Abstract of "Explaining Reinforcement Learning Agents by Policy Comparison" by Jun Ki Lee, Ph.D., Brown University, October, 2022.

Reinforcement learning (RL) techniques have led to remarkable results in challenging domains such as Atari games, Go, and Starcraft, suggesting that practical applications lie just over the horizon. Before we can trust decisions made by RL policies, however, we need more visibility into how they work. To explain a reinforcement-learning agent, I propose extending the power of counterfactual reasoning to sequential domains by comparing its policy to a baseline policy at a set of automatically identified decision points. My novel method for selecting important decision points considers a large pool of candidate states and decomposes the agent's value into the reward obtained before vs. after visiting that state. A state is considered important if the accumulated reward obtained after switching to the baseline policy is most different from that obtained after continuing its policy. The engine of this computation is a decomposition of occupancy frequencies of an agent's policy that characterize the whereabouts of an agent before and after the policy change. Structuring the policy evaluation in this way provides a causal account for its outcome. I have demonstrated the approach on a set of standard RL benchmark domains, providing explanations using the decomposed occupancy frequencies.

Explaining Reinforcement Learning Agents by Policy Comparison

by

Jun Ki Lee

B. S., Seoul National University, 2006

M. S., Massachusetts Institute of Technology, 2009

M. S., Brown University, 2015

A dissertation submitted in partial fulfillment of the

requirements for the Degree of Doctor of Philosophy

in the Department of Computer Science at Brown University

Providence, Rhode Island

October, 2022

This dissertation by Jun Ki Lee is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Michael L. Littman, Director

Recommended to the Graduate Council

Date _____ _____

George Konidaris, Reader

Date _____ _____

Stephen Bach, Reader

Approved by the Graduate Council

Date _____ _____

Thomas A. Lewis
Dean of the Graduate School

Vita

Jun Ki Lee graduated Summa Cum Laude from Seoul National University in 2006 with a B.S. in Computer Science and Engineering. He then went to Boston, MA and attended Massachusetts Institute of Technology, where he received his Masters in Media Arts and Sciences in 2009. In 2013, he began working toward his Ph.D. in Computer Science under Dr. Michael L. Littman. En route to his Ph.D., he received his Masters in Computer Science at Brown University in 2015.

During his undergraduate years, he attended four Asian Regional ACM Internationl Collegiate Programming Contests and has won the 1st, 3rd and two 5th places with his colleagues. He was also affiliated with Samsung Software Membership in Seoul, South Korea and received the Silver and Bronze awards with his colleagues. He also has won the scholarships for academic excellence from Seoul National University.

For his Master's study, he received the graduate study abroad scholarship, supported by Samsung Scholarship Foundation. He also owns two U.S. patents: *Interactive Systems Employing Robotic Companions* [86] and *Methods of Robot Behavior Generation and Robots Utilizing the Same* [27].

To my beloved wife Hyewon and my son Daniel for their unwavering support

Acknowledgements

First of all, I would like to thank my advisor Michael L. Littman. He introduced me to the field Reinforcement Learning which I always longed for since I started to study abroad. He guided me through my Ph.D. studies with his wisdom.

I would also like to thank my readers George Konidaris and Stephen Bach. I also thank Srinath Sridhar and Stefanie Tellex for coming to my defense talk. Your questions greatly helped me furthering my view.

I would like to thank my office mates, Carl Trimbach, Nakul Gopalan, Royal Wang, Elizabeth Hilliard, and Stephen Brawner.

I would also like to thank my lab mates, Lucas Lehnert, David Abel, Kavosh Asadi, Dilip Arumugam, Mark Ho, Sam Saarinen, and Omer Gottesman.

I would also like to my Brown friends, Dokook Choi, Yeon-oh Chung, Jiwon Choi, Karthik Desingh, Zhiqiang Sui, Kevin Du, Sophie Saskin, and Matt Cooper.

I would like to thank David Jensen, Sam Witty, Emma Tosch, and Akanksha Atrey at UMass Amherst for research collaboration.

