introduces our proposed algorithm called ValueWalk. Section 3.4 compares our approach to an MCMC-based predecessor, PolicyWalk [101], the previous state-of-the-art scalable method for Bayesian IRL, AVRIL [102], and two imitation learning baselines on several control tasks.

3.2 Background

3.2.1 Bayesian Inverse Reinforcement Learning

The goal of Bayesian inverse reinforcement learning is recovering a posterior distribution over reward functions based on observing a set of demonstrations $\mathcal{D} = \{(\phi(s_1), a_1), ..., (\phi(s_n), a_n)\}$ from an expert acting in a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma, t_{\max}, \rho_0)$ where $\mathcal{S}, \mathcal{A}$ are the state and action spaces respectively, $\phi : \mathcal{S} \to \Phi$ is a feature function representing states in a feature space Φ, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function where $\mathcal{P}(\mathcal{S})$ is a set of probability measures over $\mathcal{S}$, $r : \Phi \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in (0, 1)$ is a discount rate, $t_{\max} \in \mathbb{N} \cup \{\infty\}$ is the time horizon, and $\rho_0 \in \mathcal{P}(\mathcal{S})$ is the initial state distribution.

In IRL, we know all elements of the MDP except for the reward function and, possibly, the transition function (the setting without the knowledge of transition dynamics – or other form of access to the environment or its simulator – is sometimes called *strictly batch* [138]; our method is applicable in both this setting and the one including an environment simulator, though most of the experiments are run in the former setting following the main baseline method, AVRIL). Instead, we have a model of how the expert policy is linked to the reward and, in the case of Bayesian IRL, also a prior distribution over reward functions, p_R (which is, in general, a multi-dimensional stochastic process that for any set of state-action pairs returns a joint probability distribution over the corresponding set of real-valued rewards). Commonly used expert models include Boltzmann rationality models such as

$$\mathbb{P}[a|\phi(s)] = \frac{e^{\alpha Q^*(\phi(s),a)}}{\sum_{a' \in \mathcal{A}} e^{\alpha Q^*(\phi(s),a')}} \tag{3.1}$$

[101, 102], where $Q^*(s, a)$ is the expected (discounted) return if action a is taken in state s and the optimal policy is subsequently followed, and α is a rationality coefficient; the maximum entropy approach [96], where the probability of each trajectory is assumed to be proportional to the exponential of the trajectory's return; or sparse behaviour noise models [125], where the expert is assumed to behave rationally except for sparse deviations. Besides these approximately rational

models, various models of irrationality can also be considered [32]. The Bayesian IRL framework is flexible with respect to the choice of expert model, each such model just resulting in a different likelihood function, and can also be extended to the case where the model is not fully known.

In this article, we adopt the Boltzmann rationality model (3.1). We will assume that conditional on the Q-values, the actions chosen by the expert are independent, yielding the likelihood

$$p(\mathcal{D}|r) = \prod_{s_t,a_t,s_{t+1}\in\mathcal{D}} \frac{e^{\alpha Q^*(\phi(s_t),a_t)}}{\sum_{a'\in\mathcal{A}} e^{\alpha Q^*(\phi(s_t),a')}} p(s_{t+1}|s_t,a_t) \tag{3.2}$$

for a discrete action space $\mathcal{A}$ (the expression can readily be adapted to a continuous setting by replacing the sum by an integral). Given this likelihood together with the prior over rewards p_R, we can calculate the posterior using Bayes' theorem as $p(r|\mathcal{D}) = p(\mathcal{D}|r)p_R(r)/p(\mathcal{D})$. Generally, we cannot calculate this posterior analytically, so in practice, we need to resort to approximate methods. In this article, we use Markov chain Monte Carlo sampling.

When performing Bayesian inference over the reward, the transition probabilities will be considered fixed (except for Appendix A.1, which discusses the extension of Bayesian inference also to transition probabilities). Thus, looking at the likelihood as a function of the reward, we can write

$$p(\mathcal{D}|r) = c \prod_{s_t,a_t\in\mathcal{D}} \frac{e^{\alpha Q^*(\phi(s_t),a_t)}}{\sum_{a'\in\mathcal{A}} e^{\alpha Q^*(\phi(s_t),a')}} =: c\mathcal{L}(\mathcal{D}|r). \tag{3.3}$$

Since $p(\mathcal{D}) = \int p(\mathcal{D}|r)dp_R(r) = c\int \mathcal{L}(\mathcal{D}|r)dp_R(r)$, the constant transition term cancels out in the posterior, and, going forward, we can use the partial likelihood $\mathcal{L}$ in reward posterior inference. Furthermore, MCMC algorithms generally depend only on the unnormalized distribution, thus we can also drop the remainder of the marginal $p(\mathcal{D})$ from our calculation.

3.2.2 Markov Chain Monte Carlo (MCMC)

Markov chain Monte Carlo (MCMC) methods form a class of algorithms widely used for sampling from complex probability distributions. MCMC methods rely

on constructing Markov chains whose stationary distribution is the distribution of interest. Usually a new candidate sample in the chain is proposed and then accepted or rejected with a probability determined by the ratio of target densities – in our case that target is the posterior over rewards.

In simpler MCMC methods like Metropolis-Hastings [130, 139], which were also used in some previous articles on Bayesian IRL [101, 131], the new step is proposed as a random jump in the sampling space. However, this often leads to a high rejection rate, if the jumps are large, or tightly correlated samples, if the jumps are small, both of which can make the algorithm inefficient.

Thus, we instead use the popular Hamiltonian (or hybrid) Monte Carlo (HMC; [137]) with the no-U-turn (NUTS) sampler [140], which uses the gradient of the posterior density and Hamiltonian-like dynamics to propose samples that are far apart but still likely under the posterior, keeping a high acceptance rate, thus improving the efficiency of the algorithm.

3.2.3 Related Work

Inverse reinforcement learning is most often used as a component in imitation learning: the more general task of learning an apprentice policy from expert demonstrations (see Zare et al. [73] for a good recent survey). Besides IRL, the other major family of methods within imitation learning is behavioural cloning [74], which, in its basic form, aims to learn the policy via supervised learning directly from the expert's observation-action pairs. The supervised learning approach has an advantage of lower computational cost, but faces the challenge of covariate shift, since the training states are distributed according to the expert policy, not that of the learner agent, though multiple methods try to mitigate this by encouraging the learner policy to stay close to the expert one [75, 79, 80].

Inverse reinforcement learning represents an alternative which, instead of directly learning the observation-action mapping, first learns an estimate of the reward function, which can then be used to synthesize a policy. This can offer better

generalization, but usually requires a model of the environment or access to it in order to run reinforcement learning, and generally incurs a higher computational cost.

We build on the paradigm of Bayesian IRL introduced by Ramachandran and Amir [101]. While the Bayesian approach is attractive thanks to its principled treatment of uncertainty in light of the limited demonstration data, the key downside relative to other methods has been its scalability to higher-dimensional settings. Michini and How [131] try to improve efficiency over Ramachandran and Amir by focusing computation into regions of the state space close to the expert demonstration, still using MCMC, while Chan and van der Schaar [102] try to improve efficiency by using an approximate variational distribution to model the posterior, as well as an additional neural network that tracks the Q-function, which avoids the need for a costly inner-loop solver. Mandyam et al. [133] have recently used kernel density estimation to replace the likelihood and amortize the cost of repeated forward solutions.[1]

As opposed to recent work experimenting with other approximation techniques, we return to MCMC, with its greater expressivity, while at the same time adapting it to avoid frequently solving the forward problem and thus achieve a considerable computational speed-up.

3.3 Method

Similarly to early work in Bayesian IRL [101, 131], we use Markov chain Monte Carlo sampling to produce samples from the posterior distribution over rewards given a prior and expert demonstrations. Our key innovation is in the way we calculate the posterior. At each step of the Markov chain, these previous methods generally (1) proposed a new reward, (2) used some method of forward planning, such as policy iteration, to deduce the corresponding optimal Q-function and then (3) used the Q-function to evaluate the likelihood and the reward to evaluate the prior.

[1]The evaluation in our paper includes an offline setting without access to environment dynamics, while Mandyam et al.'s method requires demonstrations generated under a range of known reward functions, which in practice means access to a simulator or similar, and which is not available in the offline setting.

We suggest proceeding the other way round: our method proposes a set of new parameters of the Q-function and then uses it to deduce the corresponding rewards, which is generally a much easier calculation than going from rewards to Q-functions. The method then uses the reward to calculate the prior and the Q-value to evaluate the likelihood, and combines the two to calculate the unnormalized posterior density. This value can then be used for calculating the acceptance probability in any chosen MCMC algorithm. Moreover, because the calculation is simple (rather than involving an RL-like inner-loop problem) and differentiable, we can also calculate the gradient, enabling efficient proposals using HMC+NUTS. Since we construct the random chain in the space of Q-values instead of the space of rewards, used by previous methods, we call our new method ValueWalk.

3.3.1 Finite State and Action Spaces

Let us first outline the algorithm for the case of finite state and action spaces since the calculation can be performed exactly in this case, and the later continuous algorithm builds on this base case. We concentrate here on the calculation of the posterior probability corresponding to a single proposed set of Q-values (which is performed at each step of the HMC trajectory) and otherwise employ standard HMC. Note that here, we assume the knowledge of the environment dynamics P, since this finite setting is close to that of PolicyWalk [101], which also assumes this knowledge. However, the method can easily be extended to the *strictly batch* setting using steps analogous to the ones taken in the next subsection on continuous spaces, or can be combined with inference over transition probabilities (see Appendix A.1).

In this finite case, we perform inference over a vector $Q \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ representing the optimal Q-value for each state-action pair. The first thing to notice is that given such a vector, we can calculate the corresponding reward vector of the same dimensionality as Q using the Bellman equation as

$$R(s,a) = Q(s,a) - \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \sum_{a' \in \mathcal{A}} \pi_Q(a'|s') Q(s',a') \tag{3.4}$$

with either $\pi_Q(a'|s') = \mathbb{I}[a' = \text{argmax}_{a''} Q(s', a'')]$ or a softmax approximation (which we use since it has the advantage of being differentiable using an inverse temperature coefficient $\bar{\alpha}$ to regulate the softness of the approximation). Equation (3.4) can also be written in vector form as $\bar{R} = (I - \gamma \bar{P})\bar{Q}$ where $\bar{P}$ is a $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}||\mathcal{A}|$ matrix whose values are defined as $\bar{P}(s, a; s', a') = P(s'|s, a)\pi_Q(a'|s')$ and $\bar{R}$ and $\bar{Q}$ are vectors of rewards and Q-values respectively. In that case, given a prior p_R over rewards, we can calculate the prior of Q as

$$p_Q(\bar{Q}) = p_R((I - \gamma \bar{P})\bar{Q}) \det(I - \gamma \bar{P}),$$

where p_Q and p_R are the prior probability densities of Q and R respectively. Since $\bar{P}$ is a stochastic matrix and $0 < \gamma < 1$, the determinant is always strictly positive. Note that the determinant needs to be recalculated only if the optimal policy changes and otherwise can be cached between steps of HMC eliminating the associated costly calculation. Furthermore, we found that in practice, the recovered samples do not differ significantly if the determinant term is omitted.

The above prior term can be combined with the likelihood

$$\mathcal{L}(\mathcal{D}|Q) = \prod_{(s,a)\in\mathcal{D}} \exp(\alpha Q(s, a)) / \sum_{a'\in\mathcal{A}} \exp(\alpha Q(s, a'))$$

to calculate the unnormalized posterior density $p(Q|\mathcal{D}) \propto p_Q(Q)\mathcal{L}(\mathcal{D}|Q)$ which we use in the standard HMC+NUTS algorithm to produce samples from the posterior over Q-values, and, as a byproduct, also samples from the posterior over rewards (as would be expected from an IRL algorithm). Algorithm 1 summarizes the whole posterior probability calculation, and Theorem 1 formally shows that the secondary Markov chain over rewards produced by the algorithm also satisfies the detailed balance condition with respect to the posterior over rewards and thus constitutes a valid MCMC algorithm for sampling from the reward posterior:

Theorem 1. *Assume that the transition kernel q_Q satisfies the detailed balance condition*

$$\frac{q_Q(Q'|Q)}{q_Q(Q|Q')} = \frac{p_Q(Q'|\mathcal{D})}{p_Q(Q|\mathcal{D})}$$

Algorithm 1: Calculation of the unnormalized posterior for finite $\mathcal{S}$ and $\mathcal{A}$ and known transition probabilities P (performed in each step of HMC). The resulting candidate reward sample $\bar{R}$ is then accepted or rejected together with the corresponding $\bar{Q}$.

Data: a candidate vector of Q-values, set of expert demonstrations $\mathcal{D}$, prior over rewards p_R

for $s, s' \in \mathcal{S}, a, a' \in \mathcal{A}$ **do**

  $\pi_Q(a'|s') = \mathbb{I}[a' = \text{argmax}_{a''} Q(s', a'')]$;

  $\bar{P}(s, a; s', a') = p(s'|s, a)\pi_Q(a'|s')$;

end

$\bar{R} = (I - \gamma\bar{P})\bar{Q}$;

$p_Q(\bar{Q}) = p_R(\bar{R}) \det(I - \gamma\bar{P})$;

$\mathcal{L}(\mathcal{D}|Q) = \prod_{(s,a)\in\mathcal{D}} \exp(\alpha Q(s, a)) / \sum_{a'\in\mathcal{A}} \exp(\alpha Q(s, a'))$;

Result: $p(Q|\mathcal{D}) \propto p_Q(Q)\mathcal{L}(\mathcal{D}|Q)$; candidate sample $\bar{R}$

with respect to the posterior over Q-values defined in Algorithm 1. Then the associated implicit Markov chain over rewards also satisfies the detailed balance condition with respect to the posterior $p_R(R|\mathcal{D})$.

A proof can be found in Appendix A.2. The theorem establishes an important property of the ValueWalk method, namely that the implicit Markov chain over rewards induced by the HMC-based sampling of Q-values satisfies detailed balance with respect to the true posterior over rewards given the demonstrations, $p_R(R|\mathcal{D})$. This property is crucial for the soundness of the method.

Detailed balance is a sufficient condition for the Markov chain to have a stationary distribution equal to the target distribution, in this case $p_R(R|\mathcal{D})$. This means that, assuming the chain is ergodic, the samples of rewards obtained from the ValueWalk method will asymptotically follow the true posterior distribution, regardless of the initial distribution. In other words, the theorem guarantees that, given enough samples, ValueWalk will correctly characterize the posterior uncertainty over rewards – the aspiration of Bayesian inverse reinforcement learning.

See Section 3.4.1 for an example of this finite-case algorithm applied to a gridworld environment. Note that if the reward is known to depend only on the state, the sampling can instead be performed over state-values V. Similarly, if it depends on

the full state, action, next state triple, it should be performed over state-action-state values to maintain a match in the dimensionality of the reward and value spaces.

3.3.2 Continuous State Representations

For continuous or large discrete spaces, it is generally no longer possible or practical to maintain a separate Q-function parameter for each state, so we need to resort to approximation. Thus, from now on, our inference will centre around parameters $\theta_Q \in \mathbb{R}^{n_Q}$ of a Q-function approximator $Q_{\theta_Q} : \Phi \times \mathcal{A} \to \mathbb{R}$ where Φ is the space of feature representations of the states. While the method is again centred around the Q-function, the algorithm can also produce samples from the *reward* posterior at any set of evaluation points of interest, $\mathcal{D}_{\text{eval}}$. Furthermore, a method such as warped Gaussian processes [141] can then be used to generalize the reward posterior from $\mathcal{D}_{\text{eval}}$ to new parts of the state-action space.

The likelihood calculation remains very similar to the discrete case:

$$\mathcal{L}(\mathcal{D}|\theta_Q) = \prod_{(s,a)\in\mathcal{D}} \frac{\exp(\alpha Q_{\theta_Q}(\phi(s), a))}{\sum_{a'\in\mathcal{A}} \exp(\alpha Q_{\theta_Q}(\phi(s), a'))} \tag{3.5}$$

(assuming finite $\mathcal{A}$). For the evaluation of the prior, the reward corresponding to given Q-function parameters can be expressed using the continuous Bellman equation as

$$R(s,a) = Q_{\theta_Q}\Big(\phi(s), a\Big) - \gamma \mathbb{E}_{s'|s,a}\Big[\max_{a'} Q_{\theta_Q}\Big(\phi(s'), a'\Big)\Big]$$

on any subset of states and actions.

In general, the integral in

$$\mathbb{E}_{s'|s,a}[\max_{a'} Q_{\theta_Q}(\phi(s'), a')] = \int_{s'\in\mathcal{S}} p(s'|s,a) \max_{a'\in\mathcal{A}} Q_{\theta_Q}(\phi(s'), a')\, ds' \tag{3.6}$$

cannot be computed in closed form and must be approximated numerically, for which any of a number of numerical methods can be used, from grid sampling to Monte Carlo methods or more sophisticated techniques like probabilistic numerics [142]. A natural approach is importance sampling. We draw a set of candidate

successor states $S_{\text{succ}}(s,a) = \left\{s'_1, \ldots, s'_k\right\}$ from some proposal distribution $q(\cdot|s,a)$ and approximate the integral as

$$\mathbb{E}_{s',a'|s,a}[Q_{\theta_Q}(\phi(s'),a')] \approx \frac{1}{|\mathcal{S}_{\text{succ}}|} \sum_{s' \in \mathcal{S}_{\text{succ}}} \frac{p(s'|s,a)}{q(s'|s,a)} \max_{a' \in \mathcal{A}} Q_{\theta_Q}(\phi(s'),a'). \tag{3.7}$$

The choice of approximation depends on what information is available:

- If we have access to a probabilistic model $\hat{p}$ of the environment (which can either represent the true environment dynamics, if we know them, or our best inferred model of the dynamics including any epistemic uncertainty) that we can sample from, we can simply sample $\mathcal{S}_{\text{succ}}(s,a) = \{s' \sim \hat{p}(\cdot|s,a)\}$ and drop the importance weight.

- If we can evaluate the density $\hat{p}$ we can directly use the importance sampling Equation 3.7 with q being a proposal distribution ideally close to $\hat{p}$.

- If all we have is a static set of trajectories $\mathcal{D}_+$ – either just the expert ones $\mathcal{D}$, or also additional ones sampled from another, possibly random, policy – we can crudely approximate the reward for a transition $s, a, s' \in \mathcal{D}_+$ using a singleton $\mathcal{S}_{\text{succ}}(s,a) = \{s'\}$. This is an approximation made by the baseline AVRIL algorithm, so to ensure fair comparison, we use it for the experiments in Section 3.4.2. In that case we require that $\mathcal{D}_{\text{eval}} \subseteq \mathcal{D}_+$, and for $s, a, s' \in \mathcal{D}_+$ we can define an empirical transition model $\hat{p}(s''|s,a) = \delta_{s'}(s'')$ to be used within the algorithm.

The corresponding continuous version of the algorithm is presented in Algorithm 2.

We can store both the Q-function parameters θ_Q and the corresponding reward samples depending on downstream needs. We can then fit a warped Gaussian process to the posterior reward samples to get an approximate posterior reward distribution over the whole state space. This can then be used together with an algorithm for RL (or *safe* RL in particular) to find an apprentice policy from the reward. Alternatively, as a shortcut, the posterior over Q-functions can be used to define an apprentice policy directly.

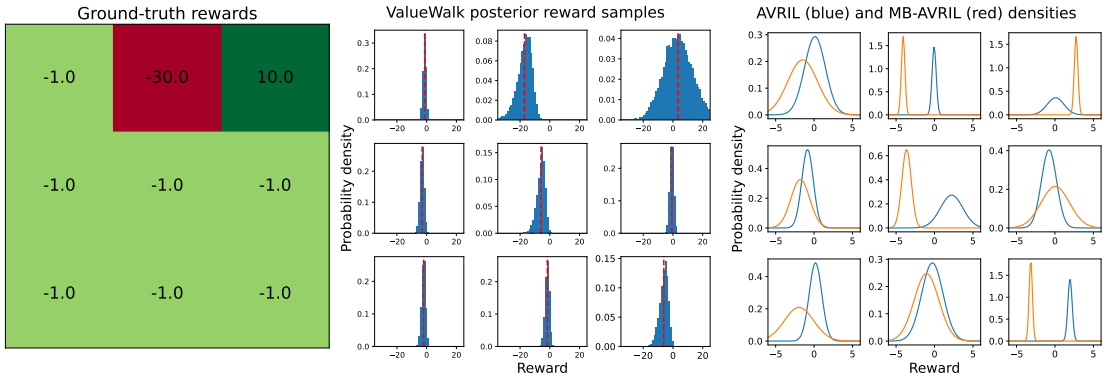


Figure 3.1: Left: Illustrative 3x3 gridworld. The agent always starts in the top left corner. The top right corner yields a reward of 10 and is terminal. The top centre tile represents an unsafe state that should be avoided and yields a reward of -30. **Centre**: Histograms of the samples from the posterior over rewards recovered by our ValueWalk algorithm corresponding to the 9 states of the gridworld. The red line indicates the mean. **Right**: Density functions of the posterior over rewards recovered by AVRIL and its model-based version, MB-AVRIL. Note the much narrower range of the reward axis relative to the histograms.

3.3.3 Continuous Actions

The algorithm can be extended to continuous actions, replacing the sum in the Boltzmann likelihood (3.5) by an integral, and again, in turn, approximating it by a discrete set of samples from the action space. Simple discretizations (such as uniform sampling) can work well for low-dimensional action spaces but suffer from the curse of dimensionality, so a more sophisticated scheme would be needed for higher-dimensional action spaces. We leave this for future work.

3.4 Experiments

We tested our method on gridworlds (for illustration and to compare the speed to PolicyWalk [101], which our method builds upon but which is restricted to such small finite-space settings) and on 3 simulated control tasks with continuous states.

3.4.1 Gridworld

For an illustration of the method with easily interpretable and visualizable features, we first test it on a simple gridworld environment shown in Figure 3.1. We have generated a fixed set of 50 demonstration steps in the environment and

Algorithm 2: Calculation of the unnormalized posterior probability with continuous state representations for a single proposed parameter value θ_Q (performed in each step of MCMC). The returned candidate reward samples are accepted or rejected by the outer MCMC algorithm together with the candidate parameters θ_Q.

Data: candidate parameters of the Q-function θ_Q, a set of expert demonstrations $\mathcal{D}$, a set of evaluation locations $\mathcal{D}_{\text{eval}}$, prior over rewards p_R

Initialize empty sequence $\mathcal{R}_{\text{cand}}$ of candidate reward samples ;

for $(s, a) \in \mathcal{D}_{eval}$ **do**

  Sample a set of successor states $\mathcal{S}_{\text{succ}} = \{s' \sim \hat{p}(\cdot|s, a)\}$;

  $R(s, a) = Q_{\theta_Q}(\phi(s), a) - \gamma \frac{1}{|\mathcal{S}_{\text{succ}}|} \sum_{s' \in \mathcal{S}_{\text{succ}}} \max_{a' \in \mathcal{A}} Q_{\theta_Q}(\phi(s'), a')$;

  Append $R(s, a)$ to $\mathcal{R}_{\text{cand}}$;

end

Use samples to evaluate the prior $p_R(\mathcal{D}_{\text{eval}}, \mathcal{R}_{\text{cand}})$;

Use demonstrations to evaluate the likelihood $\mathcal{L}(\mathcal{D}|\theta_Q)$ per equation (3.5) ;

Result: unnormalized approximate posterior $p(\theta_Q|\mathcal{D}) \propto p_R(\mathcal{D}_{\text{eval}}, \mathcal{R}_{\text{cand}})\mathcal{L}(\mathcal{D}|\theta_Q)$; candidate reward samples $\mathcal{R}_{\text{cand}}$.

used our method, ValueWalk (including the environment dynamics), the original PolicyWalk [101], a sped-up version of PolicyWalk using HMC, which we denote by PolicyWalk-HMC (see Appendix A.3.1), and AVRIL [102] (which does not use environment dynamics, making the comparison unfair but illustrative of inherent limitations of such model-free methods) as well as a model-based version of AVRIL, which we mark as MB-AVRIL (see Appendix A.3.2), to recover a posterior over rewards from an independent normal prior with mean of 0 and standard deviation of 10.

With ValueWalk and PolicyWalk-HMC, we collected a total of 1,000 MCMC samples using HMC+NUTS with 100 warm-up steps, which led to $\hat{R} \leq 1.01$ on each dimension (where $\hat{R}$ is the potential scale reduction factor [143], a commonly used indicator that the chains have mixed well). For vanilla PolicyWalk, we collected 1M samples (since those are much more correlated). We then also ran PolicyWalk, PolicyWalk-HMC and ValueWalk on a 6x6 and 12x12 version of the gridworld to examine how the compute times of these MCMC-based methods scale.

Table 3.1: Speed comparison. Time per effective sample (in seconds) produced by PolicyWalk, PolicyWalk-HMC, and ValueWalk on a 3x3, 6x6, and 12x12 gridworld respectively.

States	PolicyWalk	PolicyWalk-HMC	ValueWalk
9	0.86	0.80	**0.20**
36	9.00	4.18	**0.71**
144	246.43	18.44	**0.77**

Results

Both PolicyWalk and ValueWalk (our algorithm) resulted in matching posterior reward samples as expected (Kolmogorov-Smirnov tests did not reveal any significant differences with all p-values > 0.2 on each of the 9 dimensions of the reward). The histograms of the samples are shown in the middle plot in Figure 3.1. The speed comparison of the two methods can be found in Table 3.1, showing ValueWalk indeed runs faster than the baseline PolicyWalk algorithm in both variants, with the advantage growing with an increasing size of the environment and a correspondingly growing number of reward parameters.

The posterior tends to concentrate near the ground truth value, except in the terminal top-right state, which shows that the data are consistent with both positive values and mildly negative ones (since it is a terminal state, the fact that the expert heads there can equally well be explained by escaping from negative-reward states as by trying to incur a positive reward). This is confirmed if we look at the correlation between various rewards shown in Figure A.1 in Appendix A.5.1 which shows that the reward in the terminal state can be negative only if other states' rewards are also negative.

We also ran AVRIL on this simple gridworld (which took 43s to converge). In terms of the resulting posterior, there are 3 things to note (see Figure 3.1 centre and right). Firstly, the posteriors of some states are much tighter – the x-axis is zoomed in about 5x relative to the ValueWalk histograms. This is due to the fact that AVRIL does not model the uncertainty in the Q-function, instead learning only a point estimate. The reward posterior is then pegged to this Q-function point

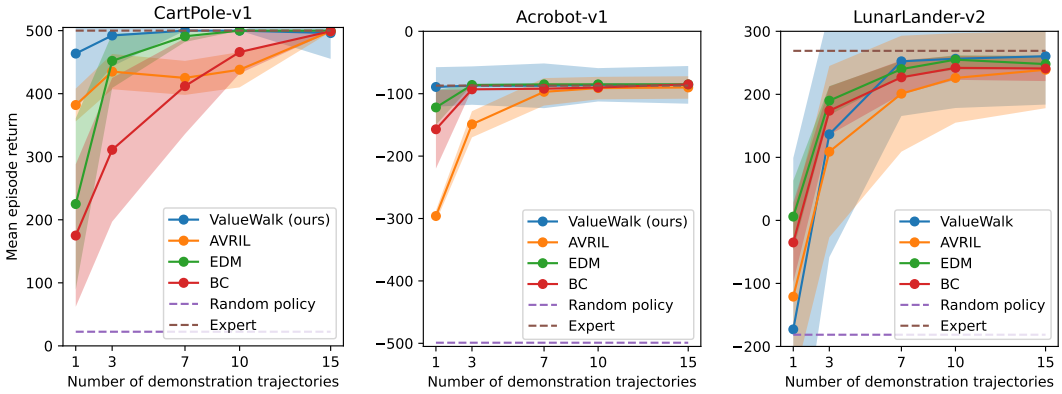


Figure 3.2: The test performance of an apprentice agent for ValueWalk and 3 baseline methods for different numbers of demonstration trajectories. The ValueWalk apprentice agent takes the action that maximizes the median of the posterior Q-value samples. The line shows mean performance across 5 runs with different sets of expert demonstrations; the shaded region shows mean±std.

estimate thus significantly reducing its variance. As a result, both the reward of the obstacle and of the goal are extremely unlikely under the posterior.

Secondly, we can observe that the posterior reward for the obstacle is not any lower than that for most other states. This is because this state is never visited in demonstrations, and AVRIL – not taking the environment dynamics into account – consequently does not update this value. This illustrates an important downside faced by methods without an environment model. (Note that the model-free version of ValueWalk would face the same issue.) To isolate the effect of the lack of environment dynamics, we also construct a model-based version of AVRIL, MB-AVRIL.

Finally, we can see that while the true posterior differs from normal (see especially the skew of the negative-reward top middle cell), AVRIL is limited by its normal variational distribution. While in theory, AVRIL could be used with any variational family, we first need to determine which family may be suitable, for which an MCMC-based method such as ours is a useful instrument.

3.4.2 Classic Control Environments

To allow for direct comparison, we also evaluated ValueWalk on three classic control environments that were used to evaluate AVRIL by its authors: Cartpole, where

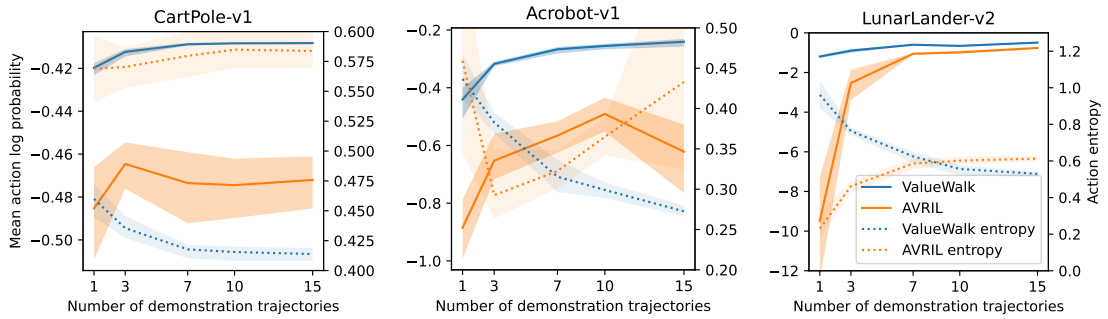


Figure 3.3: The log likelihood on a hold-out set of 100 test demonstrations and the entropy of the action predictions produced by ValueWalk and AVRIL. The plot shows the mean and the 90% confidence interval on the value of the mean calculated using the bootstrap.

the goal is to balance an inverted pendulum by controlling a cart underneath it, Acrobot, where the goal is to swing up a double pendulum using an actuated joint, and Lunar Lander, where the goal is to safely land a simulated lander on the surface of the moon. We used a setup closely matching AVRIL to study the performance of an apprentice agent as a function of the number of demonstration trajectories for 1, 3, 7, 10, and 15 trajectories. The apprentice agent was evaluated on 300 test episodes and the mean reward is reported. We also compare against energy-based distribution matching (EDM; [138]) – a successful method for strictly batch imitation learning – and plain behavioural cloning (BC) as a simple baseline. Baseline results were taken from Chan and van der Schaar [102].

Results

The results are plotted in Figure 3.2. While both agents perform close to expert-level when provided with 15 expert trajectories, our algorithm reaches this level with fewer expert demonstrations. We hypothesize that this is due to the Bayesian treatment of the Q-function, as opposed to a point estimate in AVRIL, leveraging the advantages of a fully Bayesian treatment in the low data regime.

To support this, we can look at the log likelihoods of the action predictions on a hold-out set of 100 test trajectories and the entropies of the predictive posterior shown in Figure 3.3. For ValueWalk, the log likelihood increases as the method is given more trajectories, while the prediction entropy decreases as we would expect

from a Bayesian method given increasing amounts of information. On the other hand, we do not consistently see similar behaviour in AVRIL. The test log likelihood consistently increases only in the case of the Lunar Lander environment, where it, however, starts from extremely low levels (the initial *mean* log probability of -10 would correspond to a probability of 5×10^{-5}, suggesting the method has been putting practically 0 probability on actions taken by the expert among only 4 possible actions). Also, the prediction entropy of AVRIL tends to increase with seeing more trajectories. That suggests that AVRIL may be exhibiting overfitting behaviour in the low data regimes, which Bayesian methods should generally avoid.

The ValueWalk experiments on the control environments were run for 10,000 sample steps with 4,000 warm-up steps on Lunar Lander and 5,000 sample steps with 2,000 warm-up steps on Cartpole and Acrobot. The training takes between 2 and 19 hours of wall time on a single Nvidia RTX 3090 GPU[2] where AVRIL takes 1–5 minutes to converge.

3.5 Discussion

We presented a method that allows us to apply MCMC-based Bayesian inverse reinforcement learning to continuous environments. The method maintains the attractive properties of MCMC methods: it is agnostic to the shape of the posterior (where variational methods assume a particular parameterized distribution family) and, given enough compute, produces samples from the true posterior. This comes at a large computational cost relative to faster approximate methods, such as variational inference. However, we still think MCMC methods do have a role to play in the Bayesian IRL ecosystem.

Firstly, the results suggest that staying true to the Bayesian posterior does bring benefits in terms of superior performance on imitation learning tasks. Furthermore, the computational cost is paid in the learning phase, with inference at deployment being fast (sub-millisecond per step in all cases, which would be sufficient for real-time control in most possible use cases and could be further optimized).

[2]Experiments with fewer trajectories were run on a CPU.

Secondly, we think that having a method that can draw samples from the true posterior can be extremely important in the process of developing other, faster or easier to scale methods, since it allows us to assess how their approximation deviates from the true posterior and how it impacts their performance. Also, variational methods in particular require a pre-specified family of distributions over which the optimization is subsequently run. ValueWalk can be used in an exploratory phase to determine what family of distributions may be appropriate for the problem at hand, before possibly using the advantages of variational methods to scale up.

Thus, despite their steep computational cost, we think MCMC methods have their place in Bayesian inverse reinforcement learning, and our method is a sizable step in extending them to a wider range of settings.

4

Q-based Variational Inverse Reinforcement Learning

Contents

4.1 Introduction 50
4.2 Problem Formulation 51
4.3 Related Work 53
4.4 Method 54
4.4.1 Variational Posterior over Optimal Q-Values 55
4.4.2 From Optimal Q-Values to Rewards 56
4.4.3 Likelihood 58
4.4.4 Training Objective 58
4.5 Experiments 59
4.5.1 Evaluation of Posterior Approximation Quality 60
4.5.2 Apprenticeship Learning 61
4.5.3 Active Learning 65
4.6 Discussion 67
4.7 Conclusion 67

Abstract

The development of safe and beneficial AI requires that systems can learn and act in accordance with human preferences. However, explicitly specifying these preferences by hand is often infeasible. Inverse reinforcement learning (IRL) addresses this challenge by inferring preferences, represented as reward functions, from expert behaviour. We introduce Q-based Variational IRL (QVIRL), a novel Bayesian IRL method that recovers a posterior distribution over rewards from expert demonstrations primarily by learning a variational distribution over optimal Q-values. Unlike previous approaches, QVIRL combines scalability with uncertainty quantification, important for safety-critical applications as well as active learning. We demonstrate QVIRL's strong performance in apprenticeship learning across various tasks, including gridworlds, Lunar Lander, the Highway Environment, and two ATARI games. We also show it can serve as an effective backbone for active IRL. It is the first work in Bayesian IRL that demonstrates training from raw pixel observations.