Last but not least, I would like to thank my wife Hyewon Kim and my son Daniel

Hyunjoon Lee. Without them, I would have never made this far. I also a owe great debt of gratitude to my parents and my parents in law and I would also like to thank my other family members.

Funding Support This dissertation was supported in part by the DARPA XAI project.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

With the invention of fast graphical processing units (GPUs) [52] and new deep learning techniques [50, 49, 23], there has been rapid progress in reinforcement learning. Learned agents can play some Atari games at a superhuman level [61, 20, 21] and defeat human experts in games like Go, Poker, and Starcraft [83, 63, 96]. However, there is still little understanding of how these systems work, both among machine-learning experts and domain experts. This dissertation focuses on sequential decision-making strategies learned by reinforcement-learning agents, so I revisit the current developments in this area and investigate the meaning of explainable AI (XAI) for illuminating the behavior of reinforcement-learning agents.

Some of my work relates to probing the generalization capacity of a deep RL algorithm called deep Q networks (DQN) [102, 103]. Investigating what happens inside a deep artificial neural network after training is an example of an approach

to explanation called post-hoc methods. The semantic state perturbations used in this work are specifically designed to support counterfactual reasoning in explaining the learned agent. The results show how the algorithm genuinely fails to generalize with respect to these semantic perturbations in its input states, thus disabling the counterfactual reasoning between actual states and imaginary states. Given the lack of evidence that high-level explainable structures are found within the network, users have only weak insights into how the agents work.

According to Pearl [69] and others, causality plays an essential role in human explanation. In this work, I extend the power of counterfactual reasoning to sequential domains by comparing a learned policy to a baseline policy at a set of automatically identified decision points. This novel method for selecting important decision points considers a large pool of candidate states and decomposes the agent's value into the reward obtained before vs. after visiting that state. A state is considered important if the accumulated reward obtained after switching to the baseline policy is most different from that obtained by continuing with the original policy. The engine of this computation is a decomposition of occupancy frequencies of an agent's policy that characterize the whereabouts of an agent before and after the policy change. Structuring the policy evaluation in this way provides a causal account for its outcome. I have demonstrated the approach on a set of standard RL benchmark domains, showing that the behavior of such an agent can be meaningfully explained, leading to more trustworthy and transparent AI.

Main Context

Counterfactual reasoning is central to how people make explanations. Attempting to generate explanations for reinforcement-learning agents using only their value and policy networks is unlikely to work because these networks are not designed to support counterfactual reasoning.

Thesis Statement

By comparing two policies, instead of just explaining one, it is possible to adapt counterfactual reasoning so that it applies to reinforcement-learning agents. The resulting automatically-generated explanations can help people better understand the relative strengths and weaknesses of the policies constructed in the reinforcement-learning process.

Chapter 2

Backgrounds

This chapter defines and describes the major foundational concepts this dissertation builds on. First, it explains what reinforcement learning is and the definition of Markov decision processes. Descriptions of the more recent deep versions of reinforcement learning techniques follow. Second, it tries to disambiguate the meaning of "Explainable AI (XAI)" and investigates recent related work in the context of supervised learning and reinforcement learning. Lastly, causality is introduced due to its importance in human explanations. Studies of explanations are discussed.

2.1 Reinforcement Learning

What distinguishes reinforcement learning from other topics in machine learning is that it addresses learning by acting in an environment and receiving reward in a delayed manner [87]. The decision-making of a reinforcement-learning agent is

generally modeled using the formalization of Markov Decision Processes.

2.1.1 Markov Decision Process

A Markov Decision Process (MDP) is defined by a tuple $\langle S, A, T, R, \gamma \rangle$. The set S denotes all possible states in the agent's environment and S_0 is a set of possible start states for learning and acting episodes. The transition function $T : S, A \rightarrow S$ maps a state-and-action pair to its next state. If $s \in S$ and $T(s, a, s') = 0$ for $\forall s', a$, s is considered a terminal state in that an agent in this state is completed its learning or acting episode. The set of all terminal states is denoted S_T. In this dissertation, the reward function $R : S \rightarrow \mathbb{R}$ is defined only for arrival states regardless of its initial state and an action, $R(s)$. The scalar γ is a discount rate. The objective of an agent is to maximize the accumulated sum of rewards geometrically discounted by γ.