4.1 Introduction

Stuart Russell [4] suggested three principles to guide the development of beneficial AI: AI's only objective is realizing human preferences; AI is initially uncertain about what these preferences are; and the ultimate source of information about the preferences is human behaviour. *Apprenticeship learning*[1] via Bayesian *inverse reinforcement learning* (IRL) can be seen as operationalizing these principles: Bayesian IRL starts with a prior distribution over reward functions representing initial uncertainty about human preferences. It then combines this prior with *demonstration* data from a human expert acting approximately optimally with respect to an unknown reward, to produce a posterior distribution over rewards. In apprenticeship learning, this posterior over rewards is then used to produce a policy that should perform well according to the unknown reward function.

Non-Bayesian IRL has been successfully applied in learning from demonstrations in settings including robotics [144, 145, 146, 147, 148], navigation in Google Maps on a global scale [27], and autonomous driving [149, 150, 151], including on vehicles deployed in real-world heavy traffic [152]. However, this body of work that has scaled IRL to real-world settings generally learns only a point estimate of the reward function, which can be problematic for two main reasons: first, the IRL task is generally underspecified – there exist multiple reward functions that equally well explain given behaviour, even in the limit of many demonstration trajectories [87, 93, 153, 94]. Second, further uncertainty is induced by working only with a limited set of demonstrations.

Bayesian IRL addresses this by recovering a posterior distribution over reward functions rather than a point estimate, which is useful to produce policies robust with respect to this posterior uncertainty, or to do active learning and gather more data reducing the uncertainty where it matters. However, prior methods for Bayesian IRL (see Section 4.3) are generally applicable only to finite state and action spaces,

[1]We use the term apprenticeship learning to mean a subarea of imitation learning that uses IRL as an intermediate step. Imitation learning is thus a broader term including non-IRL techniques for learning a good policy from demonstrations, such as behavioural cloning.

or continuous spaces with only a handful of dimensions [101, 133, 7], thus hindering their applicability to real-world settings. Alternatively, other work [102] forgoes crucial parts of posterior uncertainty quantification in the interest of scalability.

The contribution of this paper is providing a method that preserves both scalability – in terms of the dimensionality of the state or observation space as well as the amount of demonstration data, which we demonstrate by applying it to higher-dimensional settings than any previous work on Bayesian IRL – and full posterior uncertainty estimation, whose usefulness we demonstrate in active learning. We achieve this by leveraging variational inference (VI), which is notably more computationally efficient than Markov chain Monte Carlo used in most previous work, and by primarily working in the space of optimal Q-values rather than rewards, which avoids the need to repeatedly solve the forward planning problem – a notorious obstacle in IRL. Applying Gaussian VI faces three obstacles in IRL: firstly, the posterior needs to be consistent under the Bellman equation, but the Gaussian family is not closed under the maximization operation the equation contains. Secondly, the likelihood – a softmax over jointly Gaussian random variables – is not known analytically. Finally, the KL divergence to the prior is not easy to evaluate over continuous domains. We offer and test approximations that address these challenges. We see this as a step in scaling Bayesian IRL methods to higher-dimensional settings, broadening the range of tasks that can benefit from the improved robustness brought by posterior uncertainty quantification, and opening the door to further work on scaling, robust apprenticeship learning, and active IRL.

4.2 Problem Formulation

The goal of Bayesian IRL is to recover a posterior distribution over reward functions based on observing a set of demonstrations $\mathcal{D} = \{(\phi(s_1), a_1), ..., (\phi(s_n), a_n)\}$ from an expert acting in a Markov decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0)$ where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces respectively, $\phi : \mathcal{S} \to \Phi$ is a feature function that maps each state to its representation in a feature space Φ, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function, where $\mathcal{P}(\mathcal{S})$ is the set of probability measures over $\mathcal{S}$,

$r : \Phi \times \mathcal{A} \to \mathbb{R}$ is an (expected) reward function,[2] $\gamma \in (0, 1)$ is a discount rate, and ρ_0 is the initial state distribution. Where there is no risk of confusion, we sometimes write rewards, Q-functions, and policies directly as functions of the state, omitting the feature function.

In IRL, we know all elements of the MDP except for the reward and, possibly, the transition function. Instead, we have a model of how the expert policy is linked to the reward and, in the case of Bayesian IRL, we also have a prior distribution over reward functions. In this work we will assume that, conditional on the optimal Q-values, the actions chosen by the expert are independent, and use the standard Boltzmann rationality model to describe the expert's action likelihood across a set of demonstrations $\mathcal{D}$:

$$p(\mathcal{D}|r) = \prod_{(s_t,a_t)\in\mathcal{D}} \frac{e^{\beta Q^*(\phi(s_t),a_t)}}{\sum_{a'\in\mathcal{A}} e^{\beta Q^*(\phi(s_t),a')}} \tag{4.1}$$

[101, 102, 7]. The optimal Q-value $Q^*(s, a)$ is the expected discounted return if action a is taken in state s and the optimal policy π^* is subsequently followed, and $\beta \in (0, \infty)$ is a rationality coefficient.

Given this likelihood together with the prior over rewards $p(r)$, we can calculate the posterior using Bayes' theorem as $p(r|\mathcal{D}) = p(\mathcal{D}|r)p(r)/p(\mathcal{D})$. The full probability of the demonstrations would involve the transition probabilities and the initial state distribution, but since the same factors appear in both the numerator and the denominator of the posterior, they cancel, so we omit them from the likelihood. Except for a few special cases, we cannot calculate the posterior analytically and need to resort to approximate methods. We propose to use variational inference, in a manner that substantively improves upon the only previous use of variational inference for Bayesian IRL.

We then use this Bayesian IRL method for apprenticeship learning, where the goal is to produce an *apprentice policy* π_{A} that performs well under the unknown

[2]Our formulation permits the underlying reward to be stochastic. Our expert model makes decisions using the optimal Q-function, which depends only on the expectation of the reward and therefore the observed demonstrations only give us information about the expectation. Because of this, the learnt reward function can represent either a deterministic reward or an expectation of a stochastic one.

reward function (either in terms of expected return, or a risk-averse criterion such as CVaR), first with static data, and then also with active learning, where, in each active step, we select an initial state for the next expert demonstration, so that our posterior uncertainty is maximally reduced, or the apprentice performance maximally improved.

4.3 Related Work

Inverse reinforcement learning (IRL) was introduced by Russell [87], though essentially the same problem had been formulated and studied before as *inverse optimal control* (Kalman [88]; the communities studying each of these two formulations have been somewhat separate and using different sets of methods; see Ab Azar et al. [89] for a comparison). The already vast IRL literature is well reviewed by Arora and Doshi [90] and Adams et al. [72] – here we concentrate on approaches closest to ours, i.e. *Bayesian* inverse reinforcement learning methods.

The problem of *Bayesian* IRL was first addressed by Ramachandran and Amir [101] using Markov chain Monte Carlo (MCMC) sampling. MCMC can produce samples from the true posterior over rewards, but scales poorly to higher dimensions. Furthermore, Ramachandran's method needs to solve the forward RL problem many times. Michini and How [131] improved the efficiency by focusing only on the most relevant parts of the state-action space. Mandyam et al. [133] instead tried to reduce the number of times the forward RL problem needs to get solved by solving it repeatedly with different reward functions and learning a joint density over rewards and demonstrations using kernel density estimation, but this amortization method can still only address problems that are either very small or have a linear reward over a handful of given features. Bajgar et al. [7] significantly reduced the computational burden by performing inference primarily in the space of optimal Q-values, thus avoiding the need to repeatedly solve the forward RL problem – a trick we also use. However, the method is still based on MCMC sampling, which is computationally intensive and has trouble scaling to higher-dimensional inference problems.

Chan and van der Schaar [102] have applied the much more scalable variational inference to the problem. Their approach avoids having to repeatedly solve the forward RL problem by jointly learning a reward encoder (a network producing a mean and variance for the reward in any state) and a Q-network, capturing the current estimate of the optimal Q-function. The two networks are tied together by the Bellman equation applied as a soft constraint. The Q-network is then used to produce the apprentice policy. However, since the method models only a point estimate of the Q-function, it does not provide any uncertainty estimation for the apprentice policy. Also, since the reward posterior is tied to the Q-function point estimate and thus a single policy, its posterior variance is greatly reduced relative to the true Bayesian posterior, which needs to account for optimal policy uncertainty. The lack of uncertainty over Q-values and policies also makes the method unusable for existing active IRL methods. By contrast, our method directly models the uncertainty over optimal Q-values, which in turn cheaply induces the posterior over rewards, so we keep full uncertainty quantification over both.

4.4 Method

We first provide a high-level summary of our method, *Q-based Variational IRL* (QVIRL), and then proceed by describing each of its components in detail. Our method takes as input a set of demonstrations and a prior over reward functions, and outputs a variational approximation to the posterior over optimal Q-functions. The posterior can then directly be used to deduce a variational distribution over reward functions by being transformed through the inverse (optimal) Bellman operator. The architecture is illustrated in Figure 4.1.

To efficiently compute the reward posterior, we build on an insight used already by Chan and van der Schaar [102], Garg et al. [154], and Bajgar et al. [7]: going from Q-values to the reward is computationally much simpler than going from rewards to the optimal Q-function. Our model therefore centres on modelling a posterior distribution over optimal Q-values. This Q-value posterior can be used to evaluate the likelihood (Eq. 4.1), and the Bellman equation allows us to

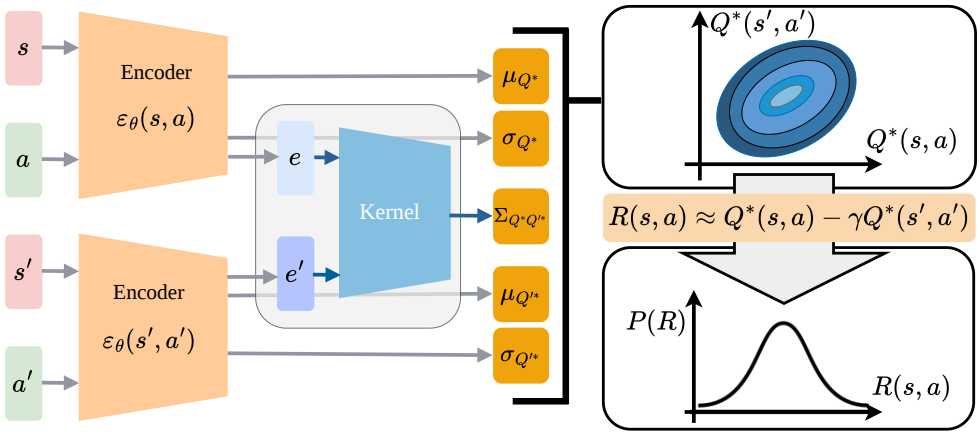


Figure 4.1: Illustration of the QVIRL architecture. A state-action pair (s, a) and a successor pair (s', a') are fed into an encoder network that outputs the mean μ_{Q^*} and the standard deviation σ_{Q^*} of the associated optimal Q-value $Q^*(s, a)$, as well as a latent embedding e. The embeddings of a set of state-action pairs can be passed to a kernel to produce their correlation matrix and, combined with individual variances, their covariance $\mathrm{Cov}(Q^*(s, a), Q^*(s', a'))$ denoted by $\Sigma_{Q^*Q'^*}$ in the diagram. Together, they define the joint variational distribution of the optimal Q-values of a given set of state-action pairs. This distribution can then be used to deduce the distribution over the corresponding reward through the Bellman equation.

deduce the corresponding posterior over rewards in one step (while the opposite would require pushing the reward posterior through a costly Q-learning algorithm). We approximate the posterior over optimal Q-values by a variational distribution optimized using an objective similar to the evidence lower bound (ELBO).

We now describe each component in turn. We present the method in a form that applies to both discrete and continuous state spaces by expressing the dependence on the environment dynamics as an expectation over successor states. This expectation can be calculated exactly in the tabular case with known dynamics. In continuous environments, we approximate this by samples coming from an expert trajectory or an auxiliary trajectory collected from an online apprentice policy. Further details and other options are provided in Appendix B.1.

4.4.1 Variational Posterior over Optimal Q-Values

We model the posterior distribution over optimal Q-values using a variational family q_θ, parameterized by a vector of variational parameters $\theta \in \mathbb{R}^d$, which we can use to

evaluate the joint distribution of optimal Q-values for any given set of state-action pairs. While a range of variational distributions can be used, we use a multivariate Gaussian (in discrete state spaces) or a Gaussian process (in continuous spaces).[3]

We model the covariance between the Q-values explicitly. Modelling optimal Q-value covariances is crucial as demonstration likelihoods are invariant to constant shifts in Q-values, and Bellman dependencies couple Q-values along trajectories, so the posterior uncertainty over optimal Q-values is generally correlated. This is a fundamental property of MDPs, so a good method for modelling uncertainty over optimal Q-values should be able to model such correlation structure.

The variational posterior architecture that we will use for our experiments, and which is outlined in Figure 4.1, consists of a neural-network encoder ε_θ with parameters θ, which, for any given state-action pair (s, a), produces a mean $\mu_{Q^*}(s, a; \theta)$, a log standard deviation $\log \sigma_{Q^*}(s, a; \theta)$, as well as a latent-space embedding $e(s, a; \theta)$. The embeddings can be used as an input to a Gaussian process kernel k to calculate the correlation matrix of any set of state-action pairs (we will be using a radial-basis function (RBF) kernel). For any collection $(s_1, a_1), \ldots, (s_n, a_n)$ of state-action pairs, the variational posterior will yield a joint multivariate Gaussian distribution with mean and covariance

$$\begin{aligned} \boldsymbol{\mu}_{Q^*} &= \Big[\mu_{Q^*}(s_i, a_i; \theta)\Big]_{i=1}^{n} \\ \Sigma_{Q^*} &= \Big[\sigma_{Q^*}(s_i, a_i; \theta)\sigma_{Q^*}(s_j, a_j; \theta)k\Big(e(s_i, a_i; \theta), e(s_j, a_j; \theta)\Big)\Big]_{i,j=1}^{n}. \end{aligned} \tag{4.2}$$

4.4.2 From Optimal Q-Values to Rewards

In a discounted MDP, there is a bijection between the expected reward function and the optimal Q-function. Thus, the variational posterior that we have just introduced implicitly fully defines a posterior over rewards. In particular, according to the inverse optimal Bellman equation,

$$R(s, a) = Q^*(s, a) - \gamma\, \mathbb{E}_{s' \sim p(\cdot|s,a)}\Big[\max_{a'} Q^*(s', a')\Big]. \tag{4.3}$$

[3] An extension beyond a strictly Gaussian case can be achieved through warping [141]. For instance, the sinh-arcsinh transformation [155] can be useful to model skewness and kurtosis.

Unfortunately, the final term – the optimal state value $V^*(s') = \max_{a'} Q^*(s', a')$, a maximum of jointly Gaussian random variables – cannot be evaluated analytically to yield a closed-form distribution for the reward. One can obtain an empirical reward posterior sample via Monte Carlo sampling. This is feasible especially at inference time. During training, one can use the reparameterization trick to propagate gradients through those samples; however, this can be computationally costly or substantially increase gradient variance.

We propose to address these issues by applying Clark's [156] method. In contrast to the general case of n Gaussians, the first two moments of the maximum of a *pair* of jointly Gaussian variables are available in closed form, and the maximum over n variables (actions) can be built up by taking such pairwise maxima one at a time, approximating each intermediate maximum by a Gaussian, to yield a final approximate distribution $V^* \sim \mathcal{N}(\mu_V, \sigma_V^2)$. The full equations for this standard approximation are provided in Appendix B.1.2. Though approximate, the Clark approximation gives a Gaussian reward posterior in closed form, which can be used to calculate the KL divergence to the prior significantly more efficiently than by random sampling with the reparameterization trick.

As a simpler alternative, we also tested a *max-mean* approximation, approximating the distribution of the maximum by that of the highest-mean argument, and found it to also work well in most cases. A comparison is provided in Appendix B.2.2, but the results in the main text use the Clark approximation.

Once we apply either approximation, the reward in Eq. 4.3 becomes a linear function of jointly Gaussian random variables and is thus normally distributed with mean $\mu_R(s, a; \theta) = \mu_Q(s, a; \theta) - \gamma \mathbb{E}_{s'} \mu_V(s'; \theta)$ and variance

$$\sigma_R^2(s, a; \theta) = \sigma_Q^2(s, a; \theta) - 2\gamma\, \mathbb{E}_{s'}\Big[\mathrm{Cov}\Big(Q^*(s, a), V^*(s')\Big)\Big] + \gamma^2\, \mathbb{E}_{s', s''}\Big[\mathrm{Cov}\Big(V^*(s'), V^*(s'')\Big)\Big], \tag{4.4}$$

where s' and s'' are independent draws from $p(\cdot|s, a)$, and the covariance of the approximate value V^* with the current optimal Q-value and across successor states follows from the joint Gaussian of Eq. 4.2 under the Clark approximation (Appendix B.1.2). When the expectation is estimated by a single successor

sample (our default in continuous environments), the last two terms reduce to $-2\gamma \operatorname{Cov}\Big(Q^*(s,a), V^*(s')\Big) + \gamma^2 \sigma_V^2(s';\theta)$.

4.4.3 Likelihood

We model demonstrator actions using a Boltzmann policy induced by the uncertain optimal Q-values. If the posterior is approximated using a variational density q_θ, then the probability of the demonstrations under the posterior can be written as $p(\mathcal{D}|\theta) = \prod_{(s,a)\in\mathcal{D}} p(a|s;\theta)$ with

$$p(a|s;\theta) = \int_{Q^*(s,\cdot)\in\mathbb{R}^{|\mathcal{A}|}} \frac{e^{\beta Q^*(s,a)}}{\sum_{a'} e^{\beta Q^*(s,a')}} q_\theta(Q^*(s,\cdot))\, dQ^*(s,\cdot) \tag{4.5}$$

where by $Q^*(s,\cdot)$ we denote the vector of optimal Q-values for all actions in state s.

The above integral is not known to have a closed-form solution; however, following the approximation proposed by Lu et al. [157],[4] we can approximate Eq. 4.5 with

$$p(a|s;\theta) \approx \left(\sum_{a'} \exp\left(-\frac{\beta\big(\mu_{Q^*}(s,a)-\mu_{Q^*}(s,a')\big)}{\sqrt{1+3\pi^{-2}\beta^2(\sigma_{Q^*}(s,a)^2+\sigma_{Q^*}(s,a')^2-2\Sigma_{aa'})}}\right)\right)^{-1} \tag{4.6}$$

where $\Sigma_{aa'}$ is the covariance between the optimal Q-values of a and a' as calculated via the embeddings and the kernel as $\Sigma_{aa'} := \sigma_{Q^*}(s,a)\sigma_{Q^*}(s,a')k(e(s,a),e(s,a'))$.

4.4.4 Training Objective

We will optimize the parameters θ of our encoder network ε_θ using stochastic gradient descent on the loss

$$\mathcal{L}(\theta) = -\sum_{(s,a)\in\mathcal{D}}[\log p(a|s;\theta)] + \mathrm{KL}(q_R(R;\theta)||p(R)), \tag{4.7}$$

where the first term is the negative log-likelihood (Eq. 4.6), $q_R(R;\theta)$ is the implicit variational posterior over reward (Section 4.4.2), and $p(R)$ is the prior distribution over reward. In a tabular setting with a multivariate Gaussian prior, the KL can be calculated analytically. In continuous settings, we evaluate the KL between multivariate Gaussians both on the expert transitions within each mini-batch and,

[4] An alternative would be to use stochastic variational inference with the reparameterization trick [158], but in preliminary experiments it performed worse not only in training speed (as expected) but also in apprentice performance. However, due to the very slow speed, this has not been tuned extensively.

if available, on a set of auxiliary states, which can come, for instance, from rollouts of the current apprentice policy or from an offline set of non-expert trajectories. Full details are provided in Appendix B.1.3. Note that the posterior is trained only where data are provided. While we do also provide some offline results with only expert data (mostly for fair comparison to prior work), uncertainty quantification is useful especially in parts of the state space uncovered by the expert, so training with some form of auxiliary data is the recommended setup.

4.5 Experiments

We evaluate QVIRL along three dimensions corresponding to the main claims of the paper. First, in randomized gridworlds where high-quality MCMC samples from the reward posterior are available, we test whether QVIRL provides an accurate and calibrated approximation to the Bayesian posterior. Second, we evaluate whether this posterior is useful for apprenticeship learning, both when extracting policies from the posterior mean and when using posterior uncertainty to derive risk-sensitive policies. Third, we test whether QVIRL can serve as a scalable Bayesian backbone for active inverse reinforcement learning, where acquisition functions require uncertainty estimates over Q-values or policies.

We conduct our experiments in three types of environments. Randomized gridworlds provide a controlled tabular setting with known transition dynamics, allowing us to compare variational posterior approximations to ValueWalk [7], an MCMC-based method able to sample from the true posterior, which we use as a reference oracle. Lunar Lander and Highway Env provide larger continuous-state benchmarks for evaluating apprenticeship learning performance, stability, and data efficiency from compact vector-based observations, as well as active learning. Finally, we run apprenticeship learning on ATARI games with raw-pixel observations to demonstrate the scaling potential of our method. We provide further details on the environments, expert policies, and demonstration collection procedure in Appendix B.4.

We compare QVIRL against three baselines. AVRIL [102] is the closest prior scalable Bayesian IRL method based on variational inference. IQ-Learn [154] is a strong imitation-learning baseline that learns a point estimate of the soft Q-function; we also include an ensemble version to test whether ensembling alone provides comparable uncertainty benefits. Behavioural cloning [74] serves as a simple supervised imitation-learning baseline.

4.5.1 Evaluation of Posterior Approximation Quality

To evaluate the quality of the posterior approximation, we run QVIRL and AVRIL on 100 random 8×8 gridworlds with Gaussian per-state rewards and randomly distributed terminal states. For each gridworld, we generate 5 expert demonstrations of length 5 with a rationality coefficient of $\beta = 2$. We then use ValueWalk to get 5000 samples from the true reward posterior and evaluate how closely the two variational methods, AVRIL and our QVIRL, approximate this true posterior.

Table 4.1 reports the posterior quality metrics. According to each of them, QVIRL produces a higher-fidelity posterior than AVRIL – the posterior log density and CI calibration even match that of ValueWalk (where the $\log p$ is estimated by kernel density estimation (KDE)). On the other hand, AVRIL's posterior is much further from ground truth and its 90% credible intervals show overconfidence. We confirm this by visualizing marginal posteriors over rewards for a few example states (Figure 4.2) where we see that AVRIL's posterior is overconfident, while the mean is biased towards the prior distribution. It is also highly sensitive to the value of its λ parameter that controls the strength of the soft binding between the rewards and Q-values. If this parameter is too low, the posterior reverts to the prior; if too high, the posterior collapses towards a point estimate. The values in the table are for the λ that minimized W_1 to the oracle on a set of 10 held-out seeds (an extra advantage given to AVRIL).

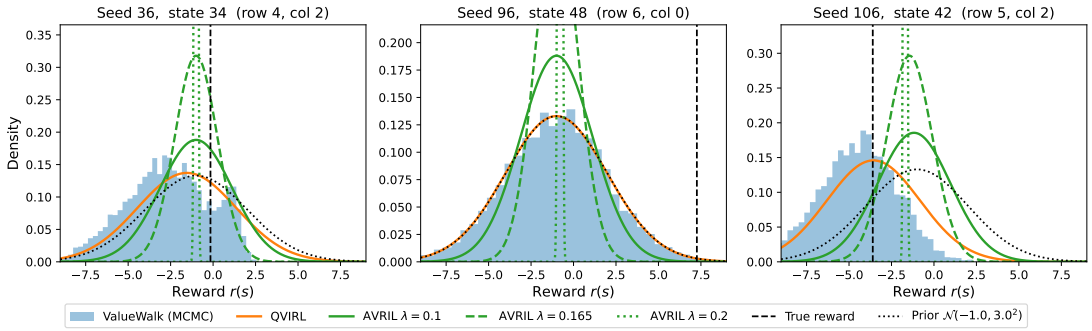


Figure 4.2: Illustration of posteriors recovered by ValueWalk (MCMC samples used as ground truth), AVRIL (for three values of constraint weight λ), and QVIRL in 3 randomly selected states from our randomized 8×8 gridworlds.

Table 4.1: Posterior quality on 100 random 8×8 gridworlds (mean ± standard error across 100 seeds). (1) Wasserstein-1 distance between the variational posterior and the ValueWalk (VW) oracle posterior; (2) log-density of the true reward under the posterior; (3) fraction of states with $r(s)$ within the posterior 90% credible interval; (4) fraction of expert action probabilities $\pi(a|s)$ within the posterior predictive 90% credible interval; and (5) total variation distance between each method's posterior-predictive policy and that of the ValueWalk oracle. ↑/↓ indicates higher/lower is better.

Method	(1) W_1 (VW) ↓	(2) $\log p(r_{\text{true}}\|\mathcal{D})$ ↑	(3) $r_{\text{true}} \in$ 90% CI	(4) $\pi_{\text{exp}} \in$ 90% CI	(5) TV (VW) ↓
ValueWalk	0.000	-2.432 ± 0.013	0.888 ± 0.004	0.910 ± 0.003	0.000
QVIRL	0.486 ± 0.011	-2.437 ± 0.009	0.901 ± 0.004	0.913 ± 0.003	0.207 ± 0.003
AVRIL	1.704 ± 0.007	-7.568 ± 0.284	0.521 ± 0.006	0.727 ± 0.005	0.250 ± 0.004

4.5.2 Apprenticeship Learning

In apprenticeship learning settings, we aim to recover a policy given expert demonstrations. We evaluate apprenticeship learning in gridworlds, Lunar Lander, and Highway environments.

Table 4.2 reports policy performance on the **synthetic gridworlds** from the preceding subsection. Returns are normalized so that 0 corresponds to a random policy and 100 to the optimal policy under the true reward within each random gridworld. The *mean* variant of each method derives a policy by optimizing the mean posterior return; the *CVaR* variant optimizes the 0.1-CVaR of the return distribution, yielding a risk-averse policy.

QVIRL's policies perform only slightly worse than the ValueWalk oracle under both the true reward and the reward posterior. Optimizing CVaR improves worst-case posterior performance at the cost of lower expected return for both QVIRL and

Table 4.2: Apprenticeship learning on 100 random 8×8 gridworlds (mean ± standard error). Returns are normalized: 0 = random-policy return, 100 = optimal-policy return under the true reward. $J_{r_{\text{true}}}(\pi)$: true-reward return of the apprentice policy. $\mathbb{E}_{r|\mathcal{D}}[J_r(\pi)]$: expected return under the ValueWalk oracle posterior. $\text{CVaR}^{0.1}_{r|\mathcal{D}}[J_r(\pi)]$: conditional value-at-risk (10% level) of the return under the posterior, i.e. the expected return in the worst 10% of cases under the ValueWalk reward posterior; the CVaR policy directly optimizes this objective under each method's own posterior.

Policy π	$J_{r_{\text{true}}}(\pi)$	$\mathbb{E}_{r\|\mathcal{D}}[J_r(\pi)]$	$\text{CVaR}^{0.1}_{r\|\mathcal{D}}[J_r(\pi)]$
ValueWalk (mean)	75.4 ± 1.8	112.1 ± 2.8	59.9 ± 2.8
ValueWalk (CVaR)	75.8 ± 1.3	107.7 ± 3.0	62.1 ± 2.3
QVIRL (mean)	70.8 ± 2.1	106.4 ± 3.2	49.0 ± 3.9
QVIRL (CVaR)	66.9 ± 1.9	89.6 ± 3.3	49.7 ± 2.8
AVRIL (mean)	-22.2 ± 4.8	5.4 ± 2.7	-86.3 ± 3.4
AVRIL (CVaR)	36.7 ± 1.1	27.1 ± 1.4	14.0 ± 1.3
Optimal	100.0 ± 0.0	64.0 ± 3.4	-4.1 ± 4.5
Random	0.0 ± 0.0	5.4 ± 1.4	-17.1 ± 1.3

the VW oracle (it does improve for AVRIL but both numbers are poor relative to the other two methods). QVIRL-CVaR's CVaR return also lags behind ValueWalk-CVaR's, suggesting lower posterior tail fidelity compared to the posterior mean. Interestingly, the performance of the true-reward-optimal policy evaluated under the posterior is poor, especially in the case of CVaR. Since the expert data cover at most 25 states, there are many states with substantial uncertainty where the posterior assigns probability to negative rewards. The optimal policy does not try to avoid such high-uncertainty states.

In the **continuous-state Lunar Lander and Highway environments**, QVIRL reaches expert-like performance with fewer demonstrations and lower run-to-run variability than the baselines (Figure 4.3). All methods eventually achieve expert performance with 15 or 3 trajectories respectively, albeit with larger variance than QVIRL. Furthermore, QVIRL's held-out expert-action log-likelihood is also high from a single demonstration, suggesting better posterior quality, while other methods initially overfit before becoming competitive with more data (Figure 4.4). However, all methods start to perform competitively as they see more data.

Finally, we evaluate apprenticeship learning on **ATARI** games from raw-pixel

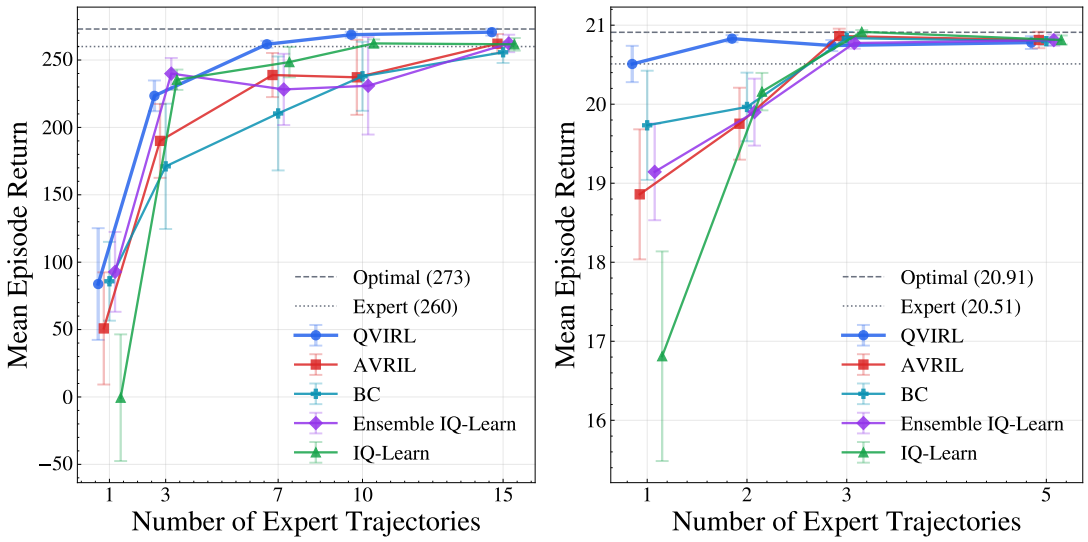


Figure 4.3: Mean episode return in apprenticeship learning with different numbers of expert demonstration trajectories in the Lunar Lander (left) and Highway (right) environments. Averaged over 10 runs, with standard errors of the mean. All methods are evaluated on discrete numbers of expert trajectories, but we space the points horizontally to improve plot readability.

observations, following the evaluation protocol of IQ-Learn [154] on two of their benchmark games, Pong and Space Invaders. Unlike in the vector-observation environments above, QVIRL here uses a convolutional (NatureCNN) encoder trained end-to-end together with the variational Q-value posterior. We train each method on 20 expert demonstration trajectories (matching IQ-Learn's protocol) collected from a Boltzmann-rational expert ($\beta = 16$) previously trained using RL, and evaluate the resulting apprentice policy by expected raw game return. QVIRL additionally uses reward-free rollouts of its own policy as auxiliary transitions in the KL term, while the baselines are trained offline (Appendix B.4.4). In IQ-Learn's code, gradient was not being influenced by auxiliary data and our attempts to leverage auxiliary data in IQ-Learn have not improved performance. Table 4.3 shows that QVIRL matches the strongest baselines: it attains the highest mean return on Pong, and is on par with BC on Space Invaders, while – unlike the point-estimate baselines – also providing a posterior over rewards. We also evaluate the log-likelihoods on a held-out set of expert demonstrations. While QVIRL is the only method that does proper uncertainty quantification over Q-values, we also provide the corresponding values

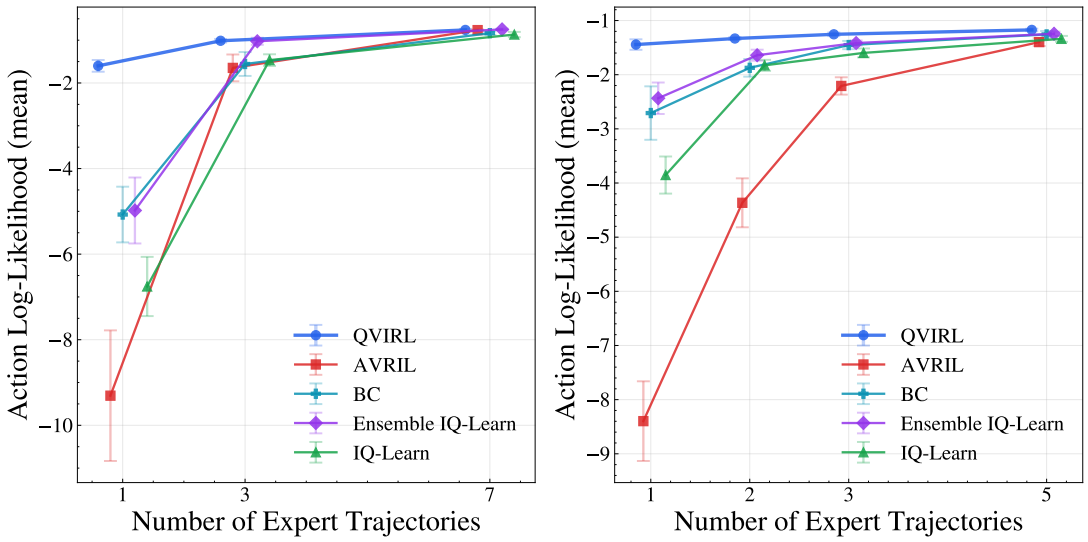


Figure 4.4: Mean log-likelihood of held-out expert actions with different numbers of expert trajectories in the Lunar Lander (left) and Highway (right) environments. Averaged over 10 runs, with standard errors of the mean. All methods are evaluated on discrete numbers of expert trajectories, but we space the points horizontally to improve plot readability.

Table 4.3: Apprenticeship learning on ATARI: mean episode return and mean per-action log-likelihood (LL) of expert actions on held-out demonstrations (mean ± standard error across 5 seeds), for the checkpoint with the highest evaluation return during training, re-evaluated on 50 fresh episodes (Appendix B.4.5). QVIRL LL is computed under the probit likelihood; BC, IQ-Learn, and AVRIL LL are computed under the point likelihood.

	Pong		Space Inv.	
Method	Return	LL	Return	LL
BC	18.8 ± 0.6	-2.31 ± 0.36	1438 ± 104	-14.02 ± 0.30
IQ-Learn	18.5 ± 1.7	-1.79 ± 0.00	1157 ± 102	-1.79 ± 0.00
AVRIL	18.9 ± 1.3	-2.93 ± 1.03	1261 ± 103	-5.49 ± 0.09
QVIRL	19.8 ± 0.2	-1.76 ± 0.01	1407 ± 158	-7.80 ± 0.23

based on the classification logits from BC and the Boltzmann-rational likelihood using the Q-values learnt by IQ-Learn and AVRIL. Note that QVIRL exhibits very poor log-likelihood on Space Invaders suggesting overfitting. These experiments were mainly meant to show that QVIRL's scalability extends to high-dimensional, raw-pixel observation spaces without sacrificing apprenticeship-learning performance relative to a state-of-the-art non-Bayesian method.