A policy, $\pi : S \rightarrow A$, is a mapping from states to actions, fully characterizing the behavior of an agent. The Q-value of a state–action pair, $Q_\pi(s, a)$, is the expected return for following π from s after taking action a, $\mathbb{E}_\pi \left[\sum_{k=1}^{\infty} \gamma^k R(s_{t+k}) \mid s_t = s, a_t = a \right]$. The value of a state, $V_\pi(s)$, is the expected return by following π from s, $Q_\pi(s, \pi(s))$. The optimal policy π^* is the policy π that maximizes $V_\pi(s), \forall s \in S$, which is equivalent to maximizing $Q_\pi(s, a) \forall s, a \in S, A$. The value of a state for a fixed policy π is given as $V_\pi(s) = Q(s, \pi(a))$.

For more background on MDPs, please see the book by Puterman [70].

2.1.2 Advances in Reinforcement Learning

A widely used class of methods for deriving policies in RL constructs an approximation of the value function represented as a function from a state–action pair to a value, $\hat{Q}(s, a)$, and then selects the action that maximizes $\hat{Q}(s, a)$ at each timestep [87]. Deep Q-networks (DQNs) are one such method, using multi-layer artificial neural networks as a nonlinear function approximation for $\hat{Q}(s, a)$ [61]. Another widely used class of methods leverages a parameterized function, $\pi_\theta(s, a)$, which can be used to choose an action without looking up a value function. The only criteria for the parameterized function is that it needs to be differentiable with respect to its parameters so that it can be trained with the policy gradient algorithm [88]. If the action space is continuous, the function can directly output actions and for MDPs with discrete action spaces the parameterized numerical preferences, $h_\theta(s, a) \in \mathbb{R}$, can be learned and used to give the actions with the highest preferences the highest probabilities by utilizing the exponential soft-max distribution, $\pi(s, a) = \frac{e^{h_\theta(s,a)}}{\sum_b e^{h_\theta(s,b)}}$.

Deep RL algorithms generally require a differentiable state-value parameterization function, $\hat{V}_{\theta_v}(s)$, and a differentiable policy parameterization function, $\pi_{\theta_p}(s, a)$, is additionally required for policy gradient algorithms [87].

A common testbed for these algorithms is the Arcade Learning Environment (ALE) [6], which supports a collection of several dozen Atari video games, instrumented to allow agents to learn to maximize score. Early DQN algorithms used enhancements such as prioritized experience replay, double Q learning, and dueling networks [76, 92,

98]. Later algorithms used distributional value functions [7], noisy DQN, and n-step bootstrapping. The combination of these techniques shows superior performance [35] compared to the original "vanilla" DQN algorithm. Distributed versions of these algorithms such as Ape-X, Gorilla, and R2D2 [38, 65] show even better performance than their single-threaded counterparts.

Compared with the above value-based algorithms, policy-gradient algorithms such as A3C [62], TRPO [79], ACER [99], ACKTR [104], PPO [80], IMPALA [20], and SEED [21] use an algorithm structure called *actor critic* for learning. Similar to value-based methods, this approach maintains a value function approximation to be consulted to learn the policy function. However, actor-critic algorithms keep an explicit representation of the policy, which is updated as part of the learning process. Actor-critic methods often use distributed actors to play multiple games simultaneously. Since policy-based methods such as these can handle continuous action spaces, this set of algorithms are often used in the environments where actions take the form of a vector of real numbers [10].