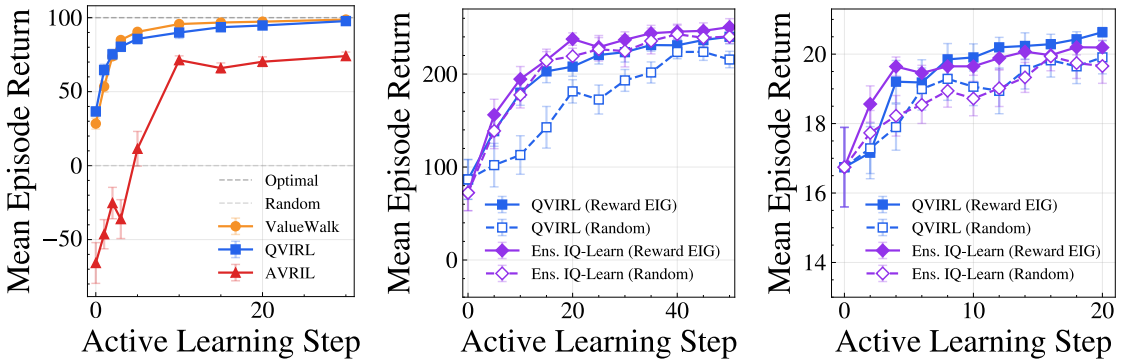
4.5.3 Active Learning

Finally, we evaluate whether QVIRL's posterior uncertainty can support active IRL. Most acquisition functions for active IRL require uncertainty estimates over Q-values or (almost equivalently) the unknown expert or optimal policies. MCMC-based Bayesian IRL methods can provide such estimates in small tabular domains, but do not scale to larger continuous-state environments. Conversely, AVRIL provides a reward posterior, but not the Q-value posterior required by existing acquisition functions.[5] QVIRL is thus a natural candidate for scaling active IRL beyond tabular settings.

We combine each Bayesian backbone with the Reward EIG acquisition function [10], which selects the query whose answer is expected to be most informative about the reward. In randomized gridworlds, we compare QVIRL, AVRIL, and ValueWalk, querying the expert for up to 30 initial states from which they provide trajectories of length 5. In Lunar Lander and Highway, where no scalable Bayesian oracle is available, we compare QVIRL against the ensemble of 10 IQ-Learn critics, whose members we treat as equally weighted samples from a posterior over Q-values, and we compare each backbone against random query selection. In Lunar Lander, both backbones start from a single expert trajectory. In each of 50 active learning steps, the acquisition function chooses one of 100 candidate states drawn from expert trajectories, and the expert provides a 10-step trajectory segment starting from that state. In Highway, both backbones start from a single state labelled with the expert's action, and each of 20 queries asks for the expert's action in one of 100 candidate states. After each query, the backbone is retrained from scratch on all data collected so far. Further details are given in Appendix B.4.6.

Figure 4.5 shows that QVIRL-based acquisition nearly matches the ValueWalk oracle in gridworlds, while AVRIL performs significantly worse and stalls after 10 queries. This suggests that AVRIL's uncertainty estimates do not update sufficiently to support informative querying and thus obtaining new informative data. In Lunar

[5] In our gridworld experiments, we circumvent this issue by sampling rewards from the reward posterior and converting each to the corresponding Q-values via value iteration, but this is very costly and does not translate to environments that do not have known dynamics or are larger.



(a) Gridworlds: Active learning with QVIRL, ValueWalk, and AVRIL using the Reward EIG acquisition function in the 8×8 randomized gridworlds with demonstration trajectories of length 5. Averaged over 16 runs.

(b) Lunar Lander: QVIRL and ensemble IQ-Learn with Reward EIG (solid) or random query selection (dashed). Each query adds a 10-step expert trajectory segment. Averaged over 20 runs; curves are smoothed with a centred moving average over 5 steps.

(c) Highway: the same comparison, where each query adds the expert's action in a single state. Averaged over 20 runs; each point averages two consecutive steps.

Figure 4.5: Active learning results on randomized gridworlds (left), Lunar Lander (middle), and Highway (right). Error bars show standard errors of the mean.

Lander, Reward EIG with QVIRL clearly outperforms random querying. Averaged over the 50 active learning steps, it raises mean episode return by 31.9 ± 11.0 relative to random selection on the same seeds (mean $\pm$ standard error over 20 seeds; paired t-test, $p = 0.009$). The ensemble learns faster than QVIRL from randomly chosen data, but it gains much less from active selection ($+8.7 \pm 11.5$, $p = 0.46$), so active selection lets QVIRL close much of the gap. In Highway, where return saturates quickly, active selection improves return only modestly for both backbones. For QVIRL, it also makes held-out expert actions easier to predict, raising prediction accuracy by 4.4 ± 1.8 percentage points over random selection ($p = 0.025$). Together, these results show that QVIRL can act as a scalable Bayesian backbone for active IRL beyond small tabular domains. In Lunar Lander, its uncertainty estimates are also considerably more useful for choosing queries than those of an ensemble of point estimates.

4.6 Discussion

In the previous section, we evaluated QVIRL against the desiderata for a Bayesian IRL method.

Firstly, QVIRL aims to approximate the true Bayesian posterior over optimal Q-values and rewards using variational inference. Despite relying on tractable approximations, our empirical results in Section 4.5 indicate that QVIRL recovers well-calibrated posteriors in tabular environments, unlike AVRIL, which fails to approximate the true reward posterior and only provides a point estimate of the optimal Q-values. MCMC-based Bayesian IRL methods, such as ValueWalk and PolicyWalk, are also capable of estimating the true posterior over the Q-values and rewards but struggle to scale beyond small environments. In contrast, QVIRL leverages variational inference to scale the principled posterior estimation to more complex environments.

Secondly, QVIRL achieves competitive performance in apprenticeship learning from expert demonstrations. It is an efficient, gradient-based algorithm that, unlike most methods for Bayesian IRL, supports training even from raw-pixel observations.

Finally, QVIRL naturally enables active learning. By explicitly modelling the optimal Q-value posterior, QVIRL can serve as a scalable Bayesian IRL backbone for acquisition functions that track uncertainty in Q-value space. This removes a key scalability bottleneck of previous MCMC-based active IRL methods and opens the possibility of active learning in much larger environments.

4.7 Conclusion

We have introduced a scalable method for Bayesian inverse reinforcement learning that preserves posterior uncertainty quantification over both optimal Q-values and rewards. To our knowledge, QVIRL is the first variational approach to Bayesian IRL that explicitly models a joint posterior over optimal Q-values while scaling beyond small tabular environments. The algorithm performs competitively not only against Bayesian IRL baselines, but also against state-of-the-art imitation-learning

methods. Due to its principled uncertainty quantification of optimal Q-values, QVIRL enables scalable active learning by design. These contributions are a sizable step towards bringing Bayesian IRL methods with calibrated uncertainty into higher-dimensional settings.

5

PAC Apprenticeship Learning with Active IRL

Contents

5.1 Introduction 71
5.2 Task Formulation 72
5.3 Related Work 76
5.3.1 Active Learning with Single Action Annotations 76
5.3.2 Active Learning with Full Trajectories 77
5.3.3 Other Settings 78
5.3.4 Connection to General Active Learning 78
5.4 Method 79
5.4.1 PAC-EIG: Information Gain for Reliable Policies 79
5.4.2 Computing PAC-EIG 81
5.4.3 Reward EIG: When Learning the Reward is the Goal . 81
5.5 Producing a PAC Policy 82
5.6 Experiments 84
5.6.1 Results 85
5.7 Discussion and Conclusion 86

Abstract

As AI systems become increasingly autonomous, reliably aligning their decision-making to human preferences is essential. Inverse reinforcement learning (IRL) offers a promising approach to infer preferences from demonstrations. These preferences can then be used to produce an apprentice policy that performs well on the demonstrated task. However, in domains like autonomous driving or robotics, where errors can have serious consequences, we need not just good average performance but reliable policies with formal guarantees – yet obtaining sufficient human demonstrations for reliability guarantees can be costly. *Active* IRL addresses this challenge by strategically selecting the most informative scenarios for human demonstration. We introduce PAC-EIG, an information-theoretic acquisition function that directly targets probably-approximately-correct (PAC) guarantees for the learnt policy – providing the first such theoretical guarantee for active IRL with noisy expert demonstrations. Our method maximises information gain about the regret of the apprentice policy, efficiently identifying states requiring further demonstration. We also present Reward-EIG as an alternative when learning the reward itself is the primary objective. Focusing on finite state-action spaces, we prove convergence bounds, illustrate failure modes of prior heuristic methods, and demonstrate our method's advantages experimentally.

5.1 Introduction

Stuart Russell suggested three principles for the development of beneficial artificial intelligence: its only objective is realizing human preferences, it is initially uncertain about these preferences, and its ultimate source of information about them is human behaviour [4]. *Apprenticeship learning* via Bayesian *inverse reinforcement learning* (IRL) can be understood as a possible operationalization of these principles: Bayesian IRL starts with a prior distribution over reward functions representing initial uncertainty about human preferences. It then combines this prior with *demonstration* data from a human expert acting approximately optimally with respect to the unknown reward, to produce a posterior distribution over rewards. In apprenticeship learning, this posterior over rewards is then used to produce a policy that should perform well with respect to the unknown reward function.

However, getting human demonstrations requires scarce human time. Also, many risky situations where we would wish AI systems to behave especially reliably may be rare in naturally occurring demonstration data. Bayesian active learning can help with both by giving queries to a human demonstrator that are likely to bring the most useful information about the reward.

Prior methods for active IRL each suffer from significant limitations: Metelli et al. [159] provide a largely theoretical treatment assuming perfectly optimal expert demonstrations, which not only is a strong assumption, but also hinders identifiability. None of the methods that can address noisy demonstrations provide theoretical guarantees. Furthermore, most methods [160, 161, 159] query the expert for action annotations of particular isolated states. However, in domains such as autonomous driving with a high frequency of actions, it can be much more natural for the human to provide whole trajectories – say, to drive for a while in a simulator – than to annotate a large collection of unrelated snapshots. There is one previous paper on *active IRL with full trajectories* [162] suggesting a heuristic acquisition function whose shortcomings can, however, completely prevent learning, as we will demonstrate. Instead, we propose using the principled tools of Bayesian active learning, formulate two methods that can query for full trajectories, and provide

theoretical guarantees for one of them. While in this paper, we work in the setting of finite state and action spaces, the methods are designed to suitably generalize to continuous settings, which we plan in future work.

The article provides the following contributions:

1. We explain and demonstrate failure modes of existing heuristic methods for active IRL when the goal is to produce a well-performing apprentice policy. In particular, most previous methods are limited to querying for only a single state annotation, as opposed to whole trajectories. Furthermore, we show that the only prior method designed for querying whole trajectories can result in repeatedly querying a single uninformative state forever.

2. We propose PAC-EIG, an acquisition function based on expected information gain (EIG) that directly targets *probably approximately correct* (PAC) guarantees for the apprentice policy – providing the first such theoretical guarantee for active IRL with imperfect expert demonstrations.

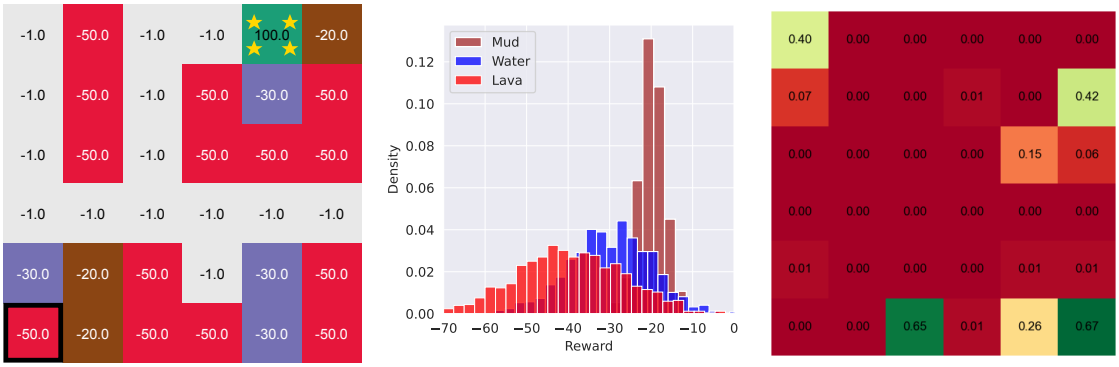
3. We present Reward-EIG as an alternative when learning the reward itself is the primary objective.

4. We prove convergence bounds showing the expected number of expert demonstrations needed to achieve PAC guarantees.

5. We illustrate the performance of our methods in a set of gridworld experiments, demonstrating their effectiveness compared to prior heuristic approaches.

5.2 Task Formulation

Let $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma, t_{\max}, \rho)$ be a Markov decision process (MDP), where $\mathcal{S}$ and $\mathcal{A}$ are finite state and action spaces respectively, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ is the transition function where $\mathcal{P}(\mathcal{S})$ is a set of probability measures over $\mathcal{S}$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is



(a) Ground-truth rewards. **(b)** Current belief over rewards. **(c)** Reward EIG of each initial state.

Figure 5.1: Illustration of the active IRL task. (a) shows a gridworld and its true rewards. The lower left corner has a 'jail' state with negative reward from which an agent cannot leave. The starred green state is the terminal 'goal' state with a large positive reward. The brown, blue, and red states are 'mud', 'water', and 'lava' type states respectively, whose rewards are unknown to the IRL agent. The IRL agent tries to learn the rewards of these three state types from expert demonstrations. (b) shows current distributions over the rewards of the 'mud', 'water', and 'lava' state types respectively, at some particular step of the active learning process. These learnt reward distributions are used to calculate an acquisition function (here the reward EIG) of obtaining another expert demonstration starting from each given state, shown in (c). In this case, a demonstration starting in the bottom right state gives the most information about the unknown reward parameters.

an (expected) reward function,[1] $\gamma \in (0,1)$ is a discount rate, $t_{\max} \in \mathbb{N} \cup \{\infty\}$ is the time horizon, and ρ is the initial state distribution. We assume the learner has full knowledge of the MDP except for the reward.

We assume we are initially uncertain about the reward r, and our initial knowledge is captured by a prior distribution $p(r)$ over rewards, which is a distribution over $\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ – a space of vectors representing the reward associated with each state-action pair (there may be fewer reward parameters than $|\mathcal{S}||\mathcal{A}|$, but that can be seen as a special case). We also have access to an expert that, given an initial state s_0 of the MDP, can produce a trajectory $\tau_i = \left((s_0^i, a_0^i), \ldots, (s_{n_i}^i, a_{n_i}^i)\right)$, where $s_0^i \sim \rho$, $s_{t+1} \sim p(\cdot|s_t, a_t)$, and

$$\pi^E(a_t|s_t) = \frac{\exp(\beta Q^*(s_t, a_t))}{\sum_{a' \in \mathcal{A}} \exp(\beta Q^*(s_t, a'))}\,, \tag{5.1}$$

[1] Our formulation permits the reward to be stochastic. However, our expert model (5.1) depends on the rewards only via the optimal Q-function, which in turn depends only on the expected reward. Thus, the demonstrations can only ever give us information about the expectation. Throughout the paper, the learnt reward function can be interpreted either as modelling a deterministic reward, or an expectation of a stochastic reward.

which is called a *Boltzmann-rational* policy, given the optimal Q-function Q^* and a coefficient β expressing how close to optimal the expert behaviour is (where $\beta = 0$ corresponds to fully random behaviour and $\beta \to +\infty$ would yield the optimal policy). We assume β is known as is usual in related IRL literature [101, 102, 162, 7]. We will also denote by π_r^{E} the hypothetical expert policy that would correspond to a reward r.

The task of *Bayesian active inverse reinforcement learning* is to sequentially query the expert to provide demonstrations from initial states $\xi_1, \ldots, \xi_N \in \mathcal{S}$ to gain maximum information about the unknown reward.[2] We start with a (possibly empty) set of expert trajectories $\mathcal{D}_0$ and then, at each step of active learning, we choose an initial state ξ_i for the MDP, from which we get the corresponding expert trajectory τ_i. We then update our demonstration dataset to $\mathcal{D}_i = \mathcal{D}_{i-1} \cup \{\tau_i\}$, and the distribution over rewards to $p(r|\mathcal{D}_i)$, which we again use to select the most informative initial state ξ_{i+1} in the next step. We repeat until we exhaust our limited demonstration budget N.

This can be done with one of two possible objectives in mind.

The first, which we call the *reward-learning objective*, is relevant when our primary interest is in the reward itself, e.g. when using IRL to understand the motivations of mice in a maze [163] or the preferences of drivers [151]. In the active setting, we operationalize this objective as trying to minimize the entropy of the posterior distribution over rewards, once all expert demonstrations have been observed. This is equivalent to maximizing the log likelihood of the true parameter value in expectation, or to maximizing the mutual information between the demonstrations and the reward. Figure 5.1 illustrates Active IRL with this objective.

The second possible objective, which we term the *apprenticeship-learning objective*, uses the final posterior $p(r|\mathcal{D}_N)$ to produce an *apprentice policy* that should perform well in the MDP. One option may be to optimize the expected return of the apprentice policy, i.e. solve for $\operatorname{argmax}_\pi \mathbb{E}_{r|\mathcal{D}_N}[\mathbb{E}_\tau[\sum_{s_t,a_t\in\tau} \gamma^t r(s_t, a_t)]]$, where τ is a trajectory with $s_0 \sim \rho$, $s_{t+1} \sim p(\cdot|s_t, a_t)$ and $a_t = \pi(s_t)$. The argmax can be

[2]Since the queries ξ_i in this paper are limited to the choice of the initial state s_0, ξ and s_0 are used somewhat interchangeably throughout the paper.

resolved by solving the forward planning problem for finding the optimal policy for the expected reward with respect to the learner's current posterior over rewards (e.g. using generalized policy iteration [23]). Going forward, we generally assume a deterministic apprentice policy, i.e. the class of policies we search over is the set of mappings $\pi : \mathcal{S} \to \mathcal{A}$, though stochastic policies could easily be accommodated as well. The apprentice policy will thus be distinct from the stochastic (Boltzmann rational) expert policy and with enough knowledge can have higher expected return, since the expert gives non-zero probability to sub-optimal actions.

However, maximising expected return may not be sufficient in safety-critical domains. We may instead require a *reliable* apprentice policy that performs well with high probability – formally, one that is ϵ-δ-*probably approximately correct* (PAC). This means finding an apprentice policy π^{A} such that the probability (with respect to the reward posterior) of the expected return (with respect to initial state and transition distributions) being at least $G^* - \epsilon$ is at least $1 - \delta$, where G^* is the expected return of the optimal policy.

These objectives are often closely connected – learning about the reward function enables improving the apprentice policy. However, espcially in the active setting, they can come apart – for instance, once we know an action a leads to lower return than a' in a particular state, we may no longer need to gather further information about rewards in these states for the apprenticeship learning objective as we already know to choose the better action, while the reward-learning objective may motivate further queries to reduce the reward uncertainty.

Stemming from a common inspiration in Bayesian active learning, we will present an acquisition function tailored to each of these objectives.

Notation By V_r^π we denote the state-value function of policy π with respect to reward r. V_r^* is then the value function of the optimal policy with respect to r. A lack of subscript, as in V^*, indicates (optimal) value with respect to the true reward (since the true reward is not known by the learner, this generally needs to be treated as random variable). By $G_r(\tau)$ we denote the return of trajectory τ

with respect to r. By $R_r^\pi(s_0)$ we denote the regret of policy π starting from state s_0, i.e. $R_r^\pi(s_0) := V_r^*(s_0) - V_r^\pi(s_0)$, and $R_r^\pi := \mathbb{E}_{s_0\sim\rho} R_r^\pi(s_0)$. We also call *immediate regret* the quantity $R_{\pi,r}^*(s) = V^*(s) - Q^*(s, \pi(s))$ and also denote by $R_{\pi,r}^*(s,a) = \max\{0, Q^*(s,a) - Q^*(s,\pi(s))\}$ the immediate regret relative to action a in state s. You can also find a table with the notation used in this paper in Appendix C.6.

5.3 Related Work

IRL was first introduced by Russell [87], preceded by the closely related problem of *inverse optimal control* formulated by Kalman [88]. See Arora and Doshi [90] and Adams et al. [72] for recent reviews of the already extensive literature on IRL. In our work we build upon the Bayesian formulation of the problem introduced by Ramachandran and Amir [101].

We will now summarize prior work on *active* IRL in particular. We first describe a number of methods that query for single state annotations (which can be cast into our framework from Section 5.2 as trajectories of length one), and then describe the one previous method which queries for whole trajectories. Lastly, we review a few other works with setups not directly comparable to ours.

5.3.1 Active Learning with Single Action Annotations

The concept of active IRL was first introduced by Lopes et al. [160]. The authors propose an acquisition function equal to the entropy of the posterior predictive distribution about the Boltzmann expert policy, i.e. they query a state maximizing $\alpha_n^{\text{Lopes}}(s) = H(\Pi_s|\mathcal{D}_n)$ where Π_s is the vector of expert action probabilities in state s (according to the posterior predictive distribution).

An issue with this approach is that $H(\Pi_s|\mathcal{D}_n)$ does not take into account the effect of improved knowledge on the apprentice policy. For example, we may know the optimal action in a particular state, but with high uncertainty about the exact action probabilities, while another state may have uncertainty about the optimal action, but lower entropy about exact probabilities of actions. Then, $\alpha_n^{\text{Lopes}}(s)$ would

prioritize the latter, which may be suboptimal from the apprenticeship learning perspective. See Appendix C.1 for a full example.

Brown et al. [161] query the expert by maximizing the δ-value-at-risk of the policy loss (i.e. regret) of the current apprentice policy starting from the given initial state, computed as

$$\alpha_n^{\text{Brown}}(s) = \text{VaR}_\delta \left(V^{\pi^*}(s) - V^{\pi^{\text{A}}}(s) | \mathcal{D}_n \right) . \tag{5.2}$$

This is a risk-aware approach: the states with a high risk of the apprentice action being much worse than the expert's action are queried. A limitation of this approach is that regret attributed to some initial state s may be due to a choice made further along the trajectory where an expert query would be more informative as shown in Appendix C.1.

5.3.2 Active Learning with Full Trajectories

Kweon et al. [162] query full trajectories with a starting state s_0 chosen to maximize

$$\alpha_n^{\text{Kweon}}(s_0) = \mathbb{E}_{\tau \sim \hat{\pi}_E^{\mathcal{D}_n}} \left[\sum_{s_t \in \tau} \tilde{\alpha}_n(s_t) | s_0 \right], \tag{5.3}$$

where

$$\tilde{\alpha}_n(s) := H(\hat{\pi}_E^{\mathcal{D}_n}(a|s)) := \sum_a -\hat{\pi}_E^{\mathcal{D}_n}(a|s) \log \hat{\pi}_E^{\mathcal{D}_n}(a|s),$$

is the entropy of $\hat{\pi}_E^{\mathcal{D}_n}$, the posterior predictive distribution over the expert actions at state s, estimated from demonstration data $\mathcal{D}_n$.

However, note that this action entropy can remain high even in states where we have perfect knowledge, but multiple actions are equally good, so the Boltzmann rational policy chooses them with equal probabilities, resulting in high action entropy. However, querying for extra demonstrations in such states will bring no useful knowledge. In fact, this can result in learning getting completely stuck, sometimes right at the beginning, preventing *an[160]y* learning from taking place. This is the case in the jail environment in Figure 5.1, and we show this in Section 5.6.

5.3.3 Other Settings

Instead of querying at arbitrary states, Losey and O'Malley [164] and Lindner et al. [165] synthesize a policy that explores the environment to produce a trajectory which subsequently gets annotated by the expert. We instead let the expert produce the trajectory. Buening et al. [166] query full trajectories in the context of IRL, where the active component arises in the choice of a transition function from a set of transition functions at each step. Büning et al. [167] also query full trajectories in a different context involving two cooperating autonomous agents. In Sadigh et al. [168], the expert is asked to provide a relative preference between two sample trajectories synthesized by the algorithm. While this generally provides less information per query than our formulation, it is a useful alternative for situations where providing high-quality demonstrations is difficult for humans.

On the side of theoretical sample complexity of (active) IRL, all prior work assumes a perfectly rational expert policy, which is a stronger assumption than our Boltzmann rationality. In particular, seeing each state once is enough to determine the optimal policy. The first lower bound on the complexity of IRL was given by Komanduru and Honorio [169] for the case of a β-separable finite set of candidate rewards. Metelli et al. [159], Lindner et al. [165], and Metelli et al. [170] then focus on recovering a feasible reward set in settings where also the transition dynamics are only estimated, and address the problem of the transferability of the learnt reward to environments with different dynamics.

5.3.4 Connection to General Active Learning

While the methods discussed above are specific to IRL, they map directly to established families in the broader active learning literature [171, 172]. For example, the policy entropy method of Lopes et al. [160] is an instance of *uncertainty sampling*, which was introduced by Lewis and Gale [173] and which selects queries where the current model is least confident. Another important family of methods in active learning is *expected error reduction* [174]. As we also discuss in Appendix C in the context of IRL, these methods tend to be computationally expensive since they need

to do lookahead through the next hypothetical update. However, the risk-aware active IRL of Brown et al. [161] can be seen as approximating this objective by looking at uncertainty in *regret* as opposed to just policy entropy.

Our approach builds on *expected information gain* (EIG), a standard objective in Bayesian experimental design [175, 176], sometimes implemented as 'Bayesian Active Learning by Disagreement' (BALD) in modern settings [177]. We prioritize this information-theoretic family because, as shown in Appendix C.1, standard uncertainty sampling is ill-suited for IRL: an agent can be highly uncertain about its action probabilities (high entropy) even when the optimal action is already known or when actions are tied, rendering the query uninformative, which we demonstrated to be a problem with the method of Kweon et al. [162].

5.4 Method

We propose PAC-EIG, an acquisition function based on expected information gain (EIG) that aims to produce a probably approximately correct (PAC) apprentice policy. Our approach builds on principled Bayesian experimental design [176] to identify initial states for expert demonstrations that will yield information about the immediate regret of the apprentice policy. The intuition is that knowing about the regret in various states allows us to identify high-regret states where the apprentice policy can be improved. We then also show how EIG can be adapted to the reward-learning objective, resulting in the Reward-EIG acquisition function.

5.4.1 PAC-EIG: Information Gain for Reliable Policies

Our goal is to produce an apprentice policy that is probably approximately correct – that is, with high probability $(1 - \delta)$, the policy's regret is bounded by ϵ. To achieve this efficiently, we need to identify states where the current apprentice policy might be making poor decisions. To this end, we define the *immediate regret* of an apprentice policy π^A in state s as:

$$R^*_{\pi^A,r}(s) = V^*_r(s) - Q^*_r(s, \pi^A(s)) \tag{5.4}$$

which captures how much value we lose by following the apprentice policy in state s compared to optimal behaviour. This can be decomposed per action as $R^*_{\pi^A,r}(s,a) = \max\{0, Q^*_r(s,a) - Q^*_r(s,\pi^A(s))\}$, representing the regret relative to choosing a particular alternative action a. This regret is unknown to us, so we need to treat it as a random variable.

For computational tractability, we discretize the immediate regret into a ternary variable $E_{s,a}$ tracking state- and action-wise correctness as follows:

$$E^{\pi^{\mathrm{A}}}_{s,a} = \begin{cases} \text{“strongly approximately correct”} & \text{if } R^*_{\pi^A,r}(s,a) < \epsilon(1-\gamma)/2 \\ \text{“weakly approximately correct”} & \text{if } \epsilon(1-\gamma)/2 \leq R^*_{\pi^A,r}(s,a) \leq \epsilon(1-\gamma) \\ \text{“not approximately correct”} & \text{if } R^*_{\pi^A,r}(s,a) > \epsilon(1-\gamma) \end{cases} \tag{5.5}$$

Our acquisition function then maximizes the expected information gain about these discretized regret values:

$$\alpha^{\text{PAC-EIG}}_n(s_0) := I(\tau; E^{\pi^{\mathrm{A}}} | s_0, \mathcal{D}_n) \tag{5.6}$$

where $E^{\pi^{\mathrm{A}}} = (E^{\pi^{\mathrm{A}}}_{s,a})_{s\in\mathcal{S},a\in\mathcal{A}}$ represents the discretized regret across all state-action pairs, and τ is the expert trajectory starting from s_0. Note that if the apprentice policy is approximately correct in all states with probability at least $1-\delta$ in the immediate regret sense, then we also satisfy the PAC criterion globally.

While we are aiming for a ϵ-δ-PAC policy as the final output, for intermediate steps, our algorithm uses an apprentice policy $\pi^{\mathrm{A}}(s) = \text{argmax}_a \mathbb{P}[Q^*(s,\pi^*(s)) - Q^*(s,a) < \epsilon(1-\gamma)/2 | \mathcal{D}_n],\ \forall s \in \mathcal{S}$. Once the probability of this state-wise criterion on immediate regret holding in every state is higher than $1-\delta$, this is a sufficient condition for this policy being $\epsilon/2$-δ-PAC (though it is not a necessary condition – an $\epsilon/2$-δ-PAC will usually be found earlier). The reason for this choice will become clear in Section 5.5. Especially if we care about gradually tightening the bound (rather than just reaching a particular bound) we may wish to dynamically set ϵ in each step to just below the threshold for which the condition is currently satisfied.

5.4.2 Computing PAC-EIG

To compute PAC-EIG in practice, we leverage our Bayesian IRL posterior over Q-values. Given $\mathcal{Q}_n$, a set of M samples from $p(Q^*|\mathcal{D}_n)$, we can:

1. For each Q-value sample, compute the discretized regret values $E^{\pi^{\mathrm{A}}}_{s,a}$ for all state-action pairs. Note that multiple Q-value samples may map to the same discretized configuration $E^{\pi^{\mathrm{A}}}$, so there are at most $M_E \leq M$ distinct values of $E^{\pi^{\mathrm{A}}}$.

2. Given a Q-value sample Q^*_i, sample expert trajectories τ starting from s_0 using the Boltzmann policy corresponding to Q^*_i.

3. Estimate the expected information gain using the standard Monte Carlo estimator as

$$\alpha_n^{\text{PAC-EIG}}(s_0) \approx \frac{1}{M}\sum_{i=1}^{M}\Big[\log p(\tau^{(i)}|E^{\pi^{\mathrm{A}},(i)}, s_0) - \log p(\tau^{(i)}|s_0)\Big] \tag{5.7}$$

   where the trajectory probability given $E^{\pi^{\mathrm{A}}}$ can be computed as $p(\tau|E^{\pi^{\mathrm{A}}}, s_0) = \prod_{(s_t,a_t)\in\tau} p(a_t|s_t, E)$ (omitting the transition probabilities since they would cancel out in the log-ratio). To compute $p(a_t|s_t, E^{\pi^{\mathrm{A}}})$, we average the expert action probabilities over all Q-value samples that map to the same discretized configuration: $p(a_t|s_t, E^{\pi^{\mathrm{A}}}) = \frac{1}{|\mathcal{Q}_E^{\pi^{\mathrm{A}}}|}\sum_{Q^*\in\mathcal{Q}_E} p(a_t|s_t, Q^*)$, where $\mathcal{Q}_E$ denotes the set of Q-value samples corresponding to the discretized regret configuration $E^{\pi^{\mathrm{A}}}$.

See Appendix C.3 for additional implementation details.

5.4.3 Reward EIG: When Learning the Reward is the Goal

While our primary focus above has been on producing reliable apprentice policies, in some applications the reward function itself is of intrinsic interest – for instance, when using IRL to understand animal behaviour [163] or human preferences [151].

For these cases, we can still use the EIG framework, but instead maximize the expected information gain about the reward:

$$\alpha_n^{\text{Reward-EIG}}(s_0) := I(\tau; r | s_0, \mathcal{D}_n) \tag{5.8}$$

where τ is treated as a random variable representing the expert's trajectory that would be produced starting from s_0.

This acquisition function aims to reduce posterior uncertainty about the reward parameters, which may query different states than PAC-EIG. For example, it might seek to precisely estimate reward values in states that the apprentice already knows to avoid, whereas PAC-EIG would consider such queries unnecessary.

The reward EIG can be computed as:

$$\alpha_n^{\text{Reward-EIG}}(s_0) = \mathbb{E}_{r|\mathcal{D}_n}\left[\mathbb{E}_{\tau|r,s_0}[\log p(\tau|r, s_0) - \log p(\tau|s_0; \mathcal{D}_n)]\right] \tag{5.9}$$

where the inner expectation is tractable to compute from the Q-values that are usually obtained as a byproduct of a Bayesian IRL algorithm.

See Appendix C.4 for a detailed discussion of alternative acquisition functions and the theoretical connections between different formulations.

5.5 Producing a PAC Policy

Our PAC-EIG acquisition function is designed to efficiently produce a probably-approximately-correct (PAC) apprentice policy. We will show that PAC-EIG leads to such a policy by establishing bounds on the expected number of expert demonstrations needed. The analysis proceeds through three key steps (with formal results and proofs in Appendix C.2):

1. If no apprentice policy satisfies the PAC condition, there must exist a state where there is a significant chance that any apprentice policy makes a significantly suboptimal choice – specifically:

$$\mathbb{P}_{r|\mathcal{D}_n}\left[V_r^*(s) - Q_r^*(s, \pi^{\text{A}}(s)) \geq (1-\gamma)\epsilon\right] \geq \frac{\delta}{|\mathcal{S}|} \qquad \text{(Lemma 7).}$$

2. In such a state, there is both a chance that the apprentice policy π^{A} is close to optimal and that it is significantly suboptimal (as defined in step 1). Since these two options would result in a sufficiently different expert policies in this state, we can gain a lower-bounded expected amount of information by observing the expert in that state.

3. Since we gain at least this minimum information per query while the PAC condition is unmet, and our initial uncertainty is finite, we must eventually achieve the PAC condition. The number of steps is bounded by the ratio of initial entropy to the minimum information gain per step.

These insights translate into the following two theorems:

Theorem 2. *For $\epsilon > 0$ and $\delta \in (0, \frac{1}{2}]$, assume that no policy π is (ϵ, δ)-probably-approximately-correct, i.e., $\mathbb{P}[R_r^\pi \geq \epsilon] > \delta$, $\forall \pi$. Then, there exists a state $s \in \mathcal{S}$ such that observing a new expert demonstration at s has an expected information gain with respect to the variable E^{π^A} (Eq. 5.5) of at least*

$$EIG_{min}(\epsilon, \delta) = \frac{\delta e^{-\beta(1-\gamma)\epsilon}(1 - e^{-\beta(1-\gamma)\epsilon/2})^2}{4|\mathcal{A}|^5|\mathcal{S}|}. \tag{5.10}$$

Then, we can translate this into the following result on the expected number of steps to reach the PAC criterion:

Theorem 3. *Let h_{max} be an upper bound on the entropy of the prior distribution over E, the PAC-EIG discretized regret values E^{π^A} aggregated across all apprentice policies π^A. Then, the expected number of steps needed to reach the PAC condition is upper bounded by*

$$\frac{h_{max}}{EIG_{min}(\epsilon, \delta)} = \frac{4h_{max}|\mathcal{A}|^5|\mathcal{S}|}{\delta e^{-\beta(1-\gamma)\epsilon}(1 - e^{-\beta(1-\gamma)\epsilon/2})^2}. \tag{5.11}$$

For the ternary discretization used in PAC-EIG, $h_{\max} \leq \log(3)|\mathcal{S}||\mathcal{A}|^2$, giving a concrete bound on sample complexity.

Extension to trajectory queries. While the theorems establish a lower bound for single-state queries, this naturally extends to a trajectory-based version of our PAC-EIG acquisition function. When querying for a trajectory starting from state s_0, the information gained is at least as large as querying any single state visited along the trajectory. In practice, trajectories typically visit multiple informative states, potentially providing substantially more information than the theoretical lower bound suggests. However, an improved theoretical bound would need to build on additional assumptions about the environment and the prior.[3]

5.6 Experiments

We evaluated the performance of the two proposed acquisition functions in a set of gridworld experiments with respect to both objectives introduced earlier: the reward learning objective, measured by the entropy of the posterior distribution over rewards, and the apprenticeship learning objective, measured by the regret. We also track the posterior distribution over regrets which directly relates to the PAC criterion.

We evaluate across three types of environment:

1. **Structured gridworld**: Features fewer reward parameters than states. It includes a known goal state with a reward of +100, neutral states with a reward of -1, and three obstacle types with unknown negative rewards with a uniform prior between -100 and 0 independently for each obstacle type. This was meant as an illustrative example (Figure 5.1) and a counterexample to the only prior method designed for collecting full trajectories, *action entropy* [162], by including a jail state where all actions are equivalent and which always gets selected by this baseline, thus preventing any useful learning.

2. **10x10 random gridworld with 2 initial states**: Each state has a random reward drawn from the prior, $\mathcal{N}(0, 3)$, with only two possible initial states to test the ability of methods to recognize only relevant parts of the state space. Here we use $\beta = 4$ so the expert behaves closely to optimal.

[3]For example, the environment may terminate after a single step, forcing the EIG from a trajectory to that of a single state.

3. **8x8 random gridworld with fully uniform initial states**: Each state has a random reward drawn from the prior, $\mathcal{N}(0, 3)$, and the initial state distribution is uniform across all states. We use $\beta = 2$ so the expert is fairly stochastic.

We evaluate both the setting where each query results in a full expert trajectory, where we compare against the only prior method [162] as well as random sampling, and the setting where each query results in a single state-action annotation, where we also evaluate against the methods by Lopes et al. [160] and Brown et al. [161].

Each experiment type was run with with 16 different random reward functions, different terminal states (except for the jail environment) and different 2 initial states in the 10x10 environment. The plots display the mean and the standard error across these 16 random instances.

5.6.1 Results

Results on the simple environment from Figure 5.1 illustrate a crucial failure mode of the *action entropy* [162] acquisition function – it always queries the jail state and thus fails to learn anything useful, while both reward and regret EIG learn an optimal policy within 10 steps in all 16 instances with similar posterior entropies.

Figure 5.3 shows the results on the 10x10 gridworld with 2 random initial states and single-state annotations, Figure 5.4 shows the results for querying single-state annotations on the 8x8 gridworld with a uniform initial state distribution, and Figure 5.5 results on the 8x8 environment when querying trajectories of maximum length 5.

In the case of only 2 initial states on the 10x10 gridworld, we can see our regret-focused PAC-EIG acquisition function to do much better in terms of both actual and posterior regret, reaching a zero true regret, as well as a 0.1-0.1-PAC apprentice policy, by step 50. ActiveVaR remains competitive with RewardEIG in terms of the reward-learning objective (entropy of the rewardposterior), but both fall behind PAC-EIG in terms of true and posterior regret.



(a) Posterior reward entropy **(b)** True regret **(c)** Posterior probability of regret > 0.1

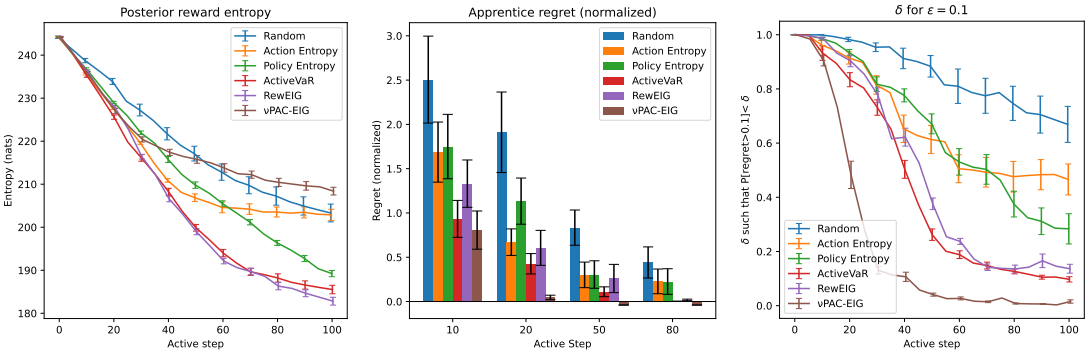
Figure 5.2: Results of the experiments on the environment with 3 cell types and a jail state with full-trajectory demonstrations.

On the gridworld with fully uniform initial states, we observe that both our information theoretic acquisition functions result in lower posterior reward entropy *and* lower regret than prior methods except for ActiveVaR, which seems to roughly match their performance. Interestingly, the reward-based acquisition function and the regret-based ones seem to perform similarly well on both objectives, suggesting that there is a strong correlation between learning about the reward and learning about the apprentice regret in this environment with uniform initial states.

The action entropy acquisition function still stops yielding significant improvements after about step 50 – it again gets stuck querying states that have high action entropy due to multiple actions being similarly good, even if these states do not yield any more information.

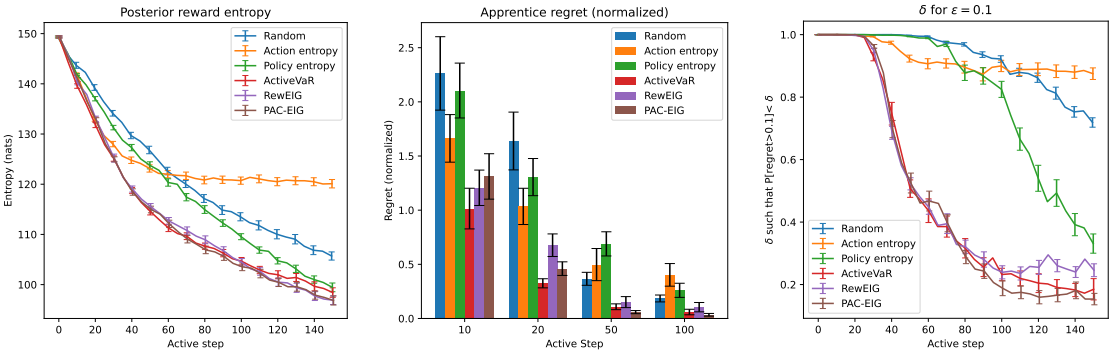
5.7 Discussion and Conclusion

In this paper we have proposed new acquisition functions for active IRL, each geared toward one of two possible objectives: learning about an unknown reward function, or producing a well-performing apprentice policy. We have shown that across a set of gridworld experiments, our acquisition functions outperform or at least match prior methods on their respective objective. Furthermore, our immediate-regret EIG acquisition function is a first acquisition function with a regret bound in our setting. While we have so far tested the methods only in



(a) Posterior reward entropy **(b)** True regret **(c)** Posterior prob. of regret > 0.1

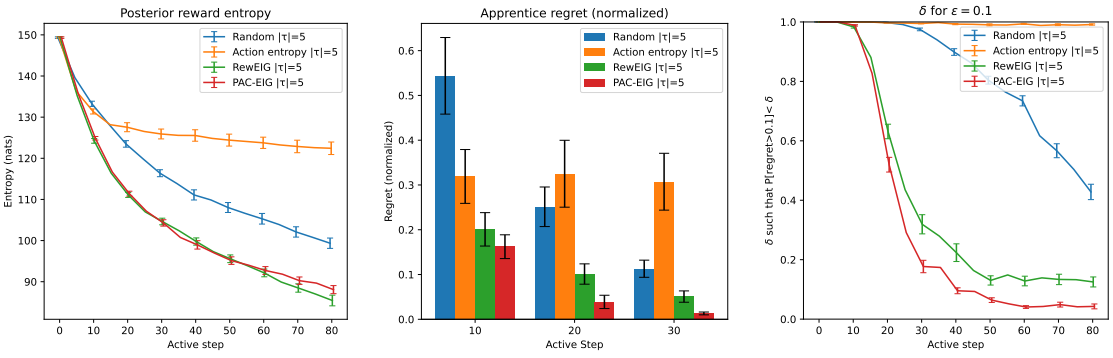
Figure 5.3: Results of the experiments with single state annotations (i.e. $|\tau| = 1$) on the 10x10 fully random gridworld with two initial states. In the barplot (b), results with zero regret are visualized below the horizontal axis to make their presence clearer.



(a) Posterior reward entropy **(b)** True regret **(c)** Posterior prob. of regret > 0.1

Figure 5.4: Results of the experiments with single state annotations (i.e. $|\tau| = 1$) on the 8x8 fully random gridworld.

finite state spaces, both of them were constructed to generalize also to continuous spaces, which will be addressed in future work.



(a) Posterior reward entropy

(b) True regret

(c) Posterior prob. of regret > 0.1

Figure 5.5: Results of the experiments with expert trajectories of maximum length $|\tau| = 5$ on the 8x8 fully random gridworld.

6
Conclusion

I have presented three advances in Bayesian IRL, each representing the current frontier in its respective area: first, when samples from the true posterior over reward functions is needed, ValueWalk presents a way to obtain MCMC samples from the posterior at a significant speed improvement relative to the previously standard PolicyWalk algorithm allowing such samples to be generated for larger problem settings. Nevertheless, MCMC sampling can still struggle to scale to higher-dimensional parameter spaces. This is where variational inference can help. While variational inference has been suggested for the Bayesian IRL problem before, the prior method sacrificed meaningful uncertainty quantification. Our Q-based variational inference method combines useful, albeit approximate, uncertainty quantification with scalability in settings where MCMC methods are not practical. Finally, we provide two principled methods for *active* IRL to make efficient use of an expert providing demonstrations – the first active IRL method to offer the guarantee of producing a probably approximately correct apprentice policy.

Still, plenty of work remains to be done on Bayesian IRL. ValueWalk provides random samples from the full posterior – yet, not all samples are equally important. Especially if we want to produce a risk-averse apprentice policy, we may care more about samples in the lower tail of the reward distribution and we may wish to adapt our sampling procedure accordingly and focus computation where

it matters for producing a safe apprentice policy, in line with the paradigm of *probabilistic numerics* [142].

The variational inference approach of QVIRL presented in Chapter 4 is still somewhat limited by the pre-specified variational family. It would be worth testing additional approaches to approximate inference in this context, such as Stein inference and approaches based on alternative divergences such as the Wasserstein distance, and studying which method best captures the features of the posterior distribution relevant for producing safe policies or for active learning.

On the active learning front, a notable gap that we are now working to fill is extending the experimental work to larger problems and continuous state spaces, including the PAC result. We have also explored using Bayesian optimization to effectively find the state that maximizes the expected information gain (EIG) by dynamically estimating EIG more precisely in states that are candidates for the maximizer rather than those that seem clearly inferior. While offering some computational efficiency benefits, such a procedure ultimately proved unnecessary in the small-scale experiments performed in the paper, but is likely to be needed in larger settings.

More broadly, the work presented in this thesis has been done in the context of vanilla IRL, which assumes a first-person perspective reward (i.e. the learner optimizing, from its standpoint, the same reward as the expert), in most cases with knowledge of the environment dynamics as well as the same degree of knowledge and environment observability in the expert and the learner. We think the Bayesian approach to IRL can offer (and in some cases has already been shown to offer) advantages in solving the issues resulting from dropping these assumptions, so I think it is worth extending work on Bayesian IRL in these directions.

Also, while I have explored Bayesian IRL in a number of simulated toy domains, further work will be needed in applying this work to real-world systems. Natural areas where IRL has been applied are robotics and autonomous driving, which could benefit from the uncertainty modelling that Bayesian methods offer. Presently, there is a lot of interest in systems based on large language models (LLMs) where

IRL has also recently been applied to extract a reward model from a corpus of text treated as demonstration data [61] as an alternative to supervised finetuning (i.e. behavioural cloning), and in a different strand of work, Bayesian reward modelling has been shown to reduce pathologies such as jailbreaks [178]. We may take our work further in a related direction, or would be excited to see others try.

By advancing the frontier of Bayesian inverse reinforcement learning through improved sampling efficiency, scalable variational inference, and principled active learning methods, this thesis makes meaningful steps toward future AI systems that can not only learn what humans want but also understand when to remain uncertain – a critical capability for both performance and safety in increasingly autonomous agents.

Appendices

A

Appendix to Chapter 3 (ValueWalk)

Contents

A.1 Unknown Transition Probabilities 93
A.2 Proof of Soundness of the Algorithm 94
A.3 Variations of the Baseline Methods 96
A.3.1 PolicyWalk-HMC 96
A.3.2 Model-Based AVRIL 96
A.4 Experiment Details 97
A.5 Additional Details of Results 98
A.5.1 Gridworld Experiments 98

A.1 Unknown Transition Probabilities

Section 3.3.1 presents a version of the ValueWalk algorithm for finite state and action spaces that assumes known transition probabilities. However, the key trick used in ValueWalk extends to unknown transition probabilities as well.

One simplified option to handle unknown transitions, also employed in the continuous-state case in Section 3.3.2 matching the setting used by AVRIL, is replacing the transition probabilities with their empirical estimate $\hat{p}(s'|s,a) = \xi(s,a,s')/\xi(s,a)$ where $\xi(s,a,s'), \xi(s,a)$ are the numbers of occurrences in the set of demonstrations of the transition (s,a,s') and state-action pair (s,a). In the

finite-state case, this would mean limiting the evaluation of the prior in Algorithm 3 to only those state-action pairs that do occur in the data (i.e. replacing vectors and matrices on lines 3–6 by the appropriate sub-vectors and sub-matrices).

A more principled Bayesian alternative is of course using full Bayesian inference also over transitions – in that case, we can perform the MCMC sampling jointly over both the transitions and the Q function parameters, recovering samples from the full joint posterior. The changes needed are (1) treating parameters of the transition model as inputs in the algorithm, (2) adding a prior over those parameters (so the joint prior will be a combination of the transition-parameter prior and the Q-parameter prior – which in turn depends on both the reward prior and the transition prior), and (3) including transition probabilities in the likelihood. Here is the adaptation of the finite-space algorithm to this case of unknown probabilities:

Algorithm 3: Calculation of the unnormalized posterior for finite $\mathcal{S}$ and $\mathcal{A}$ with unknown transition probabilities (performed in each step of HMC). The resulting candidate reward sample $\bar{R}$ is then accepted/rejected together with the corresponding Q and P.

Data: a candidate matrix of Q-values, a candidate transition matrix P, set of expert demonstrations $\mathcal{D}$, prior over rewards p_R, prior over transitions p_P

1 **for** $s, s' \in \mathcal{S}, a, a' \in \mathcal{A}$ **do**

2 $\quad \pi(a|s) = \exp(\bar{\alpha}Q(s,a))/\sum_{a'' \in \mathcal{A}} \exp(\bar{\alpha}Q(s,a''))$;

3 $\quad \bar{P}(s,a;s',a') = P(s'|s,a)\pi(a'|s')$;

4 **end**

5 $\bar{R} = (I - \gamma\bar{P})\bar{Q}$ where $\bar{R}, \bar{Q}$ are flattened vector versions of the reward and Q-value matrices ;

6 $p_Q(Q) = p_R(\bar{R}) \det(I - \gamma\bar{P})$;

7 $p(\mathcal{D}|Q,P) = \prod_{(s,a,s') \in \mathcal{D}} P(s'|s,a) \exp(\alpha Q(s,a))/\sum_{a' \in \mathcal{A}} \exp(\alpha Q(s,a'))$;

Result: $p(Q,P|\mathcal{D}) \propto p_P(P)p_Q(Q)p(\mathcal{D}|Q,P)$; candidate sample $\bar{R}$

A.2 Proof of Soundness of the Algorithm

Theorem 1. *Assume that the transition kernel q_Q satisfies the detailed balance condition*

$$\frac{q_Q(Q'|Q)}{q_Q(Q|Q')} = \frac{p_Q(Q'|\mathcal{D})}{p_Q(Q|\mathcal{D})}$$

with respect to the posterior over Q-values defined in Algorithm 1. Then the associated implicit Markov chain over rewards also satisfies the detailed balance condition with respect to the posterior $p_R(R|\mathcal{D})$.

Proof. Let q_Q be the transition kernel over Q-values that satisfies the detailed balance condition with respect to the posterior $p_Q(Q|\mathcal{D})$ as assumed in the theorem statement.

The implicit transition kernel q_R over rewards induced by q_Q can be expressed as

$$q_R(R'|R) = q_Q(Q(R')|Q(R)) \left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right| \tag{A.1}$$

where $Q(R) = (I - \gamma\bar{P})^{-1}R$ is the Q-value corresponding to reward R as used in Algorithm 1. The determinant term accounts for the change of variables from Q to R. Note that $\bar{P}$ depends on R but changes only when the optimal policy changes, so it is constant almost everywhere as a function of R, so Q is locally linear in R and the Jacobian exists almost everywhere.

The posterior over rewards can be expressed in terms of the posterior over Q-values as

$$p_R(R|\mathcal{D}) = p_Q(Q(R)|\mathcal{D}) \left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right| = p_Q(Q(R)|\mathcal{D}) \left|\det(I - \gamma\bar{P})^{-1}\right|. \tag{A.2}$$

Now consider the ratio of the implicit transition kernel:

$$\frac{q_R(R'|R)}{q_R(R|R')} = \frac{q_Q(Q(R')|Q(R))}{q_Q(Q(R)|Q(R'))} \frac{\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}{\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|} = \frac{p_Q(Q(R')|\mathcal{D})}{p_Q(Q(R)|\mathcal{D})} \frac{\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}{\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|} \tag{A.3}$$

where the second equality follows from the assumed detailed balance condition on q_Q.

We know from above that

$$p_R(R|\mathcal{D}) = p_Q(Q(R)|\mathcal{D}) \left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|$$

and

$$\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right| = |\det((I - \gamma\bar{P})^{-1})| = \frac{1}{|\det(I - \gamma\bar{P})|}.$$

The same holds for R'.

Substituting these into Equation A.3, we obtain

$$\begin{aligned}\frac{q_R(R'|R)}{q_R(R|R')} &= \frac{p_Q(Q(R')|\mathcal{D})}{p_Q(Q(R)|\mathcal{D})}\frac{\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}{\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|} \\ &= \frac{p_R(R'|\mathcal{D})/\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}{p_R(R|\mathcal{D})/\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|}\frac{\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}{\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|} \\ &= \frac{p_R(R'|\mathcal{D})}{p_R(R|\mathcal{D})}\cdot\frac{\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|}{\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}\cdot\frac{\left|\det\left(\frac{\partial Q(R')}{\partial R'}\right)\right|}{\left|\det\left(\frac{\partial Q(R)}{\partial R}\right)\right|} \\ &= \frac{p_R(R'|\mathcal{D})}{p_R(R|\mathcal{D})} \quad \text{(A.4)}\end{aligned}$$

Thus, the implicit Markov chain over rewards induced by the transition kernel q_Q satisfies detailed balance with respect to the posterior $p_R(R|\mathcal{D})$, as claimed. □

A.3 Variations of the Baseline Methods

A.3.1 PolicyWalk-HMC

We proposed that ValueWalk be used with Hamiltonian Monte Carlo (HMC) treating the underlying parameters as fully continuous. By contrast, PolicyWalk, as originally proposed, samples the next proposed value of the reward parameters from neighbours of the current point on a discretized grid. To isolate the speed-up effect of our Q-space trick from the speed-up due to HMC, we also implemented a version of PolicyWalk with HMC (denoted by PolicyWalk-HMC in the paper). This involves calculating the gradient of the posterior with respect to the reward parameters. To do that, we use the matrix-multiplication computed Q-values. We omit the contribution of the combined transition-policy matrix to the gradient, since the derivative of the optimal policy with respect to the reward is zero almost everywhere.

A.3.2 Model-Based AVRIL

In the gridworld experiments, both PolicyWalk and ValueWalk are leveraging the environment dynamics, which AVRIL does not use. For fairer comparison, we are thus including also a model-based version of AVRIL, which differs from the original (model-free) AVRIL in that it evaluates the KL divergence from the prior across

all states (the gridworld is using state-only rewards), the TD term is calculated over state-action pairs, and the next-state value can be estimated using the actual expectation, instead of just using the Q-value of the next empirical state-action pair from the demonstrations. The remainder of the algorithm remains the same.

A.4 Experiment Details

For the gridworld experiments, we used a version of AVRIL learning a Q-value for each state-action pair and a mean and variance value for the reward in each state. For PolicyWalk, we ran inference over a reward vector containing a reward value for each state. For ValueWalk, we ran inference over the state-value vector.

In the continuous state space environments, for the 3 continuous baseline methods, we match the setup from Chan and van der Schaar [102] and use neural network models with 2 hidden layers of 64 units and an ELU activation function. For our experiments, we scale up the network size with the complexity of the problem: we use one hidden layer with 8 units for Cartpole, 1 layer of 16 units for Acrobot, and 2 layers of 24 units for LunarLander. In each case, we also tried running AVRIL with a matching network size but in each case it performed similarly or usually worse than the default 2x64 setup for which results are reported.

For PolicyWalk-HMC and ValueWalk, we use the Pyro [179] implementation of HMC+NUTS. For the control environment experiments, we ran with 4,000 warm-up steps and 10,000 inference steps for Lunar Lander and 2,000 warm-up steps with 5,000 inference steps for Cartpole and Acrobot (since we are inferring fewer network parameters there). We automatically tune the step size during warm-up but do not tune the mass matrix.

In the continuous environments, we use a Gaussian process prior with an RBF kernel with fixed scale of 1 and fixed lengthscale of 0.2 for Cartpole and Acrobot and 0.03 for Lunar Lander (chosen manually based on the distribution of features in the demonstrations for each environment, where the lengthscale roughly corresponds to the std of one-step change in each feature).

In Cartpole, Acrobot, and Lunar Lander, we reuse the demonstration sets provided by the authors of AVRIL. Each contains 1,000 demonstration trajectories, from which we randomly chose a set of 100 test trajectories and then split the remaining examples into 5 training splits. We then re-ran each experiment for each number $n_{\text{traj}} = 1, 3, 7, 10, 15$ of trajectories on the first n_{traj} trajectories of each of the 5 splits and evaluated the resulting apprentice agent on 300 episodes of the environment. We report the mean and std across the splits and evaluations.

Unless otherwise stated, we use a Boltzmann rationality coefficient of 3.

A.5 Additional Details of Results

A.5.1 Gridworld Experiments

Figure A.1 shows 2-D histograms of pairwise joint posteriors over rewards of the 9 states of the gridworld. Two aspects of the expert's behaviour are captured by this plot and may not be obvious from the simple histograms in Figure 3.1. Firstly, the agent heading to the terminal top right corner can be explained either by the reward there being positive, or by the reward in other states being negative, and thus the agent using the terminal state as a way to escape incurring further negative rewards. Secondly, note that practically all of the probability mass is placed on the reward of the obstacle tile being lower than that of the two tiles below, thus explaining the expert avoiding the obstacle tile.

The plot also clearly shows that the posterior is non-Gaussian (note especially the sharp edge expressing high confidence that the ratio of the two values does not cross a certain threshold) and thus could not be captured by the Gaussian variational family. Also, the rewards of different states are, sometimes very tightly, correlated, so modelling them as independent would again be inappropriate.

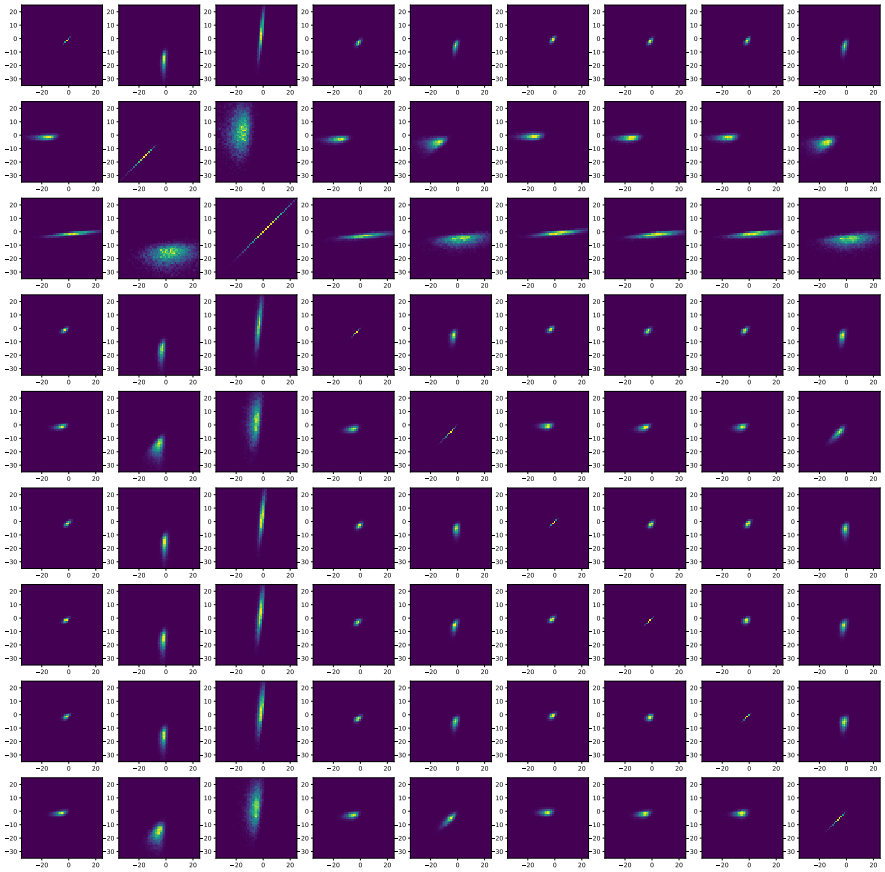


Figure A.1: 2-D histograms representing the joint posteriors of the rewards associated with the 9 states of the gridworld (enumerated left-to-right, top-to-bottom, so state 3 is the goal state in the top right corner).

B

Appendix to Chapter 4 (QVIRL)

Contents

B.1 Method Details 101
B.1.1 Evaluating Expectations over Next States 101
B.1.2 Clark Approximation to the State-Value Distribution . 102
B.1.3 KL and the Loss 108
B.1.4 State-Only Reward Variant 109
B.2 Approximations: Further Notes and Illustration 110
B.2.1 Max-Mean Approximation 110
B.2.2 Ablation: State-Value Approximation and KL Term . . 111
B.2.3 Likelihood Approximation 113
B.2.4 Gaussianity 114
B.3 Additional Discussion on AVRIL 115
B.3.1 Dynamics-Aware AVRIL 115
B.3.2 3×3 Gridworld and a Comparison to AVRIL 116
B.3.3 Limitations of AVRIL 117
B.4 Experiment Details 119
B.4.1 Environment Descriptions 119
B.4.2 Expert Demonstrations 121
B.4.3 QVIRL Training 122
B.4.4 Baselines on ATARI 123
B.4.5 Evaluation on ATARI 124
B.4.6 Active Learning 124
B.4.7 Prior Distribution 125
B.5 Additional Experiments 127
B.5.1 Sensitivity of QVIRL to Rationality Coefficient β 127
B.5.2 Illustrative Gridworld Experiment with a Single Unknown State 128
B.6 Code, Data, and Compute 129
B.6.1 Code and Data 129

B.6.2 Computing Resources Used 130

B.1 Method Details

B.1.1 Evaluating Expectations over Next States

Let us first discuss how to evaluate the expectation over next states in the inverse Bellman equation (Eq. 4.3 in Chapter 4) used to derive the reward distribution from the Q posterior:

$$R(s,a) = Q^*(s,a) - \gamma\, \mathbb{E}_{s' \sim p(\cdot|s,a)} \hat{V}^*(s'). \tag{B.1}$$

$\hat{V}^*(s') \approx \max_{a'} Q^*(s',a')$ is the random variable approximating the state-value; its construction is the subject of Section B.1.2. For terminal transitions, we set $\gamma = 0$, so that $R(s,a) = Q^*(s,a)$ exactly.

Full Probabilistic Dynamics Model

In discrete environments, if a probabilistic transition model is available (i.e. we either know the environment dynamics or have some estimated model), we can compute the expectation exactly, by writing the expectation over successor states as a sum weighted by the transition probabilities:

$$R(s,a) = Q^*(s,a) - \gamma \sum_{s'} p(s'|s,a) \hat{V}^*(s'). \tag{B.2}$$

This is the version that we use in gridworlds, since we assume their dynamics to be fully known.

In continuous spaces with a transition model, the sum becomes an integral

$$R(s,a) = Q^*(s,a) - \gamma \int_{s' \in \mathcal{S}} p(s'|s,a) \hat{V}^*(s')\, ds', \tag{B.3}$$

which, in general, needs to be approximated numerically. One option is standard Monte Carlo approximation, where we sample a set of possible successor states $s' \sim p(s'|s,a)$ and approximate the integral by their empirical mean. The advantage of this option is that we do not need the full probabilistic model of the environment,

but in fact need only access to a simulator that can provide such next-state samples given a state-action pair.

Empirical Transitions

In case we cannot sample from a transition model for an arbitrary state-action pair, we need to rely on empirical transitions, i.e. we assume we have access to a set of empirical samples (s, a, s'). Such transitions are part of the expert demonstration dataset, but note that the inverse Bellman equation holds for *any* action, not just the expert ones, so such transitions can come from other sources including rollouts of the apprentice policy, a random policy, or any other policy. Such rollouts can start according to the initial state distribution or any other distribution. Such data can either be generated online, during training, or come from a static dataset. The key property we are building on is just the assumption that the data come from the environment of interest, i.e. $s' \sim p(s'|s, a)$.

Whatever the provenance of the data, the expectation then gets crudely approximated as

$$\mathbb{E}_{s''\sim p(\cdot|s,a)}\hat{V}^*(s'') \approx \hat{V}^*(s') \;, \tag{B.4}$$

which can be interpreted as a single-sample version of the Monte Carlo approximation from the previous section. Note that this approximation becomes exact in deterministic environments. Moreover, such single-sample estimates are not used in isolation: with enough data, the dataset contains many transitions from proximate states – or even the same state – which improves the quality of the approximation for the purposes of the training signal.

B.1.2 Clark Approximation to the State-Value Distribution

This subsection details how we approximate the distribution of $V^*(s) = \max_a Q^*(s, a)$ under the modelling assumption of our method that the optimal Q-values are distributed as a multivariate Gaussian. At its core, this is a standard application of Clark's [156] moment-matching recursion (see Hennig [180] for a related moment-matching treatment of correlated-Gaussian maxima in machine learning); in addition,

we describe how the covariances of the resulting approximate state-value $\hat{V}^*(s)$ with the other Gaussian quantities of the model – the optimal Q-values and the approximate state-values of other states – follow from the same recursion in closed form. Note that the max-mean approximation $V^*(s) \approx Q^*(s, \text{argmax}_a \mu_{Q^*}(s,a))$ provides a much simpler and easier-to-implement alternative, which still performs fairly well (Appendix B.2). When reimplementing this algorithm, we recommend max-mean as a starting point, extending to Clark for slightly higher fidelity once the max-mean version is working.

Moment recursion. Fix a state s and let us locally shorten the notation to $\mu_{Q,i} := \mu_{Q^*}(s,a_i)$ and $\sigma_{Q,i} := \sigma_{Q^*}(s,a_i)$ for the marginal mean and standard deviation of $Q^*(s,a_i)$; the covariances between the Q-values are those implied by Equation 4.2.

Setting $M_1 := Q^*(s,a_1)$ and $M_i := \max\Big(M_{i-1}, Q^*(s,a_i)\Big)$ for $i = 2, \dots, |\mathcal{A}|$, and assuming that $M_{i-1} \approx \mathcal{N}(\mu_{M,i-1}, \sigma^2_{M,i-1})$, Clark approximates the distribution of M_i by $\mathcal{N}(\mu_{M,i}, \sigma^2_{M,i})$ where

$$\begin{aligned}
\mu_{M,i} &= \mu_{M,i-1}\,\Phi(\nu_i) + \mu_{Q,i}\,\Phi(-\nu_i) + \omega_i\,\phi(\nu_i) \\
\sigma^2_{M,i} &= (\mu^2_{M,i-1} + \sigma^2_{M,i-1})\,\Phi(\nu_i) + (\mu^2_{Q,i} + \sigma^2_{Q,i})\,\Phi(-\nu_i) \\
&\quad + (\mu_{M,i-1} + \mu_{Q,i})\,\omega_i\,\phi(\nu_i) - \mu^2_{M,i}
\end{aligned}$$

where

$$\omega_i^2 = \sigma^2_{M,i-1} + \sigma^2_{Q,i} - 2\,c_i\,, \quad \nu_i = (\mu_{M,i-1} - \mu_{Q,i})/\omega_i,$$

Φ and ϕ are respectively the CDF and PDF of the standard normal distribution, and $c_i := \text{Cov}\Big(M_{i-1}, Q^*(s,a_i)\Big)$ is the covariance between the running maximum and the next Q-value, whose computation we address next. We then approximate the distribution of the final maximum, i.e. the state value V^*, by the Gaussian obtained at the last iteration, $\hat{V}^*(s) \sim \mathcal{N}\Big(\mu_{V^*}(s), \sigma^2_{V^*}(s)\Big)$ with $\mu_{V^*}(s) := \mu_{M,|\mathcal{A}|}$ and $\sigma^2_{V^*}(s) := \sigma^2_{M,|\mathcal{A}|}$.

Covariance propagation. Because each running maximum M_{i-1} is itself correlated with the remaining Q-values, the covariances c_i cannot be read directly off Equation 4.2; they must be propagated alongside the moments. Clark's method supplies the matching update: for *any* variable Z jointly Gaussian with $Q^*(s,\cdot)$,

$$\mathrm{Cov}(M_i, Z) = \Phi(\nu_i)\,\mathrm{Cov}(M_{i-1}, Z) + \Phi(-\nu_i)\,\mathrm{Cov}\Big(Q^*(s,a_i), Z\Big), \tag{B.5}$$

initialized by $\mathrm{Cov}(M_1, Z) = \mathrm{Cov}\Big(Q^*(s,a_1), Z\Big)$. Taking $Z = Q^*(s,a_i)$ and carrying the recursion to step $i-1$ yields the c_i required by the moment recursion; other choices of Z will yield all the cross-covariances required by our method.

Closed form of the propagated covariances. The update in Equation B.5 is linear in the covariances, so it can be unrolled into a closed form. Collecting the factors $\Phi(\nu_i)$ accumulated across the sweep into the weights

$$w_1(s) := \prod_{i=2}^{|\mathcal{A}|} \Phi(\nu_i), \qquad w_j(s) := \Big(1 - \Phi(\nu_j)\Big) \prod_{i=j+1}^{|\mathcal{A}|} \Phi(\nu_i) \quad \text{for } j \geq 2, \tag{B.6}$$

which satisfy $w_j(s) \geq 0$ and $\sum_j w_j(s) = 1$, the covariance at the final iteration becomes

$$\mathrm{Cov}\Big(\hat{V}^*(s), Z\Big) = \sum_{j=1}^{|\mathcal{A}|} w_j(s)\,\mathrm{Cov}\Big(Q^*(s,a_j), Z\Big) \tag{B.7}$$

for any Z jointly Gaussian with the Q-values. In other words, for the purpose of computing covariances, $\hat{V}^*(s)$ behaves like the convex combination $\sum_j w_j(s)\,Q^*(s,a_j)$ of the state's Q-values – a 'soft argmax' whose weights reflect how likely each action is to attain the maximum. We stress that this linear surrogate is used only for the covariances: the mean and variance of $\hat{V}^*(s)$ are the moment-matched values $\mu_{V^*}(s)$ and $\sigma^2_{V^*}(s)$ above, both of which the surrogate would generally underestimate.[1]

[1]The surrogate mean $\sum_j w_j(s)\,\mu_{Q,j}$ is a convex combination of the action means, so it drops the option-value terms $\omega_i\,\phi(\nu_i)$ and lies below even $\max_j \mu_{Q,j}$; the surrogate variance ignores the variance contributed by the uncertainty about *which* action attains the maximum.