Model-based deep RL algorithms followed quickly after the seminal DQN work. To date, learning a model of the environment and optimizing behavior in the learned model remains an inferior approach compared to the model-free methods mentioned above [28, 42]. The Predictron learns to predict its next state, reward, and discount values using its internal representations to aid the planning procedure [84]. The model still takes a full MDP, but its representation can be an internal hidden layer values of

a deep network. TreeQN [22] and value iteration networks [89] learn an abstract and local MDP model, respectively, and bring tree-structured models and value iteration into the realm of deep neural networks. Value prediction networks learn an MDP grounded on real actions, although its representations are not the original inputs [68]. MuZero [78] is a successor to the value prediction networks algorithm in that it learns an MDP with real actions and also learns reward and values functions together. It has has shown superior performance on Atari testbeds.

2.2 Explainable Artificial Intelligence (XAI)

As artificial intelligence systems are more widely used in society, the importance of explaining these systems has grown rapidly, especially in the areas of decision-making systems and supervised learning systems [24, 59, 64, 60, 53, 72, 18, 36, 75]. There is yet no general consensus in what *explainable AI* is and researchers interchangeably use two terms, *explainability* and *interpretability*. In this dissertation, I use both terms without significant distinction between them.

2.2.1 Interpretability and Explainability

Lipton [53] provides the desiderata of the interpretable machine learning (ML): trust, causality, transferability, informativeness, and fair and ethical decision-making. Lipton [53] also distinguishes two properties of interpretable ML: *transparency* and *post-hoc explanations*.

Transparency in this context refers to asking "how does a model work?" A model here refers to deep neural networks, decision trees, linear regression, or any other type of learned structure. Transparency requires the person receiving the explanation to have prior knowledge about the workings of the specific choice of machine-learning model. Transparency can be achieved at the level of an entire model (simulatability), at the level of subcomponents (decomposability), or at the level of an algorithm (algorithmic transparency).

Post-hoc explanations are given to users by extracting information from learned models while a given model is assumed to be a black box. Such explanations include natural language explanations, visualization of learned representations, and explanations by example. Miller [59] and Biran and Cotton [9] make a distinction between *explanations* and *justifications*. Justifications can explain why a given model is making good choices, but cannot explain individual decisions.

In most cases, deep neural networks are considered to be less transparent due to their dependence on the value of their network parameters to encode behavior, despite the fact that all their numerical calculations are fully known [72]. There is numerous work in building alternative models to sit alongside the original model to explain deep neural networks in a more interpretable way. Such examples include a linear proxy model called LIME [71] and a decision-tree method called DeepRED [109]. Visualizing learned features in deep neural networks at different levels of their hierarchies has already been examined by many researchers in the field. Researchers have seen signs

of textures, shape patterns, and parts of objects [66]. Kim et al. [45]'s work is an early attempt to systematize the process of finding these features through concept activation vectors (CAV) and translating them into meaning explanations. In terms of generating images from text inputs using a Generative Adversarial Network (GAN), Hong et al. [37] used a semantically hierarchical network structure instead of a end-to-end network to improve the network's transparency. To better explain deep neural networks' decision-making, pixel level visualization using heatmaps, sometimes called saliency maps, and semantic segmentation have also been used extensively [72].

2.2.2 Explainable Reinforcement Learning (XRL)

Milani et al. [58] classify the previous work in explainable reinforcement learning (XRL) into three categories: feature importance (FI), learning process and MDP (LPM), and policy-level (PL). On the other hand, Sequeira et al. [81] separate XRL methodologies into three topics: environment, interaction and meta analysis. Milani et al. [58] also suggest two categorizations for interpretabability: *intrinsic* vs. *post-hoc* and *local* vs. *global*. These dimensions can be related to the two properties of interpretable ML from the previous section: *intrinsic* interpretability is in general considered to be more transparent and *post-hoc* interpretability is the same in both categorizations. Local interpretability mostly refers to looking into one-step analysis in MDP settings and global interpretability seeks to look in a holistic view and is similar to the notion of meta-analysis.

I explain below the current limitations of XRL methods and how some of the existing work relates to the methods introduced in this dissertation based on Milani et al.'s survey [58].