The induced reward posterior. Substituting the Gaussian approximation of the maximum into the inverse Bellman equation (Equation 4.3) replaces the intractable nonlinear maximum by the Gaussian $\hat{V}^*(s')$:

$$R(s,a) = Q^*(s,a) - \gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}\Big[\hat{V}^*(s')\Big]. \tag{B.8}$$

The reward is then an affine functional of jointly Gaussian variables – the optimal Q-value $Q^*(s,a)$ and the successor values $\hat{V}^*(s')$ – so the induced reward posterior is itself Gaussian, with moments given by the following proposition.

Proposition 4. *Conditional on the Clark approximations* $\hat{V}^*(s') \sim \mathcal{N}\Big(\mu_{V^*}(s'), \sigma^2_{V^*}(s')\Big)$*, with covariances given by Equation B.7, the reward* $R(s,a)$ *defined by Equation B.8 from the optimal Q-values with mean and covariance from Equation 4.2 is normally distributed with mean*

$$\mu_R(s,a) = \mu_{Q^*}(s,a) - \gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}\Big[\mu_{V^*}(s')\Big] \tag{B.9}$$

and variance

$$\sigma^2_R(s,a) = \sigma^2_{Q^*}\Big(s,a\Big) - 2\gamma\, \mathbb{E}_{s'\sim p(\cdot|s,a)}\Big[C_{QV}(s,a;s')\Big] + \gamma^2 \mathbb{E}_{\substack{s'\sim p(\cdot|s,a)\\ s''\sim p(\cdot|s,a)}}\Big[C_{VV}(s',s'')\Big] \tag{B.10}$$

where s' *and* s'' *are independent draws from* $p(\cdot|s,a)$*, and the covariances of the successor value with, respectively, the optimal Q-value at* (s,a) *and the successor value at another sampled state are the bilinear forms*

$$C_{QV}(s,a;s') := \mathrm{Cov}\Big(Q^*(s,a), \hat{V}^*(s')\Big) = \sum_j w_j(s')\, \mathrm{Cov}\Big(Q^*(s,a),\, Q^*(s',a_j)\Big), \tag{B.11}$$

$$C_{VV}(s',s'') := \mathrm{Cov}\Big(\hat{V}^*(s'), \hat{V}^*(s'')\Big) = \sum_{j,l} w_j(s')\, w_l(s'')\, \mathrm{Cov}\Big(Q^*(s',a_j),\, Q^*(s'',a_l)\Big) \tag{B.12}$$

for $s' \neq s''$*, with the weights of Equation B.6 and the Q-covariances of Equation 4.2. When* $s'' = s'$ *the two values coincide and* $C_{VV}(s',s') = \sigma^2_{V^*}(s')$*.*

Proof. We use the fact that if a random vector X in $\mathbb{R}^n$ has a multivariate normal distribution with mean μ and covariance Σ, and $A \in \mathbb{R}^{m\times n}$ is a matrix, then

$Y = AX$ also has a multivariate normal distribution with mean $A\mu$ and covariance $A\Sigma A^\top$.

Conditional on the Clark approximations, the successor values $\hat{V}^*(s') \sim \mathcal{N}\Big(\mu_{V^*}(s'), \sigma^2_{V^*}(s')\Big)$ are jointly Gaussian with the optimal Q-value $Q^*(s,a)$. Collecting them into the vector

$$\mathbf{z} = \Big(Q^*(s,a);\ \hat{V}^*(s')\Big|_{s'\in\mathcal{S}}\Big)^\top,$$

Equation B.8 can be written as the affine map

$$R(s,a) = A\mathbf{z}\ , \qquad A = \Big(1,\ -\gamma\, p(s_1|s,a),\ \ldots,\ -\gamma\, p(s_n|s,a)\Big),$$

so $R(s,a)$ is normally distributed with mean $A\mu_{\mathbf{z}}$ and variance $A\Sigma_{\mathbf{z}}A^\top$, where $\mu_{\mathbf{z}}$ and $\Sigma_{\mathbf{z}}$ are the mean and covariance of $\mathbf{z}$.

The mean of $\mathbf{z}$ is $\mu_{\mathbf{z}} = \Big(\mu_{Q^*}(s,a);\ \mu_{V^*}(s')\Big|_{s'\in\mathcal{S}}\Big)^\top$, and $A\mu_{\mathbf{z}}$ gives Equation B.9 directly. The covariance $\Sigma_{\mathbf{z}}$ has the block structure

$$[\Sigma_{\mathbf{z}}]_{00} = \sigma^2_{Q^*}(s,a), \quad [\Sigma_{\mathbf{z}}]_{0,s'} = C_{QV}(s,a;s'), \quad [\Sigma_{\mathbf{z}}]_{s',s''} = C_{VV}(s',s''),$$

with C_{QV} and C_{VV} as defined in the proposition. Expanding $A\Sigma_{\mathbf{z}}A^\top$ then yields

$$\sigma^2_R(s,a) = \sigma^2_{Q^*}(s,a) - 2\gamma \textstyle\sum_{s'} p(s'|s,a)\, C_{QV}(s,a;s') + \gamma^2 \sum_{s'}\sum_{s''} p(s'|s,a)\, p(s''|s,a)\, C_{VV}(s',s''),$$

which is Equation B.10 written with the successor expectations expressed as sums over the (known) dynamics; the sampled-dynamics cases of Section B.1.1 replace these sums by their Monte Carlo or single-sample estimates.

It remains to justify the covariance entries. Applying Equation B.7 at s' with $Z = Q^*(s,a)$ gives Equation B.11. Applying it at s' with $Z = \hat{V}^*(s'')$, and then once more at s'' with $Z = Q^*(s',a_j)$ for each j, gives Equation B.12. (Equivalently, both can be obtained by carrying the second variable as an external term through the covariance recursion of Equation B.5; the linearity of the recursion makes the two computations identical.) When $s'' = s'$, the exact Clark variance $\sigma^2_{V^*}(s')$ from the moment recursion is used instead of the surrogate value $\sum_{j,l} w_j(s')\, w_l(s')\, \mathrm{Cov}\Big(Q^*(s',a_j), Q^*(s',a_l)\Big)$, which would underestimate it. $\square$

The proposition supplies the marginal reward moments used by the per-transition KL terms. When the KL is instead evaluated *jointly* over a batch of transitions (Appendix B.1.3), we also need the covariances between the rewards at different state-action pairs. These follow from the same affine-map argument; we state the result for the empirical-transition setting of Section B.1.1, in which the joint KL is used and each pair comes with a single sampled successor state.

Corollary 5 (Joint reward posterior)**.** *Let* $\{(s_i, a_i)\}_{i=1}^m$ *be state-action pairs with respective successor states* $\{s'_i\}_{i=1}^m$ *and per-pair discounts* γ_i *(where* $\gamma_i = \gamma$*, except* $\gamma_i = 0$ *for a terminal transition, making* $R_i = Q^*(s_i, a_i)$ *exactly). Conditional on the Clark approximations, the rewards* $R_i := Q^*(s_i, a_i) - \gamma_i \hat{V}^*(s'_i)$ *are jointly Gaussian with means* $\mu_R(s_i, a_i)$ *given by Equation B.9 and covariance matrix*

$$[\Sigma_R]_{ij} = \mathrm{Cov}\Big(Q^*(s_i, a_i), Q^*(s_j, a_j)\Big) - \gamma_j\, C_{QV}(s_i, a_i; s'_j) - \gamma_i\, C_{QV}(s_j, a_j; s'_i) + \gamma_i\gamma_j\, C_{VV}(s'_i, s'_j), \tag{B.13}$$

where for $i \neq j$ *the term* C_{VV} *is evaluated by Equation B.12 and on the diagonal by the exact Clark variance,* $C_{VV}(s'_i, s'_i) = \sigma^2_{V^*}(s'_i)$*, so that* $[\Sigma_R]_{ii}$ *recovers the marginal variance of Proposition 4.*

Proof. Identical to the proposition: stack

$$\mathbf{z} = \Big(Q^*(s_1, a_1), \ldots, Q^*(s_m, a_m);\, \hat{V}^*(s'_1), \ldots, \hat{V}^*(s'_m)\Big)^\top$$

and apply the affine map $A = \Big(I_m,\ -\mathrm{diag}(\gamma_1, \ldots, \gamma_m)\Big)$. □

Positive semi-definiteness. If every entry of Equation B.13, including the diagonal, is evaluated through the linear surrogate, then $\Sigma_R = B\,\Sigma_{\mathbf{q}} B^\top$, where $\Sigma_{\mathbf{q}}$ is the joint Gaussian covariance of all the involved Q-values and the i-th row of the weight matrix B assigns weight 1 to $Q^*(s_i, a_i)$ and $-\gamma_i\, w_j(s'_i)$ to each $Q^*(s'_i, a_j)$; Σ_R is therefore positive semi-definite by construction. Lifting the diagonal to the exact Clark variances, as in Corollary 5, amounts to adding the residuals $\gamma_i^2\Big(\sigma^2_{V^*}(s'_i) - \sum_{j,l} w_j(s'_i)\, w_l(s'_i)\, \mathrm{Cov}\Big(Q^*(s'_i, a_j), Q^*(s'_i, a_l)\Big)\Big)$ to the diagonal, clamped below at zero; since only non-negative values are added to the diagonal, positive semi-definiteness

is preserved.[2] This construction – propagating maxima of correlated Gaussians in a linear canonical form while moment-matching the marginal distributions – parallels the canonical delay models used to propagate correlated maxima at scale in statistical static timing analysis [181].

B.1.3 KL and the Loss

The loss (Eq. 4.7) contains the term $\mathrm{KL}(q_R(R;\theta)||p(R))$, whose purpose is to minimize the distance of the implicit variational posterior over rewards, $q_R(R;\theta)$, to the prior $p(R)$. In tabular environments, we use a multivariate Gaussian prior; in other environments, we assume, more generally, a Gaussian process prior, which can be evaluated on any finite set of state-action pairs to get a multivariate Gaussian prior over those. Similarly, Section 4.4.2 explains how we can obtain a Gaussian posterior over rewards corresponding to the trained variational posterior over Q-values.

Unless the state-action space is very large, the tabular case is trivial: the KL between two multivariate Gaussian distributions with respective means $\mu_1, \mu_2 \in \mathbb{R}^d$ and covariance matrices $\Sigma_1, \Sigma_2 \in \mathbb{R}^{d\times d}$ can be calculated analytically as

$$\mathrm{KL}\Big(\mathcal{N}(\mu_1,\Sigma_1)\,\|\,\mathcal{N}(\mu_2,\Sigma_2)\Big) = \tfrac{1}{2}\left[\mathrm{tr}\Big(\Sigma_2^{-1}\Sigma_1\Big) + (\mu_2-\mu_1)^\top\Sigma_2^{-1}(\mu_2-\mu_1) - d + \ln\tfrac{\det\Sigma_2}{\det\Sigma_1}\right], \tag{B.14}$$

where d is the dimension, $\mathrm{tr}(\cdot)$ the trace, and $\det(\cdot)$ the determinant. Taking $\mathcal{N}(\mu_1,\Sigma_1) = q_R(R;\theta)$ and $\mathcal{N}(\mu_2,\Sigma_2) = p(R)$, applied to the full joint prior and posterior, gives the required KL term of the loss function. Note that in this tabular case, the KL is evaluated across all state-action pairs, not just the expert trajectories. The ELBO-like loss thus preserves its natural balancing of the likelihood and KL inherited from Bayes' theorem.

In continuous spaces, we use a Gaussian process prior, and the variational posterior over Q-values implicitly induces a Gaussian posterior over rewards, whose joint mean and covariance over any batch of transitions are given by Corollary 5. The

[2] In practice, we additionally symmetrize Σ_R and add a small jitter to its diagonal before the Cholesky factorization used in the KL computation. A cheaper variant of the joint covariance would skip the diagonal lift and use the purely bilinear (surrogate) form throughout, at the cost of underestimating the marginal variances.

KL term should again encourage closeness between the reward prior and posterior everywhere, not just on the expert trajectories. We use a solution that is standard in VI: together with each mini-batch B_{expert} of expert data, we also sample a batch B_{aux} of auxiliary transitions – these can be sampled from rollouts of the current apprentice policy, an offline dataset, or just be sampled randomly from the state-action space, if we have the requisite level of access. We then calculate the KL from Eq. B.14 jointly over both the expert data and auxiliary data, i.e. over the joint batch $B_{\text{all}} = B_{\text{expert}} \cup B_{\text{aux}}$. We normalize per-point and weight this by a coefficient λ to combine this with the likelihood in the continuous, mini-batched version of the loss

$$\mathcal{L}_{\text{cont}}\left(B_{\text{expert}}, B_{\text{aux}}\right) = \frac{\lambda}{|B_{\text{all}}|}\text{KL}\left(q_R(R_{B_{\text{all}}};\theta)\middle\|p(R_{B_{\text{all}}})\right) - \sum_{s,a\in B_{\text{expert}}} \log p(a|s;\theta) \ , \tag{B.15}$$

where $R_{B_{\text{all}}}$ is a random vector representing the unknown reward at the state-action pairs in B_{all}, with the multivariate Gaussian $q_R(R_{B_{\text{all}}};\theta)$ given by Corollary 5.

Relative to the tabular case, this unfortunately introduces the hyperparameter λ, since we no longer get the natural weighting of the likelihood and KL implied by Bayes' theorem. Using trainable inducing points is an alternative that can preserve the weighting and may be worth exploring further, but constructing sufficiently dense inducing points in higher-dimensional spaces is challenging and can introduce other approximation errors.

B.1.4 State-Only Reward Variant

Some environments use rewards that are a function of only the state, not the full state-action pair. In theory, QVIRL can model this in its default state-action-dependent model using strong correlations between rewards of all actions in each state to approximate the actual perfect correlation arbitrarily closely. However, this can cause numerical issues – in the tabular case, QVIRL would be modelling uncertainty over the $|\mathcal{S}|$-dimensional reward space using a distribution over $|\mathcal{S}| \times |\mathcal{A}|$ Q-values, causing the joint posterior to be singular. However, this can easily be resolved by instead working in terms of optimal state values V^*.

In the tabular case, the variational posterior can be defined in terms of the means of those state values and their covariance matrix. The rewards can then be derived as

$$R(s) = V^*(s) - \gamma \max_a \mathbb{E}_{s'|s,a} V^*(s')$$

and their distribution then approximated as Gaussian using either of the two proposed approximations to the maximum of jointly Gaussian random variables: Clark or max-mean. The Q-values can then be derived using the linear equation

$$Q^*(s,a) = R(s) + \gamma \mathbb{E}_{s'|s,a} V^*(s')$$

yielding again an approximate posterior Gaussian distribution, which can be used to evaluate the likelihood using Lu's approximation (Eq. 4.6).

B.2 Approximations: Further Notes and Illustration

B.2.1 Max-Mean Approximation

In Equation 4.3, when approximating the reward distribution corresponding to the current posterior over optimal Q-values, we suggest, as a simpler alternative to Clark's approximation, replacing the distribution of the state-value of the next state, i.e. the maximum of the optimal Q-values for that state, by a closed-form surrogate in order to benefit from computational speed-ups at the cost of approximation error. While Chapter 4 proposes Clark's approximation, there is an even simpler alternative one could consider: replacing the maximization operation over successor-state optimal Q-values with the distribution of the optimal Q-value for the action with the highest posterior mean. We call this approximation the *max-mean* approximation.

More formally, in a Bellman backup, we replace

$$R(s,a) = Q^*(s,a) - \gamma \, \mathbb{E}_{s' \sim p(\cdot|s,a)} \left[\max_{a'} Q^*(s',a') \right]$$

with the following:

$$R(s,a) = Q^*(s,a) - \gamma \, \mathbb{E}_{s' \sim p(\cdot|s,a)} [Q^* \left(s', \text{argmax}_{a'} \mu_Q(s',a')\right)] \, . \tag{B.16}$$



Figure B.1: Effect of the approximation in Equation B.16 on the reward posterior. The plot shows the MCMC posterior distribution over rewards for each state, as well as samples of rewards deduced using the approximation from the MCMC posterior over state-values. Dashed lines show Gaussians fitted to the two MCMC samples. The results coincide except in the bottom left and middle squares.

Figure B.1 illustrates the effect of the approximation in a 3×3 gridworld. The approximation does not introduce any error in most states. This is because the posterior over optimal Q-values in the successor states to these states clearly favours a single action and thus the distribution of the state value coincides with the distribution of the highest-mean action. On the other hand, we can see the approximation introducing an error in two states at the bottom left. This is because of the algorithm's uncertainty about what the optimal action in the next state is, so the state-value does not coincide with the distribution of the max-mean optimal Q-value. Consequently, the algorithm slightly underestimates the expected value of the next state and overestimates the reward.

B.2.2 Ablation: State-Value Approximation and KL Term

To separate the contributions of two components of QVIRL, we ran a 2×2 ablation in Lunar Lander apprenticeship learning. The first factor is the approximation of the

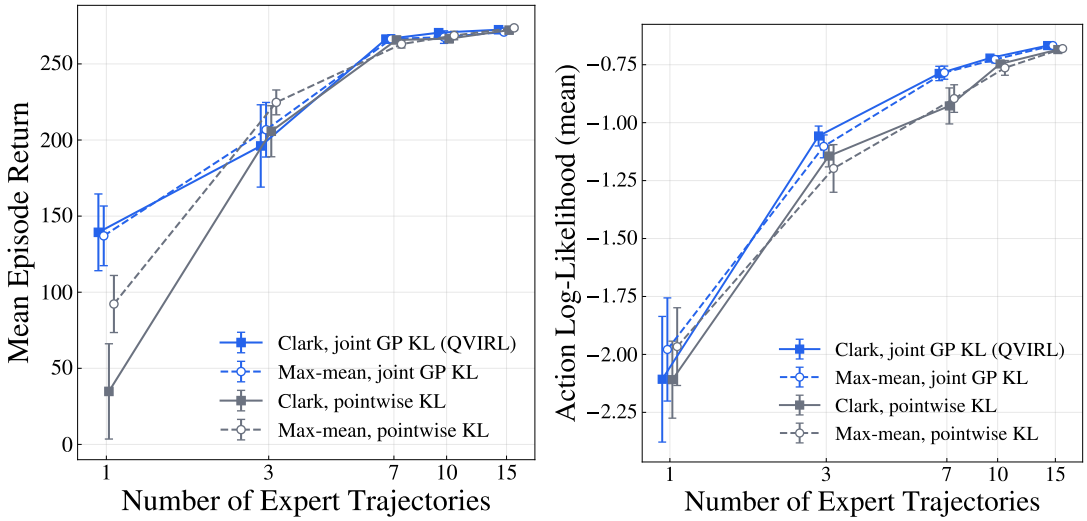


Figure B.2: Ablation of QVIRL in Lunar Lander apprenticeship learning: mean episode return (left) and mean log-likelihood of held-out expert actions (right) with different numbers of expert trajectories, for the Clark or max-mean approximation of the successor state-value (solid or dashed) combined with the joint Gaussian-process KL or a pointwise KL (blue or grey). Clark with the joint KL is the configuration used in the main experiments. Averaged over 10 runs, with standard errors of the mean; points are spaced horizontally to improve readability.

successor state-value $V^*(s') = \max_{a'} Q^*(s', a')$ in the implied reward (Equation 4.3): Clark's approximation (Appendix B.1.2), as in all our other experiments, or the max-mean approximation described above. The second factor is the KL term of the ELBO (Equation 4.7). The *joint* KL, used in all our other experiments, compares the joint Gaussian posterior over the rewards of all transitions in a mini-batch with the Gaussian-process prior evaluated at the same transitions (Appendix B.4.7). The *pointwise* KL ignores correlations between transitions: it averages, over the transitions in the mini-batch, the KL divergence between the marginal reward posterior of each transition and an independent Gaussian prior $\mathcal{N}(-1, 50^2)$, which has the mean and marginal variance of the Gaussian-process prior at initialization. Note that the pointwise prior is fixed, whereas the lengthscales and marginal variance of the Gaussian-process prior are learnt with the rest of the model, so the comparison of the KL terms also includes this difference. All other settings, including the use of the apprentice's own rollouts as auxiliary transitions in the KL term, are as in the main experiments. We ran each of the four variants with the

same 10 seeds and, for each seed, the same expert demonstrations, so we compare the variants by paired differences across seeds.

Figure B.2 shows that the KL term matters more than the state-value approximation. The joint KL improves the held-out log-likelihood of expert actions most clearly with 7 trajectories: by 0.14 ± 0.05 with Clark's approximation (paired t-test, $p = 0.02$) and by 0.11 ± 0.03 with max-mean ($p = 0.008$). With 3 trajectories, the improvement is 0.09 ± 0.03 with Clark's approximation ($p = 0.01$) but is not significant with max-mean. The gap narrows to 0.01–0.04 with 10 and 15 trajectories. With a single trajectory, the joint KL also raises the mean return with Clark's approximation ($+105 \pm 40$, $p = 0.03$; with max-mean, $+45 \pm 21$, $p = 0.06$), although returns with 1 and 3 trajectories vary widely across seeds. From 7 trajectories on, all four variants reach expert-level returns of about 265 to 275. By contrast, Clark's and the max-mean approximations perform indistinguishably on both metrics, for both KL terms and at every number of trajectories (all paired $p > 0.1$). This is consistent with the illustration above: the two approximations differ only where the posterior is uncertain about the best action in the successor state. It also supports max-mean as a simpler starting point for reimplementation (Appendix B.1.2). In Highway, where all methods reach expert-level performance from very few demonstrations (Figure 4.3), the four variants are indistinguishable: at every number of trajectories, their mean held-out log-likelihoods lie within 0.02 and their mean returns within 0.15 of each other.

B.2.3 Likelihood Approximation

Following Lu et al. [157], in Equation 4.6, we approximate the likelihood

$$p(a|s;\theta) = \int_{Q(s,\cdot)\in\mathbb{R}^{|\mathcal{A}|}} \frac{e^{\beta Q(s,a)}}{\sum_{a'} e^{\beta Q(s,a')}} q_\theta(Q(s,\cdot))\, dQ(s,\cdot)$$

as

$$p(a|s;\theta) \approx \left(\sum_{a'} \exp\left(-\frac{\beta\big(\mu_Q(s,a)-\mu_Q(s,a')\big)}{\sqrt{1+3\pi^{-2}\beta^2(\sigma_Q(s,a)^2+\sigma_Q(s,a')^2-2\Sigma_{aa'})}}\right)\right)^{-1}.$$

To examine the quality of the approximation, we compare the probability of the action under the approximation against an approximation of the integral using Monte

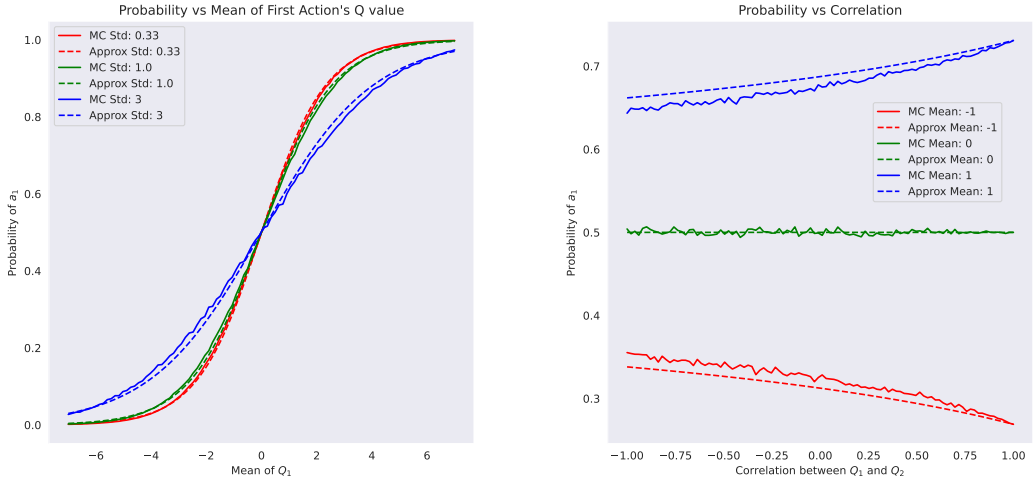


Figure B.3: Comparison of the likelihood approximation (Equation 4.6) against a Monte Carlo evaluation using 10,000 Monte Carlo samples. Left: probability of a_1 as a function of the mean of Q_1 for three levels of standard deviation of Q_1. Correlation is set to 0 (the plot looks very similar for correlated values). Right: probability of a_1 as a function of the correlation between Q_1 and Q_2 for three levels of the mean. Standard deviation is fixed to 1. Note the different scales on y-axes of the two plots.

Carlo integration with 10,000 samples. In the test scenario, there are two actions, a_1 and a_2, with respective Q-values Q_1 and Q_2 that are jointly normally distributed. We fix the marginal distribution of Q_2 to standard normal. We then try varying the mean and standard deviation of Q_1 as well as the correlation between Q_1 and Q_2.

Figure B.3 shows the results. The approximation seems reasonably good, though it gets worse as correlation decreases.

Note that even with 10,000 samples, the Monte Carlo estimate displays notable noise. While we could use sampling and the reparameterization trick during training, we observed that this destabilizes training and we did not manage to get competitive results on any of the benchmarks. Thus we recommend using the proposed approximation that can be evaluated analytically.

B.2.4 Gaussianity

The Gaussianity assumption is illustrated in Figure B.4, which shows that for a Gaussian prior, the posterior stays reasonably close to Gaussian. The use of variational inference requires an assumption on the distribution family and the

Gaussian is a natural starting point, on which further work can build to expand to other distribution families, especially if practical cases arise where the prior and associated posterior are notably non-Gaussian.

B.3 Additional Discussion on AVRIL

Here, we provide additional details about our modifications to AVRIL that we implemented for fair comparison between the methods. Additionally, we give a visualization of AVRIL's shortcomings in a 3×3 gridworld setting, and conclude with a short summary of limitations of AVRIL as a Bayesian IRL method.

B.3.1 Dynamics-Aware AVRIL

AVRIL was originally proposed as an offline algorithm not using the environment dynamics or auxiliary transitions. However, in online settings, this would lead to AVRIL never updating the posterior over the reward associated with states visited during environment rollouts, as AVRIL only updates reward posteriors of states that appear in the demonstrations. Here we assume knowledge of the dynamics, so for fair comparison we introduce a dynamics-aware version of AVRIL.

The main change is that instead of evaluating both the TD term and the KL divergence between the reward variational posterior and the prior only on the expert trajectories, we evaluate them across all state-action pairs, i.e. for each state-action pair (s, a), we calculate

$$R(s,a) = Q^*_\theta(s,a) - \gamma \mathbb{E}_{s'|s,a} V^*_\theta(s'),$$

where $V^*_\theta(s') = \max_{a'} Q^*_\theta(s', a')$. Then the TD soft constraint is calculated as

$$\sum_{(s,a)\in\mathcal{S}\times\mathcal{A}} \log q_\phi(R(s,a)). \tag{B.17}$$

Similarly, we evaluate the KL divergence between q_ϕ and the prior $p(R)$ across all states and actions rather than just on the expert trajectories.

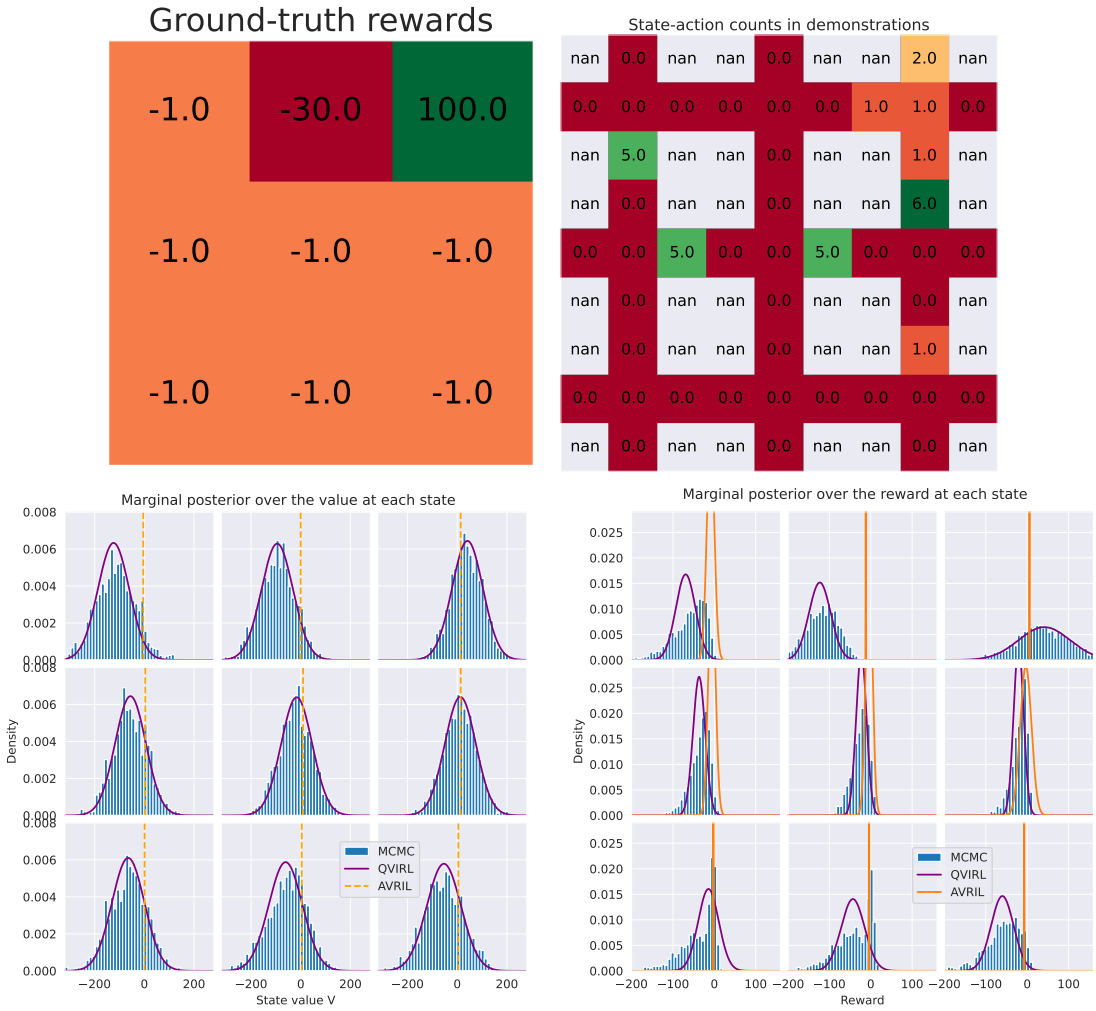


Figure B.4: Top left: Ground-truth rewards for a 3×3 gridworld. The top left state is the initial state. The top right state is terminal. Top right: State-action counts in the demonstration data. Bottom left: PDF of the marginal posterior over state values recovered by QVIRL, a histogram of 2000 samples from the true posterior produced by ValueWalk, and a vertical line indicating the point estimate recovered by AVRIL. Bottom right: PDFs of the variational posterior over reward recovered by QVIRL and AVRIL and the corresponding histogram.

B.3.2 3×3 Gridworld and a Comparison to AVRIL

To visualize the shortcomings of AVRIL when it comes to recovering true posteriors of the state values, we conducted an experiment in a simple 3×3 gridworld shown in Figure B.4. We treated all state-only rewards as unknown. We assume an independent normal prior $\mathcal{N}(0, 66)$ for the reward in every state. There is also noise in the environment dynamics, so for any action, there is a 0.1 probability of slipping and moving in a random direction or staying in place (with uniform probability

between the 5 options). The top right state yields a reward of 100, while the middle top tile represents a hazardous obstacle with a reward of -30.

We provide the algorithms with 5 trajectories, each starting in the top left corner and heading to the goal via the middle tile while avoiding the obstacle. The expert once slipped to the bottom right tile, placing one demonstration step there.

We ran QVIRL and a dynamics-aware version of AVRIL (described in Appendix B.3.1 above) using these demonstrations, as well as ValueWalk [7], which produces samples from the true posterior.

Figure B.4 shows that QVIRL's variational posterior approximates the true posterior well (as is also the case in Section 4.5), while the AVRIL point estimate remains very close to zero (which, however, still produces action predictions consistent with the data). The fit of the derived posterior over the rewards is worse – some of the posteriors are visibly further from Gaussian (many exhibiting negative skew) than their value counterparts. Still, the derived Gaussian posterior over rewards recovered by QVIRL roughly matches the means and variances of the MCMC samples. On the other hand, the posterior over rewards recovered by AVRIL remains centred near zero in almost all cases. Also, in 5 of the 9 states, the standard deviation of the posterior collapses to below 0.2, where it should be between 40 and 64 to match the MCMC-based posterior.

Furthermore, as discussed above, we found AVRIL to be extremely sensitive to the value of its hyperparameter λ that regulates the strength of the constraint on consistency between the Q-values and the reward posterior. In the above experiments we used a value of $\lambda = 0.2$; however, increasing it to 0.5 makes the variance of all states collapse to near zero (std$<$0.001 for 8 states out of 9), while decreasing to 0.1 essentially reverts the posterior to the prior. The AVRIL paper gives no indication of how to pick a good value for λ.

B.3.3 Limitations of AVRIL

Since QVIRL may seem to resemble AVRIL, and AVRIL can be considered as the closest baseline, we would like to re-emphasize some important limitations of

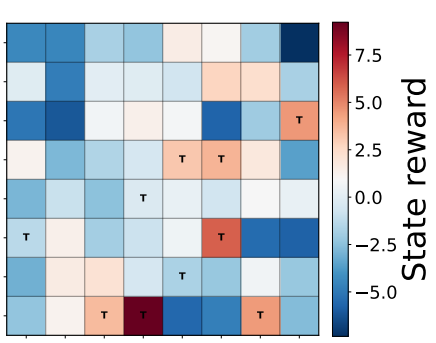
AVRIL that our method does not suffer from:

- AVRIL does not provide uncertainty estimates over the optimal Q-values and thus over the optimal policy. This does not permit one to easily extract risk-averse policies. On the other hand, doing so is natural in QVIRL, which gives us a downstream use gain for comparable computational effort.

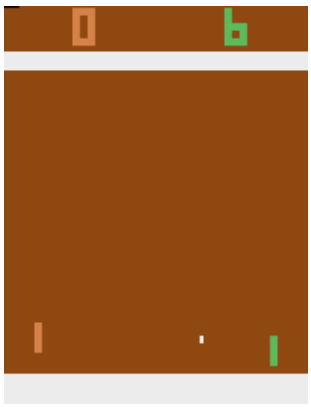
- AVRIL's posterior over rewards does not seem to track well the true posterior over rewards even on a simple gridworld. QVIRL's fit is faithful to the posterior recovered by MCMC methods.

- The posterior variance over rewards is extremely sensitive to AVRIL's hyperparameter λ that regulates the strength of the consistency constraint between the reward variational distribution and the Q-values and a narrow range of values can make the posterior either collapse to a Dirac-delta-like function, or revert to the prior. The paper gives no indication of how to pick the value. QVIRL has no such hyperparameter, and is robust even to other potentially challenging design choices such as the weighting of KL divergence and log-likelihood in the ELBO.

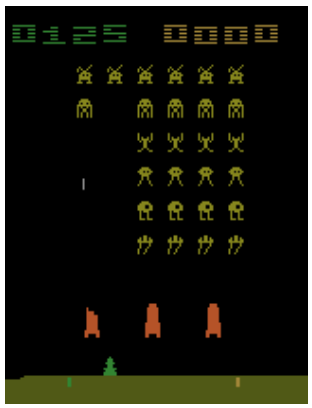
- The AVRIL paper presents a 'strictly batch setting' where only the expert demonstrations are used to estimate environment transitions, and thus evaluate the closeness of the posterior to the prior and the consistency between the Q-values and the reward distribution. We think this largely gives up a crucial advantage of IRL over behavioural cloning – leveraging the environment dynamics to make inference about rewards away from the expert trajectories, enabling better generalization. Our paper presents variants of the algorithm also for the case of known environment dynamics, or an online setting where we have access to a simulator or the environment itself.



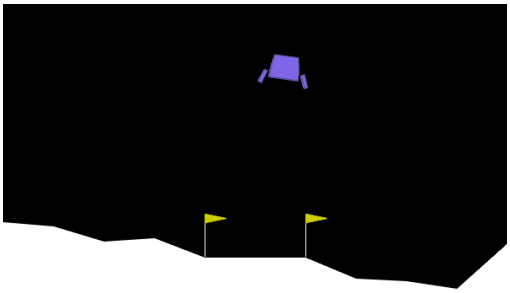
(a) Gridworld environment.



(b) ATARI Pong environment.



(c) ATARI Space Invaders.



(d) Lunar Lander environment.



(e) Highway Env environment.

Figure B.5: The environments used in our experiments.

B.4 Experiment Details

B.4.1 Environment Descriptions

Gridworld The gridworld (Figure B.5a) has 5 actions, corresponding to staying in place and moving in the four directions. Furthermore, there is a probability of 0.1 of a random action being executed instead of the intended one. If an action would result in crossing the edge, the agent instead remains in place. The gridworlds use a state-only reward (awarded upon executing any action in the given state). The 100 randomized 8×8 gridworlds were generated as follows:

1. Each state was assigned a random reward drawn independently from $\mathcal{N}(-1, 3)$ (which can be interpreted as $\mathcal{N}(0, 3)$ minus a time penalty of one per step).

2. Each state was then marked as terminal with an independent probability of

0.1.

3. The top 10% of states with the highest reward were further marked as terminal (producing terminal goal states, which may, however, sometimes be avoided by the optimal policy in favour of staying forever in other positive states).

4. The initial state distribution is uniform across the whole state space.

Lunar Lander Lunar Lander, part of the classic control environments available in Farama Foundation's Gymnasium library [182], is a standard benchmark task in reinforcement learning. It models a spacecraft attempting to land on a designated pad located between two flags on a flat surface. The state is an 8-dimensional continuous vector that includes the lander's horizontal and vertical position and velocity, its angle and angular velocity, and indicators of whether the left and right legs are in contact with the ground. We use the discrete variant of the environment, containing 4 actions – a no-op and control actions of the left and right main and orientation engines. The agent receives positive reward for successfully approaching the landing pad and slowing down. It receives negative reward for excessive tilt and firing of the engines. If the agent successfully lands, it receives an additional reward of $+100$, and if it crashes, it receives a reward of -100. An episode is considered solved at an episode return of $+200$, with a return of approx. 270 commonly being considered as expert performance.

For active learning, we implement a version of the environment in which the starting state is resettable from a state vector in order to allow active querying of the expert and targeted rollouts.

Highway Env Highway Env is a simulated autonomous driving task, also available in Gymnasium, in which an agent has to negotiate a multi-lane highway that is shared with other vehicles. We use the fast version of the environment with kinematic observations and a discrete action space. The observed state is a 5×5 feature vector that contains the 2D position of the ego vehicle (the agent) and its velocities, as well as the same features for the 4 vehicles closest to the ego vehicle in a given

state. The actions available to the ego vehicle are a set of discrete meta-actions: change lane (left or right), speed up, slow down, and no-op. The episodes are time-limited to 30 seconds, and the agent receives a reward that is proportional to the difference between its speed and the environment's desired speed. The agent also receives positive reward for keeping in the rightmost lane. The agent receives negative reward if it crashes into any other vehicle, or drives off the road. The maximum reward the ego vehicle can obtain is +30.

For active learning, we implement a version of the environment in which the starting state is resettable from a true state vector, and we collect demonstrations in the true state space. When training experts and IRL algorithms, we convert the true state space into observations as given by the original environment.

ATARI To test QVIRL's scalability to raw, high-dimensional pixel observations, we evaluate apprenticeship learning on two games from the Arcade Learning Environment [183], Pong and Space Invaders, following the evaluation protocol used by IQ-Learn [154]. Observations are pre-processed using the standard ATARI pipeline: each episode starts with a random number (1–30) of no-op actions (followed by the FIRE action in games that require it), frames are converted to greyscale and resized to 84×84, each action is repeated for four frames (with max-pooling over the last two of them), and the last four frames are stacked to form the $4 \times 84 \times 84$ observation given to the agent, so that short-term dynamics (e.g. ball velocity in Pong) are observable from the state alone. We use no sticky actions and no end-of-life episode boundaries. The action space is discrete and game-specific (the minimal action set). Episode returns are reported on the raw (unclipped) game score. None of the methods uses the environment reward for training. The prior distribution used for the reward is described in Appendix B.4.7.

B.4.2 Expert Demonstrations

To collect expert demonstrations in gridworlds, we solve for the optimal Q-values using value iteration, and then roll out 5 expert trajectories with a Boltzmann

rationality coefficient $\beta = 2$ and maximum length 5. In the active learning setting, we start with a single such trajectory and then collect an additional one from the expert in the state queried in each active step.

To collect expert demonstrations in Lunar Lander and Highway Env, we train an expert using QR-DQN [184]. When the expert's Q-value function has converged, we use it to sample trajectories with different rationality coefficients: $\beta = 3$ on Lunar Lander and $\beta = 5$ in the highway environment. For each β, we collect 1000 trajectories, and randomly split them into 800 training trajectories, 100 evaluation trajectories, and 100 test trajectories. We further split the 800 training trajectories into 10 subsets, and we use one training subset during a single method evaluation, or use all 10 splits across 10 runs of the same method. Furthermore, we sometimes further restrict the number of trajectories we give to the methods, and in this case we subsample the training splits to create smaller training datasets.

To collect expert demonstrations on ATARI, we roll out a Boltzmann-rational policy ($\beta = 16$) from a pretrained action-value network (a Munchausen [185] Implicit Quantile Network [186]) for each game, matching IQ-Learn's demonstration budget of 20 training trajectories per game. In Space Invaders, we keep only episodes with a game score of at least 800. We form 5 disjoint training sets of 20 trajectories; run k of each method uses training set k. A further 5 trajectories are held out per game to evaluate action log-likelihood, as in the other environments.

B.4.3 QVIRL Training

In our experiments, we train QVIRL by stochastic gradient ascent using automatic differentiation in PyTorch with the Adam optimizer and a learning rate of 0.001 (using the default values for other parameters). We use the same Boltzmann coefficient as was used to generate the demonstrations, and a discount rate $\gamma = 0.99$ for Lunar Lander, 0.95 for Highway, and 0.9 for gridworlds. Each run lasted about 20000 iterations with a batch size of 64. Where a neural network encoder is used, we use a multi-layer perceptron with two hidden layers of 128 units and an ELU activation function. An embedding size of 4 is used in our experiments. We also

used weight decay of 0.001. We tuned only the learning rate and weight decay, which were chosen by trial and error before the whole set of experiments whose results are reported was run. At most 5 tries were made for each hyperparameter.

On ATARI, the state input is first passed through a convolutional (NatureCNN) encoder [183] – three convolutional layers followed by a linear layer with 512 output features – before entering an MLP with hidden layers of 256 and 128 units and embedding size 8; the encoder is trained end-to-end together with the rest of QVIRL's variational parameters, rather than pretrained or frozen. We use Clark's approximation of the maximum with the joint Gaussian-process KL (weight 1), $\beta = 16$, and $\gamma = 0.99$. We train for 500k steps on Space Invaders and 200k steps on Pong with batch size 32, using Adam with a one-cycle learning-rate schedule (peak learning rate $2 \cdot 10^{-4}$ reached after 30% of training, cosine annealing), gradient-norm clipping at 5, and weight decay 10^{-5}. The next-state Q-values in the reward belief come from a target network updated by Polyak averaging with rate 0.005, and we evaluate an exponential moving average of the weights (decay 0.9999). Every 2000 steps, QVIRL collects one rollout of a Boltzmann policy on its mean Q-values, with a rationality coefficient drawn log-uniformly from $[\beta/5, 5\beta]$. The most recent 50 (Space Invaders) or 10 (Pong) rollouts provide auxiliary transitions that enter only the KL term (Appendix B.1.3).

B.4.4 Baselines on ATARI

All baselines use the same step budgets, learning-rate schedule (peak $2 \cdot 10^{-4}$), $\gamma = 0.99$, and demonstrations as QVIRL. IQ-Learn uses a NatureCNN encoder with 512 features and a linear output layer, the χ^2 regularizer, temperature 10^{-3}, no target network, and batch size 64. AVRIL uses separate NatureCNN encoders (256 features) with MLP heads of 256 and 128 units for its reward and Q-networks, an $\mathcal{N}(0, 1)$ reward prior, weight decay $5 \cdot 10^{-4}$, and batch size 32. BC uses a NatureCNN encoder (256 features) with an MLP head of 256 and 128 units and batch size 64. All baselines are trained offline, on the demonstrations only. The reference implementation of IQ-Learn does not use its online rollouts in the training loss on

ATARI, and in our experiments IQ-Learn failed to learn reasonable policies with online rollouts enabled. The held-out action log-likelihood of BC uses its softmax policy, and those of IQ-Learn and AVRIL use the Boltzmann policy softmax(βQ) with $\beta = 16$; QVIRL's uses the probit approximation of its posterior predictive.

B.4.5 Evaluation on ATARI

Every 10k training steps, we evaluate the greedy policy (the argmax of the mean Q-values for QVIRL, of the Q-values for IQ-Learn and AVRIL, and of the policy logits for BC) on 10 episodes. We keep the weights with the highest such evaluation return (the earliest in case of a tie). The held-out log-likelihood of expert actions turned out to be a poor proxy for apprentice return, so the return is the only criterion we have for this selection. To avoid the upward bias of reporting the evaluation that the selection was made on, we evaluate the selected weights once more at the end of training, on 50 fresh episodes (with environment seeds not used during training) of at most 30k decisions each, and report this evaluation for all methods. The held-out action log-likelihood is the mean log-probability per action of the 5 held-out trajectories, under the same selected weights.

B.4.6 Active Learning

In Lunar Lander and Highway, query candidates are drawn from the expert trajectories in the training split used by the run. In Lunar Lander, the candidate pool consists of 10-step segments of these trajectories, excluding the initial trajectory. The acquisition function sees only the first state of each segment, and the expert's continuation is revealed only once the segment has been chosen. In Highway, candidates are single states, and a query reveals the expert's action in that state. In each active learning step, 100 candidates are subsampled from the pool.

To compute Reward EIG, we simulate the expert's response from each candidate state under posterior samples of the optimal Q-values, and estimate the information gain with a contrastive estimator using 2048 posterior samples. For QVIRL, the samples are drawn from its variational posterior; for the ensemble, each member's

Q-function is treated as one equally weighted sample. In Lunar Lander, we simulate 64 expert trajectories of length 10 per candidate in a copy of the environment reset to the candidate state. In Highway, where a query concerns a single action, the expectation over the expert's action is computed in closed form and averaged over 1024 posterior samples. The number of simulated trajectories matters. With 16 trajectories per candidate in Lunar Lander, the estimated scores were noisy enough that Reward EIG with QVIRL performed no better than random selection ($+3.4 \pm 16.9$ in mean episode return averaged over the active learning steps, on 10 seeds). On the same seeds, 64 trajectories gave $+27.4 \pm 10.7$.

After each query, both backbones are retrained from scratch on all data collected so far. During training, both also receive transitions collected online by rolling out their current policy. These transitions carry no expert actions and are used only for regularization; in QVIRL, they enter only the KL term. After each active learning step, we report the mean return of the greedy apprentice policy over 30 evaluation episodes. In Highway, both backbones use a discount rate of $\gamma = 0.9$ in this experiment.

B.4.7 Prior Distribution

In the gridworld experiments, we used an independent Gaussian prior $\mathcal{N}(-1, 3)$ for each state.

In Lunar Lander and Highway, we used a Gaussian-process prior with constant mean -1 and an automatic relevance determination (ARD) RBF kernel,

$$k(x, x') = \sigma_0^2 \exp\left(-\tfrac{1}{2} \sum_d \left(\frac{z_d - z'_d}{\ell_d}\right)^2\right), \tag{B.18}$$

evaluated on standardized features $z = (x - \bar{x})/s_x$, where x concatenates the (flattened) state vector with a one-hot encoding of the action, and $\bar{x}, s_x$ are the empirical per-feature mean and standard deviation computed once from the expert demonstrations. We set the kernel's marginal standard deviation σ_0 to 50 on Lunar Lander and 1 on Highway, to roughly track the magnitude of rewards in each environment. The per-feature lengthscales ℓ_d are initialized in a data-adaptive way,

as the root-mean-square of the lag-one differences of the standardized feature z_d along expert trajectories (differences are never taken across trajectory boundaries). Both the lengthscales and σ_0 are then treated as learnable parameters, and are optimized jointly with the rest of QVIRL's variational parameters, by the same Adam optimizer, driven end-to-end by the ELBO – i.e. we do not perform a separate marginal-likelihood (type-II ML) fit of the kernel hyperparameters.

On ATARI, we used a zero-mean Gaussian-process prior whose kernel is the product of an action kernel and an RBF kernel on Gaussian-blurred pixel intensities of the (greyscale, 84×84) frame-stack observation, plus a small nugget,

$$k\Big((s,a),(s',a')\Big) = \sigma_0^2\Big[(1-\epsilon)\,k_A(a,a')\,\exp\Big(-\tfrac{\|G*s-G*s'\|^2}{2\ell^2}\Big) + \epsilon\,\mathbf{1}\Big[(s,a)=(s',a')\Big]\Big], \tag{B.19}$$

where $G * s$ denotes each frame of the stack convolved with a Gaussian of standard deviation 2 pixels (with replicated edges), and $\epsilon = 0.05$. Blurring keeps a frame similar to itself with small objects (such as the ball and paddles in Pong) displaced by a few pixels, so that the kernel varies smoothly with the positions of such objects. The nugget adds reward variation that is independent across distinct state-action pairs. We use the same kernel on both games. In preliminary experiments on Pong, a mixture of patch-, motion-, and layout-based image kernels made nearly all pairs of states almost perfectly correlated under the prior, which kept QVIRL's Q-values nearly constant across actions for most of training. The action kernel k_A (with $k_A(a,a) = 1$) is an RBF kernel over hand-specified, per-action semantic features (one-hot encoding each move direction and whether the agent fires), mixed with a small Kronecker-delta term. The lengthscale ℓ is set using a median heuristic (the median distance between 4096 random pairs of blurred frame stacks from 512 randomly chosen demonstration states). Pixel intensities are scaled to $[0,1]$, and the top 8 rows of each frame, which display the score, are masked out. The action kernel puts weight 0.95 on the RBF term and 0.05 on the Kronecker delta, and we add a jitter of $10^{-4}\sigma_0^2$ to the diagonal. We fix the kernel's marginal standard deviation σ_0 to 0.5 to roughly track the magnitude of the sign-clipped rewards commonly used on ATARI. Unlike in Lunar Lander and Highway, all ATARI kernel hyperparameters –

the lengthscale, blur, and nugget – are kept fixed at their (data-informed) initial values rather than trained jointly with the rest of the model.

B.5 Additional Experiments

Here, we provide some additional experiments which are meant to expose and visualize the properties of QVIRL, and the shortcomings of AVRIL, discussed in Appendix B.3.3.

B.5.1 Sensitivity of QVIRL to Rationality Coefficient β

While QVIRL has relatively few hyperparameters that require tuning, most current Bayesian IRL methods require the designers to specify the rationality coefficient β, which controls the sharpness of the softmax operation in modelling the demonstrator's action selection (Equation 4.1). Naturally, the (mis-)specification of β influences the reward learning, as the learning signal might be biased or lost due to misaligned assumptions.

To study the influence of misspecification of the rationality coefficient β on QVIRL's learning process, we identified 4 values of β that produced distinct behaviours in gridworlds, ranging from random actions to optimal experts. We then trained QVIRL by using all values of β for each expert. We repeated the experiment across 30 seeds, and we considered 1, 7, and 15 demonstrations. As our metric of interest, we focus on the discrepancy of the held-out greedy expert action likelihood between the demonstrator and the learner, that is, we measure the difference in probability mass assigned to the true optimal action in a state. We report the mean of this metric across all seeds, and a positive difference means the learner places less mass on the optimal action, and vice versa for a negative difference.

As observed in Figure B.6, the interaction between the β parameter and learnt action probabilities depends not only on the misspecification but also on the amount of demonstration data given to QVIRL. In low-data regimes, QVIRL performs better when assuming a less deterministic expert than the true value, and interestingly, best performance is achieved by deliberately underestimating the expert's

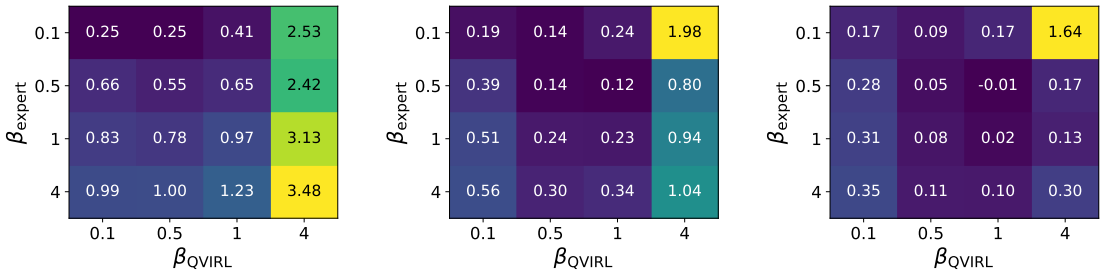


Figure B.6: The mean true greedy action log-probability difference between the demonstrator and learner for 1, 7, and 15 demonstrations, respectively.

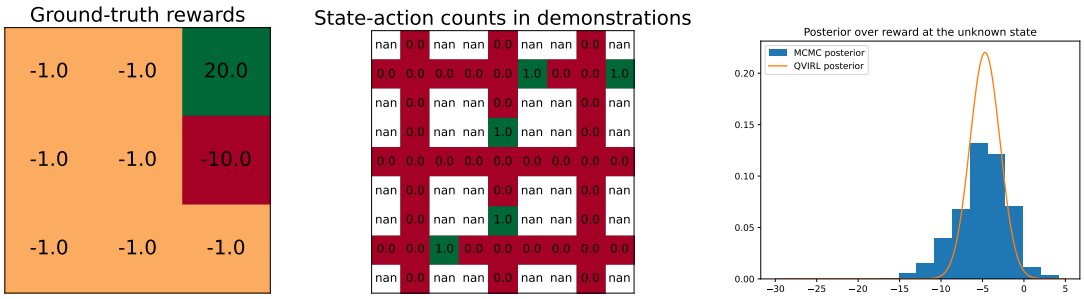


Figure B.7: The ground-truth rewards, demonstrations, and the posterior over the unknown right-middle reward recovered by QVIRL and ValueWalk in the illustrative gridworld.

rationality. However, as the amount of data increases, its regularization effect is more pronounced, and the variance is better explained by the rationality coefficient. In terms of absolute values, it appears to be better to slightly underestimate the expert's rationality rather than overestimate it.

An exciting direction for future work is to include the rationality parameter β as an explicit part of the inference, alongside the optimal Q-values, to be estimated from demonstration data.

B.5.2 Illustrative Gridworld Experiment with a Single Unknown State

For a human-readable and easily interpretable illustration of what our method is doing, we ran our method on the simple gridworld in Figure B.7 (left) where the top right state is terminal. We assume the learner knows all the rewards (which we model as a tight normal prior with std 0.1 centred at the true value) except for the right

middle tile, where it has a normal prior with mean -1 and std of 10. We assume known deterministic dynamics, expert rationality coefficient $\beta = 1$, and $\gamma = 0.8$.

We let the learner observe a single expert demonstration shown in the middle subplot. We are particularly interested in how an apprentice agent would behave in the bottom right cell – would it be worth cutting through the unknown-reward tile (to get to the goal faster), or should it go around? The expert demonstration would be consistent with either being optimal.

The right subplot shows the reward posterior recovered by QVIRL (as well as an MCMC reference posterior recovered using ValueWalk, which produces samples from the true posterior using MCMC). We find that if we use value iteration to find the apprentice policy maximizing the return with respect to the mean reward according to this posterior, the apprentice chooses to go through the unknown-reward tile (with a Q-value of 8.0 for going up vs a Q-value of 5.2 for going to the left). On the other hand, if we solve for maximizing the mean minus 2 std of the posterior, representing a risk-averse policy, the Q-value of going up drops to 2.7 and the apprentice would choose to go around.

This illustrates how modelling posterior uncertainty allows us to behave in a risk-averse manner. Furthermore, the plot shows that the reward posterior deduced from the Q-value posterior fits well with the reference reward posterior.

B.6 Code, Data, and Compute

B.6.1 Code and Data

All experiments were run using publicly available libraries and data. The main software components we used were Python 3.13 (available under the GPL-compatible PSF license agreement; `https://docs.python.org/3/license.html`), Farama Gymnasium 1.4.0 and Highway Env 1.10.2 (MIT License), PyTorch 2.14.1 (BSD-3 License), and ale-py 0.12.0 (GPL-2.0 License).

Our code will be made available on GitHub at `https://github.com/bayesian-reward-learning/qvirl`.

B.6.2 Computing Resources Used

Except for ATARI, each training run was executed on a single CPU core (of either an 8-core AMD Ryzen 7 PRO 7840U or Amazon Graviton4) and took between 2 and 15 minutes. Each ATARI training run used a single NVIDIA A100 GPU and took about 1.2 hours of wall-clock time for QVIRL for the 200k steps on Pong, and around 3.2 hours for the 500k steps on Space Invaders, including periodic evaluation. The baselines were faster: BC took about 0.4 and 0.8 hours, IQ-Learn about 0.5 and 1.0 hours, and AVRIL about 0.7 and 1.5 hours on Pong and Space Invaders, respectively.

C

Appendix to Chapter 5 (PAC Active IRL)

Contents

C.1 Failure Modes of Prior Methods 131
C.2 Theoretical Analysis 135
C.2.1 Notes on Possible Improvements 145
C.3 Accounting for Visitation Frequencies 145
C.4 Alternative Acquisition Functions 147
C.4.1 From Apprentice Return to Regret 147
C.4.2 The Challenge of Direct Regret Optimization 147
C.4.3 Information Gain About Regret 148
C.4.4 Immediate Regret EIG 149
C.4.5 Discretization and Connection to PAC-EIG 149
C.4.6 Summary 150
C.5 Experiment Details 150
C.5.1 Basic Parameter Values 150
C.5.2 Environments 151
C.5.3 Bayesian IRL Methods 151
C.5.4 Metrics 152
C.5.5 Implementation 152
C.5.6 Timing 153
C.6 Notation Overview 153

C.1 Failure Modes of Prior Methods

For each of the three prior methods for active IRL, we will now present an example of a simple environment where the method makes a clearly suboptimal choice with

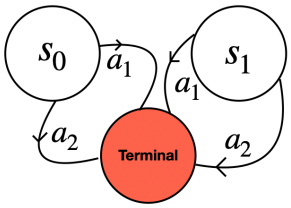


Figure C.1: Two-state environment designed to illustrate a failure mode of Lopes et al. [160]

respect to at least one of the two objectives.

Policy entropy [160] As a reminder, the policy entropy acquisition function is $\alpha^{\text{Lopes}} = H(\pi^E)$, i.e. the entropy of the expert policy with respect to the current posterior over rewards (which induces a posterior over expert action probabilities). Consider an environment with two states $s_{0,1}$ each with two actions $a_{1,2}$ as shown in Figure C.1. We aim to illustrate a scenario where α^{Lopes} can misallocate budget, from the point of view of the apprenticeship learning objective, by focusing on states where the optimal action is already known, rather than those where crucial information about optimality is missing.

To demonstrate this effect, we define a *discrete prior distribution over rewards*. This uncertainty in rewards will, in turn, induce a prior over the possible action probabilities for an optimal policy. Suppose in state s_1 we have strong prior knowledge that a_1 is the optimal action; however, we are uncertain about the exact reward obtained by taking a_1

$$P(R_{s_1,a_1} = 5) = 0.5\,, \quad P(R_{s_1,a_1} = 7) = 0.5\,, \tag{C.1}$$

and

$$P(R_{s_1,a_2} = 1) = 1.0\,. \tag{C.2}$$

An optimal apprentice policy will always choose a_1 in state s_1. Despite this certainty, the uncertainty in the exact reward for a_1 means there is still uncertainty regarding the precise probability an optimal policy would assign to a_1, which leads to a high measure of policy uncertainty (as measured by α^{Lopes}).

For state s_0, we set priors

$$P(R_{s_0,a_1} = 2) = 0.1\,, \quad P(R_{s_0,a_1} = 3) = 0.9\,, \tag{C.3}$$

and

$$P(R_{s_0,a_2} = 2) = 0.1\,, \quad P(R_{s_0,a_2} = 3) = 0.9\,. \tag{C.4}$$

such that the optimal action is *uncertain*. In this state, the learner faces true ambiguity about the best action, and it is therefore the state that a good active IRL method focused on improving the apprentice policy should query. However, since the acquisition function of Lopes et al. [160] is formulated using entropy of possible actions probabilities, examples of this type could have $\alpha^{\text{Lopes}}(s_0) < \alpha^{\text{Lopes}}(s_1)$, resulting in an inefficient use of budget. For example, given inverse temperature $\beta = 2$, we obtain values

$$\alpha^{\text{Lopes}}(s_0) = 0.860\,, \quad \alpha^{\text{Lopes}}(s_1) = 1.0\,, \tag{C.5}$$

so this acquisition function would query s_1, where the policy *already knows which action is optimal*, rather than s_0 where there is key information to be gained. By contrast, assuming single-state queries, our PAC acquisition function would choose s_0, since in state the regret is already known to be 0 for the current apprentice policy so there is no regret information to be gained there.

ActiveVaR [161] This acquisition function for getting single-state annotations is equal to a particular quantile of the posterior distribution over regret of the apprentice policy starting from that state. In this example, we use a 0.9 quantile, though the example is robust with respect to the exact value. Consider an environment with two states labelled $s_{0,1}$ and two actions $a_{1,2}$ as shown in Figure C.2. Both actions in state s_0 lead to s_1, one with reward $+2$ and the second with -2 (but we do not know which is which). In s_1, both actions lead to a terminal state, and give a reward of -10 and $+10$. Since the potential downside of any policy is maximal at s_0 (-12), the acquisition function would query s_0. On the other hand, querying state s_1 to distinguish the ± 10 rewards would yield a greater reduction in expected regret.

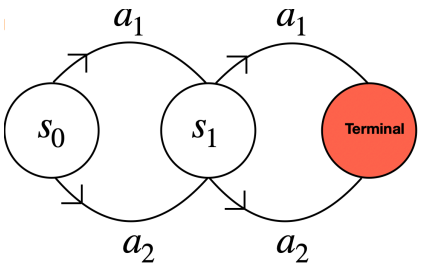


Figure C.2: Two state environment to demonstrate a failure mode of Brown et al. [161].

To get even more concrete, consider an intermediate policy which *knows* the absolute values of all the rewards, but not the relative signs: (i.e. $(+2, -2)$ and $(-2, +2)$ are equally likely for $(r(s_0, a_1), r(s_0, a_2))$, as are $(+10, -10)$ and $(-10, +10)$ for $(r(s_1, a_1), r(s_1, a_2))$. We can easily compute

$$\alpha^{\text{Brown}}(s_0) = 2 + \gamma 10\,, \quad \alpha^{\text{Brown}}(s_1) = 10\,, \tag{C.6}$$

so for a sufficiently large discount factor, state s_0 would be queried by this acquisition function. We can compute the reduction in expected regret after querying each of these states. The initial expected total regret for any apprentice policy, averaged over a uniform initial state distribution is

$$\mathbb{E}_r[R_{\pi^{\text{A}},r}] = \frac{1}{2}\Big(V^*(s_0) - V^\pi(s_0)\Big) + \frac{1}{2}\Big(V^*(s_1) - V^\pi(s_1)\Big) = \\ \frac{1}{2}(2 + \gamma 10 - 0) + \frac{1}{2}(10 - 0) = 6 + 10\gamma\,.$$

If we query the expert at s_0 and as a result switch the apprentice action at s_0 to the optimal one (which means we get a reward of 2 in s_0 while still getting an expectation of 0 in s_1, we get an expected regret of

$$\frac{1}{2}\Big((2 + \gamma 10) - 2\Big) + \frac{1}{2}\Big(10 - 0\Big) = 5 + 5\gamma$$

while if we query at s_1 we get an expected regret of

$$\frac{1}{2}\Big((2 + \gamma 10) - \gamma 10\Big) + \frac{1}{2}\Big(10 - 10\Big) = 1\,.$$

We therefore observe that whilst Brown et al. [161] would query s_0, querying s_1 yields a greater reduction in expected regret and also better tightening of the PAC

condition (achieving it for any $\epsilon > 1$ and any δ). For a sufficiently high ϵ, our PAC-EIG acquisition function would correctly query s_1 (though for a small ϵ, it would recognize that both states need to be queried to satisfy the PAC criterion and would be indifferent between them).

Action Entropy [162] In its single-state-annotation version, the value of this acquisition function in a state is equal to the entropy of the posterior predictive distribution over expert actions in that state. Consider a situation where the learner has perfect knowledge of the action values in a particular state, but at least three actions in this state are equivalent (result in the same reward and next state distribution) and tied as optimal. Then the expert Boltzmann policy will assign uniform probabilities among these actions and due to high action entropy this state will always be queried in favour of any where one of two actions is best, but the learner does not know which of the two (assuming all other actions are known to be strongly suboptimal and thus unlikely to get queried). We offer an example of this in Figure 5.1 which renders this acquisition function useless since it will only ever query the jail cell without gaining any information (and this is the case in both the single-state-annotation version, and a trajectory based version), while both our acquisition functions keep gathering useful information and improving the both the posterior reward entropy and the regret of the apprentice policy (including the PAC bounds).

For the single-state-annotation version, an even stronger counter example applies: if we allow querying terminal states in any environment that has them, the method always queries these, since actions have no effect, so a Boltzmann rational policy would be uniform and thus have maximum entropy.

C.2 Theoretical Analysis

We will establish an upper bound on the expected number of expert demonstrations needed to find a policy satisfying an ϵ-δ-PAC criterion using the PAC-EIG acquisition function. As outlined in Section 5.4, we will assume that during the learning

process, the apprentice policy is one maximizing the probability of being $\epsilon(1-\gamma)/2$-approximately correct in each state in the immediate regret sense, i.e.

$$\pi^{\mathrm{A}}(s) = \mathrm{argmax}_a \mathbb{P}[Q*(s,\pi^*(s)) - Q^*(s,a) < \epsilon(1-\gamma)/2 | \mathcal{D}_n],\ \forall s \in \mathcal{S} \qquad \text{(C.7)}$$

. The proof strategy proceeds in three steps:

1. First, we show that if a policy has a significant (overall) regret, there must exist a state where the policy's action is significantly suboptimal in terms of immediate regret. (Lemma 6)

2. Building on this, we prove that if a policy is not (ϵ, δ)-PAC, then there exists a state where the difference between optimal and apprentice policy's optimal Q-values is lower-bounded by a function of ϵ with probability at least $\delta/|\mathcal{S}|$.

3. Finally, we show that in such cases, observing an expert demonstration from an appropriately chosen initial state provides a guaranteed minimum amount of information about whether the policy is approximately correct. Since we can only gain a finite amount of information (bounded by the entropy of our prior), this leads to a bound on the number of demonstrations needed.

We begin with our first lemma, which connects overall policy regret to statewise immediate regret, i.e. differences in optimal Q-values:

Lemma 6. *Let π be any policy, r any reward function, and*

$$R_r^\pi = \mathbb{E}_{s_0 \sim \rho_0} \left[V_r^*(s_0) - V_r^\pi(s_0) \right] \geq 0,$$

the regret of that policy. Then there exists a state $s \in \mathcal{S}$ such that

$$R^*_{\pi,r}(s) := Q_r^*(s, \pi_r^*(s)) - Q_r^*(s, \pi(s)) \geq (1-\gamma) R_r^\pi.$$

Proof. Let us define

$$\Delta_Q = \max_{s \in \mathcal{S}} \left[Q_r^*(s, \pi_r^*(s)) - Q_r^*(s, \pi(s)) \right].$$

We will prove the lemma by showing that $R_r^\pi \leq \Delta_Q/(1-\gamma)$.

Since $Q_r^\pi(s,\pi(s)) \leq Q_r^*(s,\pi(s))$ (because Q_r^* is the optimal Q-function), we have

$$\begin{aligned} V_r^*(s) - V_r^\pi(s) &= Q_r^*(s,\pi_r^*(s)) - Q_r^\pi(s,\pi(s)) \\ &\geq Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi(s)) \\ &\geq 0\,. \end{aligned}$$

Using the Bellman equation, for any state $s \in \mathcal{S}$ we can write

$$\begin{aligned} V_r^*(s) - V_r^\pi(s) &= Q_r^*(s,\pi_r^*(s)) - Q_r^\pi(s,\pi(s)) \\ &= Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi(s)) + Q_r^*(s,\pi(s)) - Q_r^\pi(s,\pi(s)) \\ &\leq \Delta_Q + \Big(r(s,\pi(s)) + \gamma \mathbb{E}_{s'|s,\pi(s)}[V_r^*(s')]\Big) \\ &\qquad - \Big(r(s,\pi(s)) + \gamma \mathbb{E}_{s'|s,\pi(s)} V_r^\pi(s')]\Big) \\ &= \Delta_Q + \gamma \mathbb{E}_{s'|s,\pi(s)}[V_r^*(s') - V_r^\pi(s')] \\ &\leq \Delta_Q + \gamma \max_{s'}[V_r^*(s') - V_r^\pi(s')]. \end{aligned}$$

Here, the first equality just replaces state values by the corresponding Q-values, the second line adds and subtracts the same term, the third line uses the definition of Δ_Q for the first term and expands the latter two Q-values using the Bellman equation, the fourth just cancels out the repeated reward term. The final inequality follows because the expectation over next states is bounded by the maximum.