FI mostly deals with direct relationships between state and actions that are generally represented as a policy function $\pi : S \rightarrow A$. The first approach is to directly model a policy function with intrinsically interpretable policies. The most common method is to use decision trees (DTs). Notable results include Silva et al. [82]'s differentiable soft decision trees and Topin and Veloso [90]'s CUSTARD, an augmented MDP method designed so its action set includes actions for constructing a DT. However, DTs are not suitable for high-dimensional states and each element in a DT only divides states parallel to its axes. Alternative approaches in building intrinsically interpretable policies include using logical expressions and representing policies with more complex functions and fuzzy logics [34, 48, 33, 108].

Instead of learning interpretable policies directly, other researchers tried to convert an existing policy to similarly interpretable formats as the above methods [41, 8, 107]. All these approaches still share the pitfall that, as their explanations get more complex, they become innately harder to comprehend, resulting in high cognitive load for the user. Also, the performance of more simplified interpretable models may not meet the performance the original policy to be explained.

Through forms of natural language and saliency maps, researchers have tried to directly generate explanations from the learned models. Based on agents' state

features, Ehsan and Riedl [19] trained a network to separately output natural language explanations using an LSTM (long short-term memory), and Wang et al. [97] used outputs of an attention network to generate template-based natural language explanations. Hayes and Shah [32] used templates to generate explanations using heuristic algorithms. Although their approach helps to increase the trust of a user, they do not provide evidence that these explanations are grounded in the agents' actual decision-making. My method is also template-based. However, it uses a counterfactual approach to strengthen the explanations compared to prior methods.

Wang et al. [98], Greydanus et al. [26], and Weitkamp et al. [100] developed methods to display saliency maps with the goal of showing the focus of attention in a learned deep neural network. Anderson et al. [2] combined saliency maps with reward decomposition and conducted a user study to verify human users' understanding of the given information by checking a user's ability to predict next outcomes. More detailed classifications of the saliency map method were provided by Atrey et al. [3]. In the following chapter, we show the saliency maps on the modified Atari game Amidar based on Greydanus et al. [26]'s method.

Mnih et al. [61] visualized the distribution of the internal representation values with respect to the distances between input vectors using the t-SNE method. The AlphaStar team at Deepmind [95] used the transformer network [94] to beat human experts in Starcraft and the method was able to visualize which location of the game board the trained agent is focusing on and which actions it is currently considering.

However, this visualization can be viewed as an output-based visualization of high-level concepts of the game instead of an explanation per se.

The value decomposition method that will be introduced in Chapter 4 lies between Milani et al.'s LPM and PL categories. It uses MDP properties such as the transition function T and reward function R, giving it elements in common with the LPM category. But it also reasons at a policy level going beyond single steps, like other examples of the PL category. However, unlike other methods introduced in Milani et al.'s survey [58], the method introduced in this dissertation considers policy choices and trajectories of an agent during its simulation or testing phase rather than during its training phase. This property enables a user to consider what an agent would do when the real situation occurs. On the other hand, the existing method considers which training point or trajectory contributes to specific decisions.

LPM methods learns an approximate model of a transition function $\hat{T}$ and try to produce explanations based on the extracted data from $\hat{T}$. This data can be task success rates [12, 13] or intended outcome [105]. Madumal et al. [57] build a structural causal model to understand the relationship between actions and state features. Our method learns occupancy frequencies to also understand the intended outcome and success rate of given tasks as well. Different rewards often means different goals or outcomes that agents pursue or avoid. Decomposition of rewards into meaningful contexts can lead to interesting explanations and understanding of how such rewards influence agents' decisions [2]. Khan et al. [44] and Anderson et al. [2] factor rewards

into the set of reward types $R(s,a) = \sum_{c \in C} R_c(s,a)$, which are also used in our methods. The methods of Dao et al. [15] and Gottesman et al. [25] try to find most influential training points for their agents' decision-making, which is related to the selection of high impact decisions in our method.

Policy level (PL) explanations seek to provide a summarization of transitions [1, 39, 47] and convert their inner recurrent neural network's (RNN's) information into an interpretable format [46, 14, 30]. Other approaches either extract clusters [85] or use abstract states [90]. The abstraction of states can be directly used alongside our value decomposition methods. An interesting direction for future work would be seeking different methods of interpretable state-abstraction methods.

2.2.3 Causality

The idea of causality is central to how people create and understand explanations. Let's take a moment to examine how causality is viewed in computational terms to provide the basis for later discussion of explanations.

Based on the regularity theory of Hume [40], if one type of event always occurs before the other, there is a causal relationship between those two types of events. However, like the fact that a rooster crows before sunrise does not indicate a rooster is the cause of a sunrise, a mere association between two events is not sufficient to claim that one event is the cause of the other event. To handle this situation, Pearl and Mackenzie [69] suggests that causal reasoning should be performed on at least three

levels: association, intervention, and counterfactuals. Pearl calls this set of levels the *Ladder of Causation*.

The first rung of the ladder is *association*. This level of reasoning can be understood by *seeing* and *observing*. Most current statistics and supervised learning techniques work at this level. The reasoning here can answer questions: "How are two variables related?" and "How would seeing an event X can change my belief in event Y?"

The second rung of the ladder is *intervention*. Activities relating to this level of reasoning include *doing* and *intervening*. The causal reasoning at this level can answer questions: "What is the difference in the expected outcome when I do A instead of B?" and "What action is required to make Y happen?" Reinforcement learning and learning causal Bayesian models work at this level [5].

The third and topmost rung is *counterfactuals*. *Imagining*, *retrospection*, and *understanding* activities fall into this category. What if and Why questions are examples of counterfactual questions. Counterfactuals are the hypothetical events that do not cause an event that is to be explained [59].

Halpern and Pearl [29] formally define an *actual cause* of an event $X = x$ as a set of events E (an individual variable is expressed as a form of $Y = y$) if the following criteria hold:

1. In a real situation, both X and E have to be true.

2. If E had some *counterfactual* value, then the event $X = x$ would not have been true.

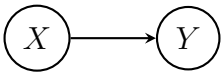


Figure 2.1: A causal diagram showing X causes Y.

3. E has to be minimal. That is E should not contain any irrelevant variables.

Causal relationships can be represented as a graph, with each node representing a variable and a directed edge from node X pointing to node Y meaning X causes Y as in Fig. 2.1.

In Rubin [73]'s definition, a potential outcome of a variable Y when X is assigned with value x is $Y_{X=x}$. The probability of a potential outcome of Y holding a specific value y when X holds value x is defined as $P(Y_{X=x} = y)$. Using this notation, both the probability of necessity and sufficiency can be defined [69]:

- Probability of Necessity (PN): $P(Y_{X=0} = 0|X = 1, Y = 1)$.
- Probability of Sufficiency (PS): $P(Y_{X=1} = 1|X = 0, Y = 0)$.

When there is more than one cause for a single outcome, comparison of these two values for two different causes can give us insights into which cause is more important than the other. The probability of necessity tells us that, when the cause is not present, how likely the expected event is to not occur given that the cause led to the desired outcome in reality. The probability of sufficiency tells us, when the cause is present, how likely the outcome is to occur given that the lack of cause led to no desired outcome. Pearl and Mackenzie [69] states that this mechanism can play an important role when determining the most probable cause in autonomous systems.

Lastly, Pearl and Mackenzie [69] introduce *do*-operator to properly acknowledge intervention in probabilities. While $P(Y|X)$ only captures mere association between two events, $P(Y \mid do(X))$ represents an intervention on X to influence Y and also inhibition of all other effects directing to X. I make use of precisely this setup in the method introduced in this dissertation.

2.2.4 Explanations

Pearl and Mackenzie [69] claim that causation plays an essential role in human explanation. Every time we interact with our world, we ask questions such as why events happen in particular ways, why objects have certain properties, and why people behave in such a way [54]. Lewis [51] defines explanation as a way to provide information about an event's causal history.