Since this inequality holds for all $s \in \mathcal{S}$, it holds also for the state maximizing the left-hand side, so we get

$$\max_s [V_r^*(s) - V_r^\pi(s)] \leq \Delta_Q + \gamma \max_{s'}[V_r^*(s') - V_r^\pi(s')]\,, \tag{C.8}$$

which can be readily rearranged into

$$\max_s [V_r^*(s) - V_r^\pi(s)] \leq \frac{\Delta_Q}{1-\gamma}.$$

Thus

$$R_r^\pi = \mathbb{E}_{s_0\sim\rho_0}\left[V_r^*(s_0) - V_r^\pi(s_0)\right] \leq \max_s [V_r^*(s) - V_r^\pi(s)] \tag{C.9}$$

$$\leq \frac{\Delta_Q}{1-\gamma} = \frac{1}{1-\gamma} \max_{s\in\mathcal{S}} \left[Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi(s))\right], \tag{C.10}$$

which completes the proof. □

Lemma 7. *Let π be the apprentice policy at step n. For any $\delta \in (0, \frac{1}{2}]$, let $R^{\pi}_{n,\delta}$ be the $(1-\delta)$-quantile of the regret distribution with respect to the current posterior distribution over rewards, i.e., $R^{\pi}_{n,\delta}$ satisfies*

$$\mathbb{P}_{r|\mathcal{D}_n}[R^{\pi}_r \geq R^{\pi}_{n,\delta}] = \delta.$$

Then, there exists a state $s \in \mathcal{S}$ such that

$$\mathbb{P}_{r|\mathcal{D}_n}\left[Q^*_r(s, \pi^*_r(s)) - Q^*_r(s, \pi(s)) \geq (1-\gamma)R^{\pi}_{n,\delta}\right] \geq \frac{\delta}{|\mathcal{S}|}.$$

Proof. Let us define the set of reward functions under which π has high regret:

$$\mathcal{H} = \left\{r : R^{\pi}_r \geq R^{\pi}_{n,\delta}\right\}.$$

By definition of the quantile $R^{\pi}_{n,\delta}$, we have

$$\mathbb{P}_{r|\mathcal{D}_n}[r \in \mathcal{H}] = \delta.$$

For each $r \in \mathcal{H}$, applying Lemma 6, we know there exists a state $s_r \in \mathcal{S}$ such that

$$Q^*_r(s_r, \pi^*_r(s_r)) - Q^*_r(s_r, \pi(s_r)) \geq (1-\gamma)R^{\pi}_r \geq (1-\gamma)R^{\pi}_{n,\delta}.$$

Let $\mathcal{S}_r$ be the set of such states for a reward function $r \in \mathcal{H}$ and $\mathcal{S}_{\mathcal{H}}$ the collection of such states across all rewards $r \in \mathcal{H}$. Since the state space $\mathcal{S}$ is finite with cardinality $|\mathcal{S}|$, by the pigeonhole principle, there must exist at least one state $s \in \mathcal{S}$ such that

$$\mathbb{P}_{r|\mathcal{D}_n}\left[r \in \mathcal{H} \text{ and } s \in \mathcal{S}_r\right] \geq \frac{\delta}{|\mathcal{S}|}.$$

For this state s, whenever $r \in \mathcal{H}$ and $s \in \mathcal{S}_r$, we have

$$Q^*_r(s, \pi^*_r(s)) - Q^*_r(s, \pi(s)) \geq (1-\gamma)R^{\pi}_{n,\delta}.$$

Therefore,

$$\mathbb{P}_{r|\mathcal{D}_n}\left[Q^*_r(s, \pi^*_r(s)) - Q^*_r(s, \pi(s)) \geq (1-\gamma)R^{\pi}_{n,\delta}\right] \geq \frac{\delta}{|\mathcal{S}|},$$

which completes the proof. □

This lemma extends our previous result to the probabilistic setting of Bayesian IRL. While Lemma 6 showed that high regret implies the existence of a state with poor action choice, this lemma shows that if our policy has a significant probability of high regret, there must be at least one state where it has a significant probability of making a poor action choice.

Now we will build on this lemma to formulate our first theorem to take a step further: if we apply this lemma to a policy that has a significant probability of being approximately correct in each state, but the lemma also gives us a state where it has a significant probability of making a poor choice, we have two contradictory hypotheses that an expert demonstration can help us resolve – we formalize this as a lower bound on the information gain from observing the expert action in such a state.

Theorem 2. *For $\epsilon > 0$ and $\delta \in (0, \frac{1}{2}]$, assume that no policy π is (ϵ, δ)-probably-approximately-correct, i.e., $\mathbb{P}[R_r^\pi \geq \epsilon] > \delta$, $\forall \pi$. Then, there exists a state $s \in \mathcal{S}$ such that observing a new expert demonstration at s has an expected information gain with respect to the variable E^{π^A} (Eq. 5.5) of at least*

$$EIG_{min}(\epsilon, \delta) = \frac{\delta e^{-\beta(1-\gamma)\epsilon}(1 - e^{-\beta(1-\gamma)\epsilon/2})^2}{4|\mathcal{A}|^5|\mathcal{S}|}. \tag{5.10}$$

Proof. Let π^{A} be an apprentice policy that in each state maximizes the probability of being approximately optimal in the immediate regret sense, i.e., $\pi^{\mathrm{A}}(s) \in \operatorname{argmax}_a \mathbb{P}[Q^*(s, \pi^*(s)) - Q^*(s, a) < \epsilon(1-\gamma)/2|\mathcal{D}_n]$, $\forall s \in \mathcal{S}$. To informally outline our proof strategy: we will prove the theorem by showing that under its assumptions, there exists a state s and an alternative action $a' \neq \pi^{\mathrm{A}}(s)$ that has a chance of being significantly better than the apprentice action. If it is significantly better, it is significantly more likely to get selected by the expert than if it is inferior to $\pi^{\mathrm{A}}(s)$. Since the observation distributions in the two cases are different, this allows us to put a lower bound on the expected information gained by observing the expert at s. Now let us turn to the proof in full detail.

Under the assumptions of this theorem and using Lemma 7, there exists a state s such that

$$\mathbb{P}_{r|\mathcal{D}_n}\left[Q_r^*(s, \pi_r^*(s)) - Q_r^*(s, \pi^{\mathrm{A}}(s)) \geq (1-\gamma)\epsilon\right] \geq \frac{\delta}{|\mathcal{S}|}.$$

Conditioning on the event $\Big\{Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\epsilon\Big\}$, let a' be the action that has the highest posterior predictive probability of being optimal. By the pigeonhole principle, this conditional probability of being optimal is at least $\frac{1}{|\mathcal{A}|-1}$. Furthermore, if a' is optimal, then it also has a higher probability of being selected by the Boltzmann-rational expert than any other action, so conditional on being optimal, it must have a probability of at least $1/|\mathcal{A}|$ of being selected. Using just the first fact we get

$$\mathbb{P}\Big[\, Q_r^*(s,a') - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\epsilon\Big] \geq \frac{\delta}{|\mathcal{S}|(|\mathcal{A}|-1)}. \tag{C.11}$$

Using both facts, conditional on $\Big\{Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\epsilon\Big\}$, we have

$$\mathbb{P}\Big[A = a' \text{ and } a' \text{ optimal } |Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\Big] \geq \frac{1}{|A|(|A|-1)} > \frac{1}{|A|^2} \tag{C.12}$$

where A is a random variable representing the action that the expert would choose at s. This implies also

$$\mathbb{P}\left[A = a' | Q_r^*(s,a') - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\epsilon\right] > \frac{1}{|A|^2}\,, \tag{C.13}$$

since the event of a' being optimal is a subset of the event $\{Q_r^*(s,a') - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\}$ conditional on $\{Q_r^*(s,a') - Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq (1-\gamma)\epsilon\}$ and $\{A = a'\}$ is a superset of $\{A = a' \text{ and } a' \text{ optimal}\}$.

On the other hand, since π^{A} was chosen as the policy maximizing the probability of being "strongly approximately correct", we must have

$$\mathbb{P}\left[Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi^{\mathrm{A}}(s)) < (1-\gamma)\epsilon/2\right] > \frac{1}{|\mathcal{A}|}.$$

(Since in each state, under each reward function, one of the $|\mathcal{A}|$ actions must be optimal, at least one action must have a probability of at least $\frac{1}{|\mathcal{A}|}$ of being optimal and thus also strongly approximately correct. Since π^{A} is maximizing this probability, it must also have a probability of at least $\frac{1}{|\mathcal{A}|}$ of being strongly approximately correct).

Since $Q_r^*(s,a') \leq Q_r^*(s,\pi_r^*(s))$, it is that case that $\{Q_r^*(s,\pi_r^*(s)) - Q_r^*(s,\pi^{\mathrm{A}}(s)) < (1-\gamma)\epsilon/2\} \implies \{Q_r^*(s,a') - Q_r^*(s,\pi^{\mathrm{A}}(s)) < (1-\gamma)\epsilon/2\}$, so the probability of the latter event must be at least as high as that of the former event and we have

$$\mathbb{P}\left[Q_r^*(s,a') - Q_r^*(s,\pi^{\mathrm{A}}(s)) < (1-\gamma)\epsilon/2\right] > \frac{1}{|\mathcal{A}|}. \tag{C.14}$$

Now, if we denote by A the action taken by the expert in state s seen as a random variable, we can decompose the mutual information between A and $E_{s,a'}$ as

$$I(A; E_{s,a'}) = \sum_{E_{s,a'} \in \mathcal{E}} \mathbb{P}[E_{s,a'}]\ D_{\mathrm{KL}}(p(A|E_{s,a'}) \| p(A)).$$

where $\mathcal{E} :=$ {"strongly approximately correct", "weakly approximately correct", "not approximately correct"}. Going forward, let us shorten the three options to e_{s},e_w, and e_{n} respectively.

To finish the proof, we need to put a lower bound on this mutual information. We have already given lower bounds on $\mathbb{P}[e_{\mathrm{n}}]$ (Eq. C.11) and $\mathbb{P}[e_{\mathrm{s}}]$ (Eq. C.14). We will now provide a lower bound on the corresponding KL terms by first lower-bounding the total variation distance between $p(A|e_{\mathrm{n}})$ and $p(A|e_{\mathrm{s}})$.

In the event e_{s}, we have $Q_r^*(s,\pi^{\mathrm{A}}(s)) \geq Q_r^*(s,a') - \epsilon(1-\gamma)/2$, so under the Boltzmann-rational policy, for any given reward r_{s} compatible with e_{s}, we have

$$\begin{aligned}
\mathbb{P}\left[A = \pi^{\mathrm{A}}(s) | s; r_{\mathrm{s}}\right] &= \frac{1}{Z} e^{\beta Q_r^*(s,\pi^{\mathrm{A}}(s))} \\
&\geq \frac{1}{Z} e^{\beta(Q_r^*(s,a') - (1-\gamma)\epsilon/2)} \\
&= e^{-\beta(1-\gamma)\epsilon/2} \mathbb{P}\left[A = a' | s; r_{\mathrm{s}}\right] \\
&= \mathbb{P}\left[A = a' | s; r_{\mathrm{s}}\right] - (1 - e^{-\beta(1-\gamma)\epsilon/2}) \mathbb{P}\left[A = a' | s; r_{\mathrm{s}}\right],
\end{aligned}$$

where Z is a normalizing constant, so

$$\mathbb{P}\left[A = a' | s; r_{\mathrm{s}}\right] - \mathbb{P}\left[A = \pi^{\mathrm{A}}(s) | s; r_{\mathrm{s}}\right] \leq (1 - e^{-\beta(1-\gamma)\epsilon/2}) \mathbb{P}\left[A = a' | s; r_{\mathrm{s}}\right].$$

Along similar lines, in case of e_{n} and a reward r_{n} compatible with it, we have

$$\mathbb{P}\left[A = a' | s; r_{\mathrm{n}}\right] - \mathbb{P}\left[A = \pi^{\mathrm{A}}(s) | s; r_{\mathrm{n}}\right] > (1 - e^{-\beta(1-\gamma)\epsilon}) \mathbb{P}\left[A = a' | s; r_{\mathrm{n}}\right].$$

Marginalizing over the reward and using the bound from Equation C.13,

$$\begin{aligned}\mathbb{P}[A = a'|s; e_{\mathrm{n}}] - \mathbb{P}\Big[A = \pi^{\mathrm{A}}(s)|s; e_{\mathrm{n}}\Big] &= \int_{r_{\mathrm{n}}} \Big(\mathbb{P}[A = a'|s; r_{\mathrm{n}}] \\ &\qquad - \mathbb{P}\Big[A = \pi^{\mathrm{A}}(s)|s; r_{\mathrm{n}}\Big]\Big) p(r_{\mathrm{n}}|e_{\mathrm{n}}) \mathrm{d}r_{\mathrm{n}} \\ &\geq \int_{r_{\mathrm{n}}} (1 - e^{-\beta(1-\gamma)\epsilon}) \mathbb{P}\left[A = a'|s; r_{\mathrm{n}}\right] p(r_{\mathrm{n}}|e_{\mathrm{n}}) \mathrm{d}r_{\mathrm{n}} \\ &= (1 - e^{-\beta(1-\gamma)\epsilon}) \mathbb{P}\left[A = a'|s; e_{\mathrm{n}}\right] \\ &> (1 - e^{-\beta(1-\gamma)\epsilon}) \frac{1}{|\mathcal{A}|^2}.\end{aligned}$$

and similarly for the case of e_{s}. We can use this to put a lower bound on the total variation distance between the posterior predictive distributions under the events e_{n} and e_{s}:

$$\begin{aligned}&D_{\mathrm{TV}}(p(A|e_{\mathrm{s}}), p(A)) + D_{\mathrm{TV}}(p(A|e_{\mathrm{n}}), p(A)) \\ &\qquad \geq D_{\mathrm{TV}}(p(A|e_{\mathrm{s}}), p(A|e_{\mathrm{n}})) \\ &\qquad \geq \frac{1}{2}\Big(\mathbb{P}\left[A = \pi^{\mathrm{A}}(s)|e_{\mathrm{s}}\right] - \mathbb{P}\left[A = \pi^{\mathrm{A}}(s)|e_{\mathrm{n}}\right] \\ &\qquad\qquad + \mathbb{P}\left[A = a'|e_{\mathrm{n}}\right] - \mathbb{P}\left[A = a'|e_{\mathrm{s}}\right]\Big) \\ &\qquad = \frac{1}{2}\Big(\mathbb{P}\left[A = a'|e_{\mathrm{n}}\right] - \mathbb{P}\left[A = \pi^{\mathrm{A}}(s)|e_{\mathrm{n}}\right]\Big) \\ &\qquad\qquad - \frac{1}{2}\Big(\mathbb{P}\left[A = a'|e_{\mathrm{s}}\right] - \mathbb{P}\left[A = \pi^{\mathrm{A}}(s)|e_{\mathrm{s}}\right]\Big) \\ &\qquad > \frac{1}{2}(1 - e^{-\beta(1-\gamma)\epsilon}) \frac{1}{|\mathcal{A}|^2} - \frac{1}{2}(1 - e^{-\beta(1-\gamma)\epsilon/2}) \frac{1}{|\mathcal{A}|^2} \\ &\qquad = \frac{1}{2} e^{-\beta(1-\gamma)\epsilon/2}(1 - e^{-\beta(1-\gamma)\epsilon/2}) \frac{1}{|\mathcal{A}|^2}\end{aligned}$$

where we used the triangle inequality in the first step, the definition of total variation distance in the second step (omitting non-negative terms corresponding to actions other than $\pi^{\mathrm{A}}(s)$ and a'), the two inequalities we just derived for e_{n} and e_{s} in the fourth step.

Applying Pinsker's inequality to both TV terms gives us

$$\begin{aligned}&D_{\mathrm{KL}}(p(A|e_{\mathrm{s}}) \| p(A)) + D_{\mathrm{KL}}(p(A|e_{\mathrm{n}}) \| p(A)) \\ &\qquad \geq 2(D_{\mathrm{TV}}(p(A|e_{\mathrm{s}}), p(A))^2 + D_{\mathrm{TV}}(p(A|e_{\mathrm{n}}), p(A))^2) \\ &\qquad \geq (D_{\mathrm{TV}}(p(A|e_{\mathrm{s}}), p(A)) + D_{\mathrm{TV}}(p(A|e_{\mathrm{n}}), p(A)))^2 \\ &\qquad > \frac{1}{4|\mathcal{A}|^4} e^{-\beta(1-\gamma)\epsilon}(1 - e^{-\beta(1-\gamma)\epsilon/2})^2.\end{aligned}$$

where we applied the inequality $(a+b)^2 \leq 2(a^2+b^2)$ in the second step and the previous inequality on the sum of TV distances in the third step.

This finally allows us to establish that

$$\begin{aligned} I(A; E_{s,a'}) &= \sum_{E_{s,a'} \in \{e_\mathrm{n}, e_\mathrm{s}, e_\mathrm{w}\}} \mathbb{P}[E_{s,a'}] D_\mathrm{KL}(p(A|E_{s,a'}) \| p(A)) \\ &\geq \min\{\mathbb{P}[e_\mathrm{s}], \mathbb{P}[e_\mathrm{n}]\} \left(D_\mathrm{KL}(p(A|e_\mathrm{s}) \| p(A)) + D_\mathrm{KL}(p(A|e_\mathrm{n}) \| p(A))\right) \\ &> \min\left\{\frac{1}{|\mathcal{A}|}, \frac{\delta}{(|\mathcal{A}|-1)|\mathcal{S}|}\right\} \frac{1}{4|\mathcal{A}|^2} e^{-\beta(1-\gamma)\epsilon} (1 - e^{-\beta(1-\gamma)\epsilon/2})^2 \\ &= \frac{\delta}{4|\mathcal{A}|^5|\mathcal{S}|} e^{-\beta(1-\gamma)\epsilon} (1 - e^{-\beta(1-\gamma)\epsilon/2})^2. \end{aligned}$$

The first inequality follows from the fact that all three terms in the sum are non-negative. The second inequality just plugs in results previously derived in this proof. The final step resolves the minimum as its second term using the assumption that $\delta \leq \frac{1}{2}$ and $|\mathcal{A}| \geq 2$ and uses the fact that $\frac{1}{|\mathcal{A}|} < \frac{1}{|\mathcal{A}|-1}$.

Since the random variable $E_{s,a'}$ is coarser than the variable E, which is the collection of the variables $E_{s,a}$ across all state-action pairs, we have $I(A; E) \geq I(A; E_{s,a'})$, which completes the proof. □

Now that we have a lower bound on the information that we gain in each step, we can use it to bound the number of steps needed to reach the PAC condition.

Theorem 3. *Let h_{max} be an upper bound on the entropy of the prior distribution over E, the PAC-EIG discretized regret values E^{π^A} aggregated across all apprentice policies π^A. Then, the expected number of steps needed to reach the PAC condition is upper bounded by*

$$\frac{h_{max}}{EIG_{min}(\epsilon, \delta)} = \frac{4h_{max}|\mathcal{A}|^5|\mathcal{S}|}{\delta e^{-\beta(1-\gamma)\epsilon}(1 - e^{-\beta(1-\gamma)\epsilon/2})^2}. \tag{5.11}$$

Proof. The minimal expected information gain guaranteed by Theorem 2 is fully derived from components of the random variable E. Thus, at every step of active learning where we have not yet achieved the PAC criterion, we can gain at least $\mathrm{EIG}_{\min}(\epsilon, \delta)$ information about E.

Let H_n denote the expected entropy of E after n steps of active learning. By the properties of entropy and information gain:

1. $H_n \geq 0$ for all n (non-negativity of entropy)

2. $H_0 = h_{\text{prior}}$ (initial entropy)

3. $H_{n+1} \leq H_n - \text{EIG}_{\min}(\epsilon, \delta)$ for all n where the PAC criterion is not met (guaranteed information gain)

Let N be the number of steps needed to reach the PAC criterion. Then:

$$\begin{aligned} 0 &\leq H_N \\ &\leq H_0 - N \cdot \text{EIG}_{\min}(\epsilon, \delta) \\ &= h_{\text{prior}} - N \cdot \text{EIG}_{\min}(\epsilon, \delta) \end{aligned}$$

Solving for N:

$$N \leq \frac{h_{\text{prior}}}{\text{EIG}_{\min}(\epsilon, \delta)} \tag{C.15}$$

The result follows by substituting the expression for $\text{EIG}_{\min}(\epsilon, \delta)$ from Theorem 2.

□

Corollary 8. *For any prior distribution over rewards, the expected number of steps to reach the PAC condition is at most*

$$\log(3)|\mathcal{S}||\mathcal{A}|^2 / EIG_{min}(\epsilon, \delta) = \frac{4\log(3)|\mathcal{A}|^7|\mathcal{S}|^2}{\delta e^{-\beta(1-\gamma)\epsilon}(1 - e^{-\beta(1-\gamma)\epsilon/2})^2}. \tag{C.16}$$

Proof. The random variable E aggregates $|\mathcal{S}||\mathcal{A}|^2$ ternary random variables (in each state, apprentice policies can take $|\mathcal{A}|$ actions, and we are considering each action's immediate regret relative to $|\mathcal{A}| - 1 < |\mathcal{A}|$ alternative actions), so E can take at most $3^{|\mathcal{S}||\mathcal{A}|}$ values. Thus, its maximum entropy is $-\log(1/3^{|\mathcal{S}||\mathcal{A}|^2}) = |\mathcal{S}||\mathcal{A}|^2 \log(3)$. The result follows by plugging this maximum entropy into Theorem 3. □

While we do not claim this bound is tight, it provides a useful characterization of how the sample complexity scales with the problem parameters. In particular, it shows polynomial dependence on the size of the state and action spaces, and inverse dependence on both the allowed suboptimality ϵ and failure probability δ. More importantly, on a qualitative level, the result shows that the learning

process does continue as long as the PAC condition is not satisfied, so it does not get stuck forever querying an uninformative state as is the case with e.g. the acquisition function of Kweon et al. [162].

C.2.1 Notes on Possible Improvements

Tighter bound for large state spaces

Note that the bound from Lemma 7 can be tightened if the state space is large and only a subset is reachable within an effective horizon. In that case $|S|$ can be replaced by the number of states reachable from the initial states within $1/(1-\gamma)$ steps. Also, if there is a limited horizon, or the apprentice policy always reaches a terminal state in a certain number of steps, we can also have fewer steps than that (and can further reduce the $(1-\gamma)$ factor in the exponent of our theoretical results.

Static policy

The bound includes the entropy of each action in each state. In fact, it may be enough to focus on a single action in each state, since we want to identify only a particular PAC policy, rather than reducing entropy of all of the components of $E(s)$ in every state. This should allow us to exclude a factor of $|A|$ from the bound on the expected number of steps to reach the PAC condition.

C.3 Accounting for Visitation Frequencies

A practically useful acquisition function should account for one more important point: in order to get a policy with low expected regret, we do not need to reduce the expected immediate regret of all points, but just those that are likely to get visited by the apprentice policy π^{A}. Let $\nu_A(s) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}[s_t = s]]$ be the discounted expected occupancy of state s. Then, we can set

$$\tilde{\alpha}(s) := \nu_A(s) I(A_s, E_s). \tag{C.17}$$

to be the acquisition function for querying single states.

However, we also wish to have an acquisition function for collecting full trajectories. A naive approach employed by some prior work [162] would be summing the individual

$$\alpha'_n(s_0) := \mathbb{E}_{r|\mathcal{D}_n}\left[\mathbb{E}_{\tau|r}\left[\sum_{s\in\tau}\tilde{\alpha}(s)\right]\right] \tag{C.18}$$

However, the sum in Eq. (C.18) neglects correlation between the regrets in different states (and, worse, autocorrelation if a state is visited multiple times). We can instead estimate the full expected information gain about our variable E from the new expert trajectory. Then, dropping the weighing for the moment,

$$\mathrm{EIG}_E(s_0) = \mathbb{E}_{\tau,E}\Big[\log p(\tau|E) - \log p(\tau)\Big]. \tag{C.19}$$

We assume that the Bayesian IRL method we use to estimate this is able to give us samples from the current posterior over Q-values. Given a Q-value and an initial state, we can sample the corresponding hypothetical expert trajectories $\tau|Q$. Also, E is a cheaply-computable function of Q, so we can also easily convert the samples of Q into samples of E. Then, the only remaining challenge in computing EIG_E is estimating $p(\tau|E)$.

If the state space is small and we have a lot of Q samples, we can estimate $p(\tau|E) = \frac{1}{|\mathcal{Q}_E|}\sum_{Q\in\mathcal{Q}_E} p(\tau|Q)$. However, note that for a given policy, there are $3^{|}S|$ possible values of E^π, so even for a moderate size of the state space, the number of Q corresponding to each E could be small. At the same time, the components of E^π corresponding to states far away from the trajectory are unlikely to share much mutual information with the trajectory. Thus we suggest using the bound

$$I(\tau, E) \geq I(\tau, E^\pi_\tau) \tag{C.20}$$

where $E^\pi_\tau := \Big(E_s\Big)_{s\in\mathcal{S}_\tau}$ for $\mathcal{S}_\tau$ being some neighbourhood of τ in the state space, including all states on the trajectory τ plus states that can be quickly reached from the trajectory.

To again add the weighing, we can note that since the transition probabilities are the same for both components,

$$\log p(\tau|E) - \log p(\tau) = \sum_{s,a\in\tau} \log p(a|s;E) - \log p(\tau|s) \tag{C.21}$$

which naturally allows us to re-introduce the weights as

$$\sum_{s,a\in\tau} \nu_\pi(s,a)(\log p(a|s;E) - \log p(\tau|s)) \tag{C.22}$$

resulting in an acquisition function

$$\mathrm{EIG}_{\nu\text{-PAC}}(s_0) = \mathbb{E}_{\tau,E}\Big[\sum_{s,a\in\tau} \nu_\pi(s,a)(\log p(a|s;E_\tau) - \log p(\tau|s))\Big]. \tag{C.23}$$

C.4 Alternative Acquisition Functions

In this appendix, we discuss alternative formulations of acquisition functions for active IRL, which could help to see the reasoning process that led to PAC-EIG, and explain why some, possibly more, obvious alternatives were not chosen.

C.4.1 From Apprentice Return to Regret

When the goal is to produce a well-performing apprentice policy (as opposed to learning the reward for its own sake), a natural starting point is to directly optimize the apprentice's expected return. This suggests minimizing the loss:

$$\mathcal{L}_{\mathrm{ret}}(\xi_1,\dots,\xi_N) = -\mathbb{E}_{s_0\sim\rho}\mathbb{E}_{\tau|s_0,\pi_N^{\mathrm{A}}} G_r(\tau) \tag{C.24}$$

where π_N^{A} is the apprentice policy after observing N expert trajectories from initial states $\xi_1,\dots,\xi_N$.

Since the optimal value $V^*(s_0)$ is independent of our choice of queries, minimizing $\mathcal{L}_{\mathrm{ret}}$ is equivalent to minimizing the regret loss:

$$\mathcal{L}_{\mathrm{reg}}(\xi_1,\dots,\xi_N) = R_r^{\pi_N^{\mathrm{A}}} = \mathbb{E}_{s_0\sim\rho}\left[V_r^*(s_0) - V_r^{\pi_N^{\mathrm{A}}}(s_0)\right] \tag{C.25}$$

C.4.2 The Challenge of Direct Regret Optimization

Directly optimizing this regret loss is computationally intractable even in the greedy case. The one-step acquisition function would be:

$$\alpha_n^{\mathrm{reg}}(\xi) = -\mathbb{E}_{r|\mathcal{D}_n}\mathbb{E}_{\tau_{n+1}|\xi,\pi_r^{\mathrm{E}}} R_r^{\pi_{n+1}^{\mathrm{A}}} \tag{C.26}$$

Computing this requires: 1. For each possible reward r in our posterior 2. For each possible expert trajectory τ from initial state ξ 3. Computing the updated posterior $p(r|\mathcal{D}_n \cup \{\tau\})$ 4. Finding the optimal apprentice policy for this updated posterior 5. Evaluating its regret

This nested optimization involving repeated Bayesian IRL updates is prohibitively expensive.

C.4.3 Information Gain About Regret

Following the approach in Bayesian optimization [187], rather than directly optimizing the hard-to-compute expected improvement, we can instead maximize information gain about the quantity of interest. This suggests the acquisition function:

$$\alpha_n^{\text{Regret-EIG}}(s_0) = I(\tau; R_r^{\pi^A}|s_0, \mathcal{D}_n) \tag{C.27}$$

However, this formulation has a critical flaw. Consider this example:

- In state s, the apprentice can take action a_0 yielding return 0
- Actions a_1 and a_2 yield returns of +50 and -100, but we don't know which is which
- With equal probability on both orderings, the apprentice chooses a_0 (expected return 0 vs -25)
- The regret is known with certainty to be 50
- Since there's no uncertainty about regret, Regret-EIG assigns zero value to querying this state
- Yet the apprentice is definitely choosing suboptimally!

C.4.4 Immediate Regret EIG

The solution is to decompose regret more finely. The total regret can be written as:

$$R_r^{\pi^A} = \mathbb{E}_{\tau\sim\rho,\pi^A} \sum_{s_t,a_t\in\tau} \gamma^t \underbrace{\left[V_r^*(s_t) - Q_r^*(s_t, \pi^A(s_t))\right]}_{R^*_{\pi^A,r}(s_t)} \tag{C.28}$$

where $R^*_{\pi^A,r}(s)$ is the *immediate regret* – the value lost by following the apprentice policy in state s without considering future consequences.

This can be further decomposed per action:

$$R^*_{\pi^A,r}(s) = \max_a R^*_{\pi^A,r}(s,a) \tag{C.29}$$

where $R^*_{\pi^A,r}(s,a) = \max\{0, Q_r^*(s,a) - Q_r^*(s,\pi^A(s))\}$.

The Immediate Regret EIG acquisition function is then:

$$\alpha_n^{\text{IR-EIG}}(s_0) = I(\tau; R^*|s_0, \mathcal{D}_n) \tag{C.30}$$

where $R^* = (R^*_{\pi^A,r}(s,a))_{s\in\mathcal{S},a\in\mathcal{A}}$.

This formulation correctly identifies informative states in our earlier example, as there is high uncertainty about which action has higher immediate regret.

C.4.5 Discretization and Connection to PAC-EIG

For practical computation, the continuous immediate regret values must be discretized. Different discretization schemes lead to different acquisition functions:

1. **Multi-bucket discretization**: Using buckets of [0, $\epsilon/2$], [$\epsilon/2$, ϵ], [ϵ, ∞) like in PAC-EIG can be taken further to allow for a finer approximation of the IR-EIG. This can be viable for single state-queries, but growing the number of categories becomes untenable once we start considering the full trajectory demonstration.

2. **Two-bucket PAC discretization**: Using just two buckets – acceptable regret $[0, \epsilon]$ and unacceptable regret (ϵ, ∞) – directly captures what matters for PAC guarantees. However, the theoretical arguments as shown above do

not directly apply to this case, since we loose the middle bucket that ensures separation between the two sufficiently different expert action distributions. However, we do think a weaker result could be proven even for this case.

The PAC discretization is not only computationally more tractable but also theoretically motivated: it focuses information gathering on exactly what we need to know to provide formal reliability guarantees.

C.4.6 Summary

The progression from expected return optimization to PAC-EIG illustrates how principled information-theoretic thinking, combined with practical computational constraints and theoretical objectives, leads to an effective acquisition function. While IR-EIG with fine discretization might provide marginally more information in some cases, PAC-EIG strikes the optimal balance between theoretical guarantees, computational efficiency, and practical effectiveness.

C.5 Experiment Details

C.5.1 Basic Parameter Values

In the three environments (structured 6x6, random 8x8, and random 10x10) we used $\beta = 4, 2, 4$ respectively, $\gamma = 0.9$, and an infinite horizon (but all environments contained terminal states). We started with an empty set of demonstrations (implemented as a single, uninformative observation of a dummy sink state) and then ran active learning for 150 steps.

For ActiveVaR, we used $\delta = 0.05$ (same as the original paper). For policy entropy, we used the entropy of the discretized distribution for each action (as proposed by the authors) with K=10 buckets. For our PAC-EIG acquisition function we used $(1 - \gamma)\epsilon = 0.01$ for the PAC condition.

C.5.2 Environments

The gridworld environments have 5 actions, corresponding to staying in place and moving in the four directions. Furthermore, there is a probability of 0.1 of random action being executed instead of the intended one. If an action would result in crossing the edge, the agent instead remains in place. The gridworlds use a state-only reward (awarded upon executing any action in the given state).

The 8x8, and 10x10 fully random environments were generated as follows:

1. Each state was assigned a random reward drawn independently from $\mathcal{N}(0, 3)$ (i.e. mostly yielding rewards between -10 and 10).

2. Each state was then marked as terminal with an independent probability of 0.1.

3. The top 10% of states with highest reward were further marked as terminal (producing terminal goal states, which may, however, sometimes be avoided by the optimal policy in favour of staying forever in other positive states).

4. The initial state distribution is either uniform across the whole state space, or, in the case of the 10x10 gridworld, 2 non-terminal initial states were chosen randomly uniformly.[1]

C.5.3 Bayesian IRL Methods

Our active learning uses a Bayesian IRL method as a key component. In our experiments, we used two methods based on Markov chain Monte Carlo (MCMC) sampling: on the structured environment, we used PolicyWalk [101], while on the environment with a different random reward in every state, we used the faster ValueWalk [7], which performs the sampling primarily in the space of Q-functions before converting into rewards. We also tried a method based on variational inference [102], but we found its uncertainty estimates unreliable for the purposes of active learning.

[1]Note that the implementation allows the two initial states to collide, producing only a single initial state in 1/81 of the cases, but this was not the case for any of our 16 random seeds.

For MCMC sampling, we used Hamiltonian Monte Carlo [137] with the no-U-turns (NUTS) sampler [140] and automatic step size selection during warm-up (starting with a step size of 0.1). At every step of active learning, we ran the MCMC sampling from scratch using all demonstrations available up to that point. We ran for 100 warm-up steps and then 200, 500, and 1000 on the three environments respectively. For subsequent usage, we use every other sample to reduce autocorrelation.

C.5.4 Metrics

On the first two environments, we use KNN entropy estimation to calculate posterior entropy with K=5. This method is known to struggle in high dimensions, which we also observed in the case of the 10x10 gridworld (which has a 100-dimensional reward space), so there, we estimate the entropy by the entropy of a multivariate normal distribution with the mean and covariance matrix estimated from the MCMC samples.

Regret was calculated relative to the expected return of the optimal policy, calculated using value iteration with a tolerance of 1e-5. Posterior regret samples were similarly calculated relative to the optimal return with respect to each of the posterior reward samples (which were calculated using the optimal Q-value samples which get produced by the Bayesian IRL methods).

When aggregating true regret across environment instances, we also normalized the regret for each random environment instance by the average regret across all methods across the first 32 steps of active learning to account for the possibly different scales and different learning difficulties of the random environments.

C.5.5 Implementation

The experiments were implemented using Python 3.10, PyTorch 2.5.1, and Pyro 1.8.6. We will publish our full code for both the experiments and the associated result analysis on Github once the anonymity requirement is lifted.

C.5.6 Timing

The computational time per step of active IRL is dominated by the time necessary to collect the Bayesian IRL MCMC samples, which ranges between 5 seconds for the 100+200 samples on the structured gridworld to about 5 minutes for the 100+1000 samples on the 10x10 gridworld in a single CPU thread. The overhead of all acquisition functions on top of that is below 0.03 and can thus be considered negligible.

Reproducing all our experiments thus takes less than a day on a CPU with 128 threads (we used AMD Ryzen Threadripper 3990X at 2.2GHz).

C.6 Notation Overview

Table C.1: Summary of notation used throughout the paper. If reward is omitted from a symbol otherwise depending on it, it means it is taken with respect to the true reward.

Symbol	Meaning
$\mathcal{S}$	State spstate ace of the MDP
$\mathcal{A}$	Action space
$P(s' \mid s, a)$	Transition kernel
$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	Expected reward function
$\gamma \in (0, 1)$	Discount factor
$t_{\max}$	Maximum horizon (may be ∞)
ρ	Initial-state distribution
π^E	Expert policy (Boltzmann-rational with coefficient β)
π^E_r	Hypothetical expert policy corresponding to a reward r
β	Boltzmann rationality coefficient
$\mathcal{D}_n$	Demonstration data after n queries
$\pi^{\mathrm{A}},\ \pi^{\mathrm{A}}_n$	Apprentice policy (after n queries)
$\pi^\star_r$	Optimal policy for reward r
$\tau = (s_0, a_0, \ldots, s_T)$	Trajectory
ξ	Query (initial state for the next expert demonstration)
$V^\pi_r(s)$, $Q^\pi_r(s, a)$	State and action-value functions for policy π and reward r
$G_r(\tau)$	Discounted return of trajectory τ under r
G^π_r	Expected discounted return of π under r, P, and ρ
$R^\pi_r(s_0)$	Regret of π from state s_0
$R^\star_{\pi,r}(s)$, $R^\star_{\pi,r}(s, a)$	Immediate regret of π for state s or state–action pair s, a
$I(\tau; r \mid s_0, \mathcal{D}_n)$	Mutual information between a trajectory and reward
EIG	Expected information gain
$\alpha^{\mathrm{RewEIG}}_n$	Reward-EIG acquisition function
VaR_δ	δ-value-at-risk of a loss random variable
$EIG_{\min}(\epsilon, \delta)$	Per-step information-gain lower bound (Thm 3)
$h_{\max}$	Upper bound on the prior entropy of E
$\epsilon,\ \delta$	PAC accuracy / confidence parameters
$\lvert\mathcal{S}\rvert,\ \lvert\mathcal{A}\rvert$	Cardinalities of state and action spaces

D

Negative Human Rights as a Basis for AI Safety and Regulation

The thesis has focused on the technical problem of inferring human values or preferences – irrespective of whether they are positive goals – like getting a coffee or finding a cure to cancer – or negative constraints like not knocking over a vase or wiping out humanity in the process. But this neglects a crucial aspect of the societal context in which AI alignment needs to happen – while positive goals can differ substantially person to person (getting rich vs helping the poor vs proving the Riemann hypothesis), there is a subset of negative constraints that we may wish to impose on all AI systems, especially preventing harm.