According to Gilpin et al. [24], a good explanation can be dependent on the question and pays particular interest to two types of why-questions: why and why-should. He also claims that a good explanation in general comes from good inference, but it can also come from the use of abductive reasoning: finding all possible causes of an effect and finding the best one. Pearl and Mackenzie [69]'s Ladder of Causation can be leveraged to answer questions like What happened, How it happened, and Why it happened, at each respective level.

Lombrozo [54] notes that explanation is a product and a process at the same time. Gilpin et al. [24] argue that there are two processes and a product in explaining: a

cognitive process, a product, and a social process. If explanation is a product, then explaining has to go through both a cognitive and social processes. In this dissertation, I focus on the cognitive process and the product first when generating explanations and then, in future work, seek to consider explanations as a tool for social communication.

Chapter 3

Difficulty in Explaining a Deep RL Agent

Portions of this chapter appeared in the earlier paper, "Measuring and Characterizing Generalization in Deep Reinforcement Learning" [102, 103] with Sam Witty, Emma Tosch, Akanksha Atrey, Michael Littman, and David Jensen.

3.1 The Role of Generalization in Explaining

As discussed in the background chapter, a key capability in explaining a learned agent is being able to reason about counterfactual situations and their outcomes. In this chapter, I describe joint work between me and several colleagues in developing a method that perturbs input states in a semantically meaningful way to create

counterfactual situations. Although the true intention of this setup is to see how an agent behaves in counterfactual situations to better explain the rationale behind its actions, the performance of an agent after perturbation is far inferior to that of non-perturbed situations. I claim that this phenomena is mainly due to the lack of generalization in current Deep RL training method.

To assess generalization in an RL context, we proposed a method to categorize a state set of an MDP given by its domain and present performance measures based on this categorization. In this chapter, we examine both the GridWorld and the Atari game Amidar. To generate contextually perturbed states for testing, we provided a modified Atari environment called Intervenidar modeled after the original Amidar game. This environment enabled us to examine estimated value differences and expected rewards in various possible counterfactual states. It let us answer questions like "What would an agent do if an enemy did not exist in this location?" However, in our test results, we show that the trained policies make poor decisions in such unfamiliar states. As a consequence, it makes it difficult to generate insightful explanations as the agent has not explored enough to correctly predict the counterfactual outcome. Therefore, we question the extent to which learned deep Q networks actually construct generalized representations and show counterfactually reason within a given policy.

Prior Work on Generalization in RL.

Generalization has been a long discussed topic in reinforcement learning [87]. Kakade [43] provided a theoretical framework for bounding the amount of training

data needed for a discrete state and action RL agent to achieve near optimal reward. Nouri et al. [67] and Zhang et al. [106] discuss how to apply the idea of a training/testing split from supervised learning in the context of offline policy evaluation with batch data in RL. Whiteson et al. [101] and Zhang et al. [106] claims that to avoid overfitting it is necessary to diversify the environments that agents are trained on. In other cases, researchers tried to introduce novel states to agents by adding stochasticity to the policy [31], having the agent take random steps, no-ops, steps from human play [65], or probabilistically repeating the agent's previous action [56]. These existing methods provide diverse training data to avoid overfitting and enhance generalization of agents, but none tries to introduce the states that are unreachable but also contextually valid states. We claim such states provide better evidence for high-level generalization.

3.2 State Categorization and Performance Measures

Traditionally, generalization is tested by leaving out the part of training examples as a test set. However, due to the nature of an RL agent's training procedure, distinctions between a training and test set pair can be unclear. This issue is exacerbated in the Arcade Learning Environment (ALE) [6] since most of its games only allow small variations in the beginning of each game by starting at a different time step. Moreover, in real world scenarios, there will always be a case where an agent encounter unseen states no matter how well training and test sets are designed for the agents. In the following, we define *reachable*, *unreachable*, *on-policy*, and *off-policy* states to better

partition the set of states of an MDP in a way that supports assessing generalization for RL agents.