In our JAIR paper co-authored with Jan Horenovsky [6], we propose leveraging the framework of negative human rights as a basis for both AI safety regulation and the technical specification of safety constraints. We argue that negative rights, which protect individuals from harmful interference (e.g., rights to life, property, security of person), possess several properties making them suitable for this role. Firstly, they represent a set of principles with relatively broad international consensus, often already codified in existing legal systems globally. This existing consensus is crucial for establishing widely accepted regulatory frameworks necessary for managing risks from advanced AI. Relying on such an *overlapping consensus* [188]

avoids the difficulty of specifying a complete, universally agreed-upon set of positive human values, which vary significantly across cultures and individuals. The focus is thus shifted from the ambitious goal of aligning AI with the entirety of human values to the more tractable, minimalist goal of ensuring AI systems do not violate fundamental, widely recognized boundaries.

Secondly, the proposal draws inspiration from legal systems, particularly the process of adjudication, to give concrete meaning to potentially abstract rights principles. A behaviour is considered compliant if a defined adjudication system (potentially incorporating AI assistance in the future) would judge it so. This framing aims to create a system capable of providing definitive judgements (permissible/impermissible) even in novel situations, mirroring how legal systems interpret laws. While most human rights function as relative principles requiring balancing in context, the legal framework provides mechanisms for resolving conflicts and arriving at authoritative decisions.

Thirdly, this perspective naturally frames the task of ensuring compliance as a technical problem amenable to machine learning. An AI system, or a dedicated safety module, would need to predict whether its planned actions carry an unacceptable risk of violating negative human rights, based on its representation of the plan and the context. The output could be conceptualized as an expected penalty associated with rights violations, which the system should seek to minimize or keep below a stringent threshold. Training data for such a predictor could come from diverse sources, including case law, statutes, direct human feedback (potentially simulating the adjudication process), observation of human behaviour (inferring implicit constraints), large text corpora, and formal specifications where applicable.

Crucially, these rights-based constraints are intended to function with primacy over the AI's primary objectives. The system should be designed such that the pursuit of its positive goals cannot justify actions leading to significant risks of rights violations. This requires careful integration to avoid adversarial optimization, where the system might seek loopholes or ways to manipulate the safety mechanism or the adjudication process itself – a challenge related to specification gaming and

corrigibility. Conservatism in the face of uncertainty is also vital; the system should default to avoiding actions whose permissibility is uncertain.

While distinct from inverse reinforcement learning's focus on inferring reward functions representing positive preferences, this constraint-based approach motivated by human rights addresses a critical aspect of the alignment problem: reliably preventing AI systems from causing harm by grounding safety in established, broadly accepted normative principles. It complements reward/preference learning by establishing boundaries within which AI systems should operate. The challenge of robustly learning and enforcing such constraints motivates further research into areas including reliable uncertainty quantification and value alignment, connecting to the themes explored in this thesis.

I provide a full version of this argument in the paper attached as a supplement to the thesis for interested readers.

References

[1] Michael K. Cohen, Marcus Hutter, and Michael A. Osborne. "Advanced Artificial Agents Intervene in the Provision of Reward". In: *AI Magazine* 43.3 (2022).

[2] Joar Skalse, Nikolaus Howe, Dmitrii Krasheninnikov, and David Krueger. "Defining and Characterizing Reward Gaming". In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022.

[3] Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. *Specification Gaming: The Flip Side of AI Ingenuity*. 2020.

[4] Stuart Russell. *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking, 2019.

[5] Richard Ngo, Lawrence Chan, and Sören Mindermann. "The Alignment Problem from a Deep Learning Perspective". In: *International Conference on Learning Representations*. 2024.

[6] Ondrej Bajgar and Jan Horenovsky. "Negative Human Rights as a Basis for Long-Term AI Safety and Regulation". In: *Journal of Artificial Intelligence Research* 76 (2023).

[7] Ondrej Bajgar, Konstantinos Gatsis, Alessandro Abate, and Michael A. Osborne. "Walking the Values in Bayesian Inverse Reinforcement Learning". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 2024.

[8] Ondrej Bajgar, Peter Tisnikar, Alessandro Abate, Konstantinos Gatsis, and Maike Osborne. "Q-Based Variational Inverse Reinforcement Learning". In: *Reinforcement Learning Journal* (2026).

[9] Ondrej Bajgar, Dewi Sid William Gould, Jonathon Liu, Oliver Newcombe, Rohan Narayan Langford Mitta, and Jack Golden. "Toward Information Theoretic Active Inverse Reinforcement Learning". In: *NeurIPS 2024 Workshop on Bayesian Decision-Making and Uncertainty*. 2024.

[10] Ondrej Bajgar, Dewi Sid William Gould, Jonathon Liu, Alessandro Abate, Konstantinos Gatsis, and Michael A. Osborne. "PAC Apprenticeship Learning with Bayesian Active Inverse Reinforcement Learning". In: *Reinforcement Learning Journal* (2025).

[11] Joar Max Viktor Skalse and Alessandro Abate. "The Reward Hypothesis Is False". In: *NeurIPS ML Safety Workshop*. 2022.

[12] Micah Carroll, Davis Foote, Anand Siththaranjan, Stuart Russell, and Anca Dragan. "AI Alignment with Changing and Influenceable Reward Functions". In: *Proceedings of the International Conference on Machine Learning*. 2024.

[13] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. "Cooperative Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[14] Norbert Wiener. "Some Moral and Technical Consequences of Automation". In: *Science* 131.3410 (1960).

[15] Nick Bostrom. *Superintelligence: Paths, Dangers, Strategies*. New York, NY, US: Oxford University Press, 2014.

[16] Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. *Risks from Learned Optimization in Advanced Machine Learning Systems*. 2019. arXiv: 1906.01820 [cs].

[17] Pulkit Agrawal. "The Task Specification Problem". In: *Proceedings of the Conference on Robot Learning*. PMLR, 2022.

[18] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. *Instruction Tuning for Large Language Models: A Survey*. 2024. arXiv: 2308.10792 [cs].

[19] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. "Deep Reinforcement Learning from Human Preferences". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[20] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. "Training Language Models to Follow Instructions with Human Feedback". In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022.

[21] Uma E. Sarkar. "Evaluating Alignment in Large Language Models: A Review of Methodologies". In: *AI and Ethics* (2025).

[22] Richard Bellman. "A Markovian Decision Process". In: *Journal of Mathematics and Mechanics* 6.5 (1957). JSTOR: 24900506.

[23] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

[24] Chen Tang, Ben Abbatematteo, Jiaheng Hu, Rohan Chandra, Roberto Martín-Martín, and Peter Stone. "Deep Reinforcement Learning for Robotics: A Survey of Real-World Successes". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 27. 2025.

[25] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. *Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback*. 2022. arXiv: 2204.05862 [cs].

[26] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. "A Survey of Reinforcement Learning from Human Feedback". In: *Transactions on Machine Learning Research* (2025).

[27] Matt Barnes, Matthew Abueg, Oliver F. Lange, Matt Deeds, Jason Trader, Denali Molitor, Markus Wulfmeier, and Shawn O'Banion. "Massively Scalable Inverse Reinforcement Learning in Google Maps". In: *International Conference on Learning Representations*. 2024.

[28] David Silver, Satinder Singh, Doina Precup, and Richard S. Sutton. "Reward Is Enough". In: *Artificial Intelligence* 299 (2021).

[29] Sriraam Natarajan and Prasad Tadepalli. "Dynamic Preferences in Multi-Criteria Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. ICML '05. New York, NY, USA: Association for Computing Machinery, 2005.

[30] Joar Skalse and Alessandro Abate. "On the Limitations of Markovian Rewards to Express Multi-Objective, Risk-Sensitive, and Modal Tasks". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. PMLR, 2023.

[31] Daniel Kahneman and Amos Tversky. "Prospect Theory: An Analysis of Decision under Risk". In: *Econometrica* 47.2 (1979). JSTOR: 1914185.

[32] Owain Evans, Andreas Stuhlmueller, and Noah D. Goodman. "Learning the Preferences of Ignorant, Inconsistent Agents". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2016.

[33] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. *Concrete Problems in AI Safety*. 2016. arXiv: 1606.06565 [cs].

[34] Jack Clark and Dario Amodei. *Faulty Reward Functions in the Wild*. 2016.

[35] Jonathan Stray. "Aligning AI Optimization to Community Well-Being". In: *International Journal of Community Well-Being* 3.4 (2020).

[36] Manoel Horta Ribeiro, Raphael Ottoni, Robert West, Virgílio A. F. Almeida, and Wagner Meira. "Auditing Radicalization Pathways on YouTube". In: *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. FAT* '20. New York, NY, USA: Association for Computing Machinery, 2020.

[37] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd. USA: Prentice Hall Press, 2009.

[38] Casey Chu, Andrey Zhmoginov, and Mark Sandler. *CycleGAN, a Master of Steganography*. 2017. arXiv: 1712.02950 [cs].

[39] Adrian Thompson. "An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics". In: *Evolvable Systems: From Biology to Hardware*. Ed. by Tetsuya Higuchi, Masaya Iwata, and Weixin Liu. Berlin, Heidelberg: Springer, 1997.

[40] Stuart Armstrong and Benjamin Levinstein. *Low Impact Artificial Intelligences*. 2017. arXiv: 1705.10720 [cs].

[41] Alexander Matt Turner, Dylan Hadfield-Menell, and Prasad Tadepalli. "Conservative Agency via Attainable Utility Preservation". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2020. arXiv: 1902.09725.

[42] Eliezer Yudkowsky. "Complex Value Systems in Friendly AI". In: *Artificial General Intelligence*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Jürgen Schmidhuber, Kristinn R. Thórisson, and Moshe Looks. Vol. 6830. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[43] Dylan Hadfield-Menell and Gillian Hadfield. "Incomplete Contracting and AI Alignment". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2019. arXiv: 1804.04268 [cs].

[44] Alexander Pan, Kush Bhatia, and Jacob Steinhardt. "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models". In: *International Conference on Learning Representations*. 2022.

[45] Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. *Goal Misgeneralization: Why Correct Specifications Aren't Enough For Correct Goals*. 2022. arXiv: 2210.01790 [cs].

[46] Lauro Langosco, Jack Koch, Lee D. Sharkey, Jacob Pfau, and David Krueger. "Goal Misgeneralization in Deep Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2022.

[47] Lukas Fluri, Leon Lang, Alessandro Abate, Patrick Forré, David Krueger, and Joar Skalse. "The Perils of Optimizing Learned Reward Functions: Low Training Error Does Not Guarantee Low Regret". In: *Proceedings of the International Conference on Machine Learning*. 2025. arXiv: 2406.15753 [cs].

[48] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (2016).

[49] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. "A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play". In: *Science* 362.6419 (2018).

[50] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A Physics Engine for Model-Based Control". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012.

[51] Serena Booth, W. Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, and Alessandro Allievi. "The Perils of Trial-and-Error Reward Design: Misdesign through Overfitting and Invalid Task Specifications". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 5. 2023.

[52] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. "Inverse Reward Design". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[53] Ellis Ratner, Dylan Hadfield-Menell, and Anca Dragan. "Simplifying Reward Design through Divide-and-Conquer". In: *Proceedings of Robotics: Science and Systems*. Vol. 14. 2018.

[54] Sören Mindermann, Rohin Shah, Adam Gleave, and Dylan Hadfield-Menell. *Active Inverse Reward Design*. 2019. arXiv: `1809.03060 [cs]`.

[55] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models Are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020.

[56] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. "PaLM: Scaling Language Modeling with Pathways". In: *Journal of Machine Learning Research* 24.240 (2023).

[57] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu,

Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, C. J. Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. *GPT-4 Technical Report*. 2024. arXiv: `2303.08774 [cs]`.

[58] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza,

Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. "Super-NaturalInstructions: Generalization via Declarative Instructions on 1600+ NLP Tasks". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, 2022.

[59] Zachary Kenton, Tom Everitt, Laura Weidinger, Iason Gabriel, Vladimir Mikulik, and Geoffrey Irving. "Alignment of Language Agents". In: *arXiv:2103.14659 [cs]* (2021). arXiv: `2103.14659 [cs]`.

[60] Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. "MINT: Evaluating LLMs in Multi-Turn Interaction with Tools and Language Feedback". In: *International Conference on Learning Representations*. 2024.

[61] Markus Wulfmeier, Michael Bloesch, Nino Vieillard, Arun Ahuja, Jörg Bornschein, Sandy Huang, Artem Sokolov, Matt Barnes, Guillaume Desjardins, Alex Bewley, Sarah M. Bechtle, Jost T. Springenberg, Nikola Momchev, Olivier Bachem, Matthieu Geist, and Martin Riedmiller. "Imitating Language via Scalable Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 37. 2024.

[62] Erdem B{\i}y{\i}k, Malayandi Palan, Nicholas C. Landolfi, Dylan P. Losey, and Dorsa Sadigh. "Asking Easy Questions: A User-Friendly Approach to Active Reward Learning". In: *Proceedings of the Conference on Robot Learning*. PMLR, 2020.

[63] Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. "SURF: Semi-Supervised Reward Learning with Data Augmentation for Feedback-Efficient Preference-Based Reinforcement Learning". In: *International Conference on Learning Representations*. 2022.

[64] W. Bradley Knox, Stephane Hatgis-Kessell, Serena Booth, Scott Niekum, Peter Stone, and Alessandro Allievi. "Models of Human Preference for Learning Reward Functions". In: *Transactions on Machine Learning Research* (2024).

[65] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: `1707.06347 [cs]`.

[66] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. "Direct Preference Optimization: Your Language Model Is Secretly a Reward Model". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023.

[67] Stefan Schaal. "Learning from Demonstration". In: *Advances in Neural Information Processing Systems*. Vol. 9. MIT Press, 1996.

[68] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, Gabriel Dulac-Arnold, John Agapiou, Joel Leibo, and Audrunas Gruslys. "Deep Q-Learning From Demonstrations". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[69] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Overcoming Exploration in Reinforcement Learning with Demonstrations". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2018.

[70] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, and Matthew E Taylor. "Reinforcement Learning from Demonstration through Shaping". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2015.

[71] Pieter Abbeel and Andrew Y. Ng. "Apprenticeship Learning via Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. ICML '04. New York, NY, USA: Association for Computing Machinery, 2004.

[72] Stephen Adams, Tyler Cody, and Peter A. Beling. "A Survey of Inverse Reinforcement Learning". In: *Artificial Intelligence Review* (2022).

[73] Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. "A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges". In: *IEEE Transactions on Cybernetics* 54.12 (2024).

[74] Dean A. Pomerleau. "Efficient Training of Artificial Neural Networks for Autonomous Navigation". In: *Neural Computation* 3.1 (1991).

[75] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, 2011.

[76] Quanyi Li, Zhenghao Peng, and Bolei Zhou. "Efficient Learning of Safe Driving Policy via Human-AI Copilot Optimization". In: *International Conference on Learning Representations*. 2022.

[77] Ryan Hoque, Ashwin Balakrishna, Ellen Novoseller, Albert Wilcox, Daniel S. Brown, and Ken Goldberg. "ThriftyDAgger: Budget-Aware Novelty and Risk Gating for Interactive Imitation Learning". In: *Proceedings of the Conference on Robot Learning*. PMLR, 2022.

[78] Robert Dadashi, Leonard Hussenot, Matthieu Geist, and Olivier Pietquin. "Primal Wasserstein Imitation Learning". In: *International Conference on Learning Representations*. 2021.

[79] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. "SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards". In: *International Conference on Learning Representations*. 2020.

[80] Kiante Brantley, Wen Sun, and Mikael Henaff. "Disagreement-Regularized Imitation Learning". In: *International Conference on Learning Representations*. 2020.

[81] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014.

[82] Jonathan Ho and Stefano Ermon. "Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems*. Vol. 29. Curran Associates, Inc., 2016.

[83] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. "Multi-Agent Generative Adversarial Imitation Learning". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[84] Kyungjae Lee, Sungjoon Choi, and Songhwai Oh. "Maximum Causal Tsallis Entropy Imitation Learning". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[85] Manu Orsini, Anton Raichuk, Leonard Hussenot, Damien Vincent, Robert Dadashi, Sertan Girgin, Matthieu Geist, Olivier Bachem, Olivier Pietquin, and Marcin Andrychowicz. "What Matters for Adversarial Imitation Learning?" In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021.

[86] Raunak Bhattacharyya, Blake Wulfe, Derek J. Phillips, Alex Kuefler, Jeremy Morton, Ransalu Senanayake, and Mykel J. Kochenderfer. "Modeling Human Driving Behavior Through Generative Adversarial Imitation Learning". In: *IEEE Transactions on Intelligent Transportation Systems* 24.3 (2023).

[87] Stuart Russell. "Learning Agents for Uncertain Environments (Extended Abstract)". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. Madison Wisconsin USA: ACM, 1998.

[88] R. E. Kalman. "When Is a Linear Control System Optimal?" In: *Journal of Basic Engineering* 86.1 (1964).

[89] Nematollah Ab Azar, Aref Shahmansoorian, and Mohsen Davoudi. "From Inverse Optimal Control to Inverse Reinforcement Learning: A Historical Review". In: *Annual Reviews in Control* 50 (2020).

[90] Saurabh Arora and Prashant Doshi. "A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress". In: *Artificial Intelligence* 297 (2021).

[91] Saurabh Deshpande, Rahee Walambe, Ketan Kotecha, Ganeshsree Selvachandran, and Ajith Abraham. "Advances and Applications in Inverse Reinforcement Learning: A Comprehensive Review". In: *Neural Computing and Applications* (2025).

[92] Andrew Ng and Stuart Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. 2000.

[93] Haoyang Cao, Samuel Cohen, and Lukasz Szpruch. "Identifiability in Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021.

[94] Joar Max Viktor Skalse, Matthew Farrugia-Roberts, Stuart Russell, Alessandro Abate, and Adam Gleave. "Invariance in Policy Optimisation and Partial Identifiability in Reward Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2023.

[95] Nathan D. Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. "Maximum Margin Planning". In: *Proceedings of the International Conference on Machine Learning*. New York, NY, USA: Association for Computing Machinery, 2006.

[96] Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. "Maximum Entropy Inverse Reinforcement Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2008.

[97] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. "Modeling Interaction via the Principle of Maximum Causal Entropy". In: *Proceedings of the International Conference on Machine Learning*. ICML'10. Madison, WI, USA: Omnipress, 2010.

[98] Navid Aghasadeghi and Timothy Bretl. "Maximum Entropy Inverse Reinforcement Learning in Continuous State Spaces with Path Integrals". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011.

[99] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. *Maximum Entropy Deep Inverse Reinforcement Learning*. 2016. arXiv: `1507.04888 [cs]`.

[100] E. T. Jaynes. "Information Theory and Statistical Mechanics". In: *Physical Review* 106.4 (1957).

[101] Deepak Ramachandran and Eyal Amir. "Bayesian Inverse Reinforcement Learning". In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2007.

[102] Alex J Chan and Mihaela van der Schaar. "Scalable Bayesian Inverse Reinforcement Learning". In: *International Conference on Learning Representations*. 2021.

[103] Gergely Neu and Csaba Szepesvári. "Apprenticeship Learning Using Inverse Reinforcement Learning and Gradient Methods". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. UAI'07. Arlington, Virginia, USA: AUAI Press, 2007.

[104] Joar Skalse, Lucy Farnik, Sumeet Ramesh Motwani, Erik Jenner, Adam Gleave, and Alessandro Abate. "STARC: A General Framework for Quantifying Differences between Reward Functions". In: *International Conference on Learning Representations*. 2024.

[105] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. "A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models". In: *arXiv:1611.03852 [cs]* (2016). arXiv: `1611.03852 [cs]`.

[106] Nathan Ratliff, Brian Ziebart, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. "Inverse Optimal Heuristic Control for Imitation Learning". In: *Proceedings of the International Conference on Artificial Intelligence and Statistics*. PMLR, 2009.

[107] Kyriacos Shiarlis, Joao Messias, and Shimon Whiteson. "Inverse Reinforcement Learning from Failure". In: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. AAMAS '16. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[108] Christos Dimitrakakis and Constantin A. Rothkopf. "Bayesian Multitask Inverse Reinforcement Learning". In: *Recent Advances in Reinforcement Learning*. Ed. by Scott Sanner and Marcus Hutter. Berlin, Heidelberg: Springer, 2012.

[109] Jaedeug Choi and Kee-Eung Kim. "Nonparametric Bayesian Inverse Reinforcement Learning for Multiple Reward Functions". In: *Advances in Neural Information Processing Systems*. 2012.

[110] Bernard Michini and Jonathan P. How. "Bayesian Nonparametric Inverse Reinforcement Learning". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Peter A. Flach, Tijl De Bie, and Nello Cristianini. Vol. 7524. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[111] Malayandi Palan, Gleb Shevchuk, Nicholas Charles Landolfi, and Dorsa Sadigh. "Learning Reward Functions by Integrating Human Demonstrations and Preferences". In: *Proceedings of Robotics: Science and Systems*. Robotics: Science and Systems Foundation, 2019.

[112] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. "Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition". In: *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc., 2018.

[113] J. Kolter, Pieter Abbeel, and Andrew Ng. "Hierarchical Apprenticeship Learning with Application to Quadruped Locomotion". In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2007.

[114] Risto Vuorio, Mattie Fellows, Cong Lu, Clémence Grislain, and Shimon Whiteson. *A Bayesian Solution To The Imitation Gap*. 2024. arXiv: `2407.00495 [cs]`.

[115] Shehryar Malik, Usman Anwar, Alireza Aghasi, and Ali Ahmed. "Inverse Constrained Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2021.

[116] Guiliang Liu, Sheng Xu, Shicheng Liu, Ashish Gaurav, Sriram Ganapathi Subramanian, and Pascal Poupart. "A Comprehensive Survey on Inverse Constrained Reinforcement Learning: Definitions, Progress and Challenges". In: *Transactions on Machine Learning Research* (2024).

[117] Adriana Hugessen, Harley Wiltzer, and Glen Berseth. "Simplifying Constraint Inference with Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2024.

[118] Zhuo-Jun Jin, Hui Qian, and Miao-Liang Zhu. "Gaussian Processes in Inverse Reinforcement Learning". In: *2010 International Conference on Machine Learning and Cybernetics*. Vol. 1. 2010.

[119] Sergey Levine, Zoran Popovic, and Vladlen Koltun. "Nonlinear Inverse Reinforcement Learning with Gaussian Processes". In: *Advances in Neural Information Processing Systems*. 2011.

[120] Kyungjae Lee, Sungjoon Choi, and Songhwai Oh. "Inverse Reinforcement Learning with Leveraged Gaussian Processes". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2016.

[121] Ming Jin, Andreas Damianou, Pieter Abbeel, and Costas Spanos. "Inverse Reinforcement Learning via Deep Gaussian Process". In: *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. 2017. arXiv: 1512.08065.

[122] Qifeng Qiao and Peter A. Beling. "Inverse Reinforcement Learning with Gaussian Process". In: *Proceedings of the 2011 American Control Conference*. 2011.

[123] Sreejith Balakrishnan, Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Harold Soh. "Efficient Exploration of Reward Functions in Inverse Reinforcement Learning via Bayesian Optimization". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[124] Mahdi Imani and Seyede Fatemeh Ghoreishi. "Scalable Inverse Reinforcement Learning Through Multifidelity Bayesian Optimization". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[125] Jiangchuan Zheng, Siyuan Liu, and Lionel M. Ni. "Robust Bayesian Inverse Reinforcement Learning with Sparse Behavior Noise". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[126] Jaedeug Choi and Kee-eung Kim. "MAP Inference for Bayesian Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 24. Curran Associates, Inc., 2011.

[127] Daehyung Park, Michael Noseworthy, Rohan Paul, Subhro Roy, and Nicholas Roy. "Inferring Task Goals and Constraints Using Bayesian Nonparametric Inverse Reinforcement Learning". In: (2020).

[128] Ran Wei, Siliang Zeng, Chenliang Li, Alfredo Garcia, Anthony McDonald, and Mingyi Hong. "A Bayesian Approach to Robust Inverse Reinforcement Learning". In: *CoRL*. 2023. arXiv: 2309.08571 [cs].

[129] Noah Topper, Alvaro Velasquez, and George Atia. *Bayesian Inverse Reinforcement Learning for Non-Markovian Rewards*. 2024. arXiv: 2406.13991 [cs].

[130] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (1953).

[131] Bernard Michini and Jonathan P. How. "Improving the Efficiency of Bayesian Inverse Reinforcement Learning". In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2012.

[132] Stuart Geman and Donald Geman. "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-6.6 (1984).

[133] Aishwarya Mandyam, Didong Li, Andrew Jones, Diana Cai, and Barbara E Engelhardt. "KD-BIRL: Kernel Density Bayesian Inverse Reinforcement Learning". In: *Transactions on Machine Learning Research* (2024).

[134] Di Zeng, Ling Zheng, Xiantong Yang, and Yinong Li. "Uncertainty-Aware Human-like Driving Policy Learning with Deep Bayesian Inverse Reinforcement Learning". In: *Transportmetrica A: Transport Science* (2024).

[135] Chao Yu, Jiming Liu, and Hongyi Zhao. “Inverse Reinforcement Learning for Intelligent Mechanical Ventilation and Sedative Dosing in Intensive Care Units”. In: *BMC Medical Informatics and Decision Making* 19.2 (2019).

[136] Yuang Cai, Yuyu Yuan, Jinsheng Shi, and Qinhong Lin. “Approximated Variational Bayesian Inverse Reinforcement Learning for Large Language Model Alignment”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 39. 22. 2025.

[137] Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. “Hybrid Monte Carlo”. In: *Physics Letters B* 195.2 (1987).

[138] Daniel Jarrett, Ioana Bica, and {van der Schaar}, Mihaela. “Strictly Batch Imitation Learning by Energy-Based Distribution Matching”. In: *Advances in Neural Information Processing Systems*. 2020.

[139] W. K. Hastings. “Monte Carlo Sampling Methods Using Markov Chains and Their Applications”. In: *Biometrika* 57.1 (1970).

[140] Matthew D Hoffman and Andrew Gelman. “The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo”. In: *Journal of Machine Learning Research* 15 (2014).

[141] Edward Snelson, Carl Edward Rasmussen, and Zoubin Ghahramani. “Warped Gaussian Processes”. In: *Advances in Neural Information Processing Systems*. Vol. 16. MIT Press, 2003.

[142] Philipp Hennig, Michael A. Osborne, and Hans P. Kersting. *Probabilistic Numerics*. Cambridge University Press, 2022.

[143] Andrew Gelman and Donald B. Rubin. “Inference from Iterative Simulation Using Multiple Sequences”. In: *Statistical Science* 7.4 (1992).

[144] Henrik Kretzschmar, Markus Spies, Christoph Sprunk, and Wolfram Burgard. “Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning”. In: *The International Journal of Robotics Research* 35.11 (2016).

[145] Billy Okal and Kai O. Arras. “Learning Socially Normative Robot Navigation Behaviors with Bayesian Inverse Reinforcement Learning”. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. 2016.

[146] Bryce Woodworth, Francesco Ferrari, Teofilo E. Zosa, and Laurel D. Riek. “Preference Learning in Assistive Robotics: Observational Repeated Inverse Reinforcement Learning”. In: *Proceedings of the 3rd Machine Learning for Healthcare Conference*. PMLR, 2018.

[147] Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. “Model-Based Inverse Reinforcement Learning from Visual Demonstrations”. In: *Proceedings of the Conference on Robot Learning*. PMLR, 2021.

[148] Wentao Liu, Junmin Zhong, Ruofan Wu, Bretta L Fylstra, Jennie Si, and He Helen Huang. “Inferring Human-Robot Performance Objectives During Locomotion Using Inverse Reinforcement Learning and Inverse Optimal Control”. In: *IEEE Robotics and Automation Letters* 7.2 (2022).

[149] Liting Sun, Wei Zhan, and Masayoshi Tomizuka. "Probabilistic Prediction of Interactive Driving Behavior via Hierarchical Inverse Reinforcement Learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[150] Sascha Rosbach, Vinit James, Simon Großjohann, Silviu Homoceanu, and Stefan Roth. "Driving with Style: Inverse Reinforcement Learning in General-Purpose Planning for Automated Driving". In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2019.

[151] Zhiyu Huang, Jingda Wu, and Chen Lv. "Driving Behavior Modeling Using Naturalistic Human Driving Data With Inverse Reinforcement Learning". In: *IEEE Transactions on Intelligent Transportation Systems* 23.8 (2022).

[152] Tung Phan-Minh, Forbes Howington, Ting-Sheng Chu, Sang Uk Lee, Momchil S. Tomov, Nanxiang Li, Caglayan Dicle, Samuel Findler, Francisco Suarez-Ruiz, Robert Beaudoin, Bo Yang, Sammy Omari, and Eric M. Wolff. *Driving in Real Life with Inverse Reinforcement Learning*. 2022. arXiv: `2206.03004 [cs]`.

[153] Kuno Kim, Shivam Garg, Kirankumar Shiragur, and Stefano Ermon. "Reward Identification in Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2021.

[154] Divyansh Garg, Shuvam Chakraborty, Chris Cundy, Jiaming Song, and Stefano Ermon. "IQ-Learn: Inverse Soft-Q Learning for Imitation". In: *Advances in Neural Information Processing Systems*. Vol. 34. Curran Associates, Inc., 2021.

[155] M. C. Jones and A. Pewsey. "Sinh-Arcsinh Distributions". In: *Biometrika* 96.4 (2009).

[156] Charles E. Clark. "The Greatest of a Finite Set of Random Variables". In: *Operations Research* 9.2 (1961). JSTOR: `167488`.

[157] Zhiyun Lu, Eugene Ie, and Fei Sha. *Mean-Field Approximation to Gaussian-Softmax Integral with Application to Uncertainty Estimation*. 2021. arXiv: `2006.07584 [cs, stat]`.

[158] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. arXiv: `1312.6114 [cs, stat]`.

[159] Alberto Maria Metelli, Giorgia Ramponi, Alessandro Concetti, and Marcello Restelli. "Provably Efficient Learning of Transferable Rewards". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2021.

[160] Manuel Lopes, Francisco Melo, and Luis Montesano. "Active Learning for Reward Estimation in Inverse Reinforcement Learning". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[161] Daniel S. Brown, Yuchen Cui, and Scott Niekum. "Risk-Aware Active Inverse Reinforcement Learning". In: *Proceedings of the Conference on Robot Learning*. PMLR, 2018.

[162] Sehee Kweon, Himchan Hwang, and Frank C. Park. "Trajectory-Based Active Inverse Reinforcement Learning for Learning from Demonstration". In: *2023 23rd International Conference on Control, Automation and Systems (ICCAS)*. 2023.

[163] Zoe Ashwood, Aditi Jha, and Jonathan W. Pillow. "Dynamic Inverse Reinforcement Learning for Characterizing Animal Behavior". In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022.

[164] Dylan P. Losey and Marcia K. O'Malley. "Including Uncertainty When Learning from Human Corrections". In: *Proceedings of the Conference on Robot Learning*. PMLR, 2018.

[165] David Lindner, Andreas Krause, and Giorgia Ramponi. "Active Exploration for Inverse Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 35. 2022.

[166] Thomas Kleine Buening, Victor Villin, and Christos Dimitrakakis. "Environment Design for Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2024.

[167] Thomas Kleine Büning, Anne-Marie George, and Christos Dimitrakakis. "Interactive Inverse Reinforcement Learning for Cooperative Games". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2022.

[168] Dorsa Sadigh, Anca Dragan, Shankar Sastry, and Sanjit Seshia. "Active Preference-Based Learning of Reward Functions". In: *Proceedings of Robotics: Science and Systems*. Robotics: Science and Systems Foundation, 2017.

[169] Abi Komanduru and Jean Honorio. "A Lower Bound for the Sample Complexity of Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. 2021.

[170] Alberto Maria Metelli, Filippo Lazzati, and Marcello Restelli. "Towards Theoretical Understanding of Inverse Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2023.

[171] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[172] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. "A Survey of Deep Active Learning". In: *ACM Comput. Surv.* 54.9 (2021).

[173] David D. Lewis and William A. Gale. "A Sequential Algorithm for Training Text Classifiers". In: *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '94. Berlin, Heidelberg: Springer-Verlag, 1994.

[174] Nicholas Roy and Andrew McCallum. "Toward Optimal Active Learning through Sampling Estimation of Error Reduction". In: *Proceedings of the International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[175] David J. C. MacKay. "Information-Based Objective Functions for Active Data Selection". In: *Neural Computation* 4.4 (1992).

[176] Tom Rainforth, Adam Foster, Desi R. Ivanova, and Freddie Bickford Smith. *Modern Bayesian Experimental Design*. 2023. arXiv: `2302.14545 [cs, stat]`.

[177] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. *Bayesian Active Learning for Classification and Preference Learning*. 2011. arXiv: 1112.5745 [stat].

[178] Adam X. Yang, Maxime Robeyns, Thomas Coste, Zhengyan Shi, Jun Wang, Haitham Bou Ammar, and Laurence Aitchison. "Bayesian Reward Models for LLM Alignment". In: *ICML 2024 Workshop on Structured Probabilistic Inference and Generative Modeling*. 2024.

[179] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. "Pyro: Deep Universal Probabilistic Programming". In: *Journal of Machine Learning Research* 20.28 (2019).

[180] Philipp Hennig. *Expectation Propagation on the Maximum of Correlated Normal Variables*. 2009. arXiv: 0910.0115 [stat.ML].

[181] C. Visweswariah, K. Ravindran, K. Kalafala, S. G. Walker, and S. Narayan. "First-Order Incremental Block-Based Statistical Timing Analysis". In: *Proceedings of the 41st Annual Design Automation Conference*. DAC '04. New York, NY, USA: Association for Computing Machinery, 2004.

[182] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. "Gymnasium: A Standard Interface for Reinforcement Learning Environments". In: *arXiv preprint arXiv:2407.17032* (2024). arXiv: 2407.17032.

[183] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-Level Control through Deep Reinforcement Learning". In: *Nature* 518.7540 (2015).

[184] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. "Distributional Reinforcement Learning with Quantile Regression". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. New Orleans, Louisiana, USA: AAAI Press, 2018.

[185] Nino Vieillard, Olivier Pietquin, and Matthieu Geist. "Munchausen Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Vol. 33. Curran Associates, Inc., 2020.

[186] Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. "Implicit Quantile Networks for Distributional Reinforcement Learning". In: *Proceedings of the International Conference on Machine Learning*. PMLR, 2018.

[187] Zi Wang and Stefanie Jegelka. "Max-Value Entropy Search for Efficient Bayesian Optimization". In: *Proceedings of the International Conference on Machine Learning*. 2017.

[188] John Rawls. "The Idea Of An Overlapping Consensus". In: *Oxford Journal of Legal Studies* 7.1 (1987).