Based on the reachability of a state, we define two categories of a state space S:

Definition 3.2.1 (Reachable State)**.** *Given an MDP $M = \langle S, A, T, R, \gamma \rangle$ with a set of start states S_0, S_t is defined as below.*

For $t > 0$,

$$S_t = \{s \in S \,|\, T(s_p, a, s) > 0, \forall s_p \in S_{t-1} \wedge \forall a \in A\}.$$

A state s is reachable *if $s \in S_t$ for any $t \geq 0$.*

A set of reachable states *of an MDP M is defined as*

$$S_{reachable} = S_0 \cup S_1 \cup \cdots \cup S_\infty.$$

Definition 3.2.2 (Unreachable State)**.** *A* set of unreachable states *of an MDP M is defined as*

$$S_{unreachable} = S \setminus S_{reachable}.$$

A state s is unreachable *if $s \in S_{unreachable}$.*

The above notions can be extended to define on-policy and off-policy states for a given policy as below.

Definition 3.2.3 (On-policy State)**.** *Given an MDP $M = \langle S, A, T, R, \gamma \rangle$ with a set of start state S_0, and a deterministic policy π, S_t^π is defined as below.*

For $t = 0$,

$$S_0^\pi = S_0.$$

For $t > 0$,

$$S_t^\pi = \{s \in S \mid T(s_p, \pi(s_p), s) > 0, \forall s_p \in S_{t-1}^\pi\}.$$

A state s *is* on-policy *if* $s \in S_t^\pi$ *for any* $t \geq 0$.

A set of on-policy states *of an MDP* M *is defined as*

$$S_{on}^\pi = S_0^\pi \cup S_1^\pi \cup \cdots \cup S_\infty^\pi.$$

Definition 3.2.4 (Off-policy State)**.** *A* set of off-policy states *of an MDP* M *is defined as*

$$S_{off}^\pi = S_{reachable} \setminus S_{on}^\pi.$$

A state s *is* off-policy *if* $s \in S_{off}^\pi$.

We look at two performance measures. First, we compare value-estimate errors (VEEs). We compare the estimated value of each state from the learned model to the expected discounted return from the rollouts taken after that state:

$$\mathrm{VEE}_\pi(s) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{k=1}^{\infty} R(s_{t+k}) \mid s_t = s, a_t = \pi(s)\right] - \hat{V}^\pi(s).$$

This VEE reflects the extent to which the estimated value correctly predicts the future outcome.

The second value we examine is an expected discounted return (EDR) from a start

state.

$$\mathrm{EDR}_\pi(s_0) = \mathop{\mathbb{E}}_{\pi}\left[\sum_{k=1}^{\infty} R(s_k) \,|\, s_0 = s, a_0 = \pi(s)\right].$$

It is essentially the (undiscounted) value of the start state s_0, $V^\pi(s_0)$. We use the estimation of this quantity from the finite number of rollouts, $\tilde{V}^\pi(s_0)$.

3.3 Case Study: GridWorld

We first demonstrate an example of these four categories in the GridWorld domain in Fig. 3.1. The GridWorld used in the chapter consists of 25×16 grid and in the middle and it has a wall at $x = 13$ that spans from $y = 1$ to $y = 12$. In this environment, an agent begins each episode at a start state s_0 and can take actions *right*, *right up*, *right down*. The agent obtains a terminal reward of $+1$ when it reaches the goal state s_g and an intermediate reward of $+0.1$ when it reaches a state where $x = 2 + 3i$ and $y = 2 + 3j$ for all $i, j \geq 0$. Since it always advances to the right, there are three regions that are unreachable from the agent's start state: the upper left corner, the lower left corner, and the lower left corner after the wall. Note that the agent would be able to reach the goal from unreachable states in the upper left corner.

In our first case study, we ran tabular Q-learning on the GridWorld domain and its progression during learning is depicted in Fig. 3.3. The optimal value function V^* calculated via value iteration is shown in Fig. 3.2.

As depicted in Fig. 3.3, Q-learning gradually makes the value of each state closer to its optimal value. However, it usually takes very long time to converge to an optimal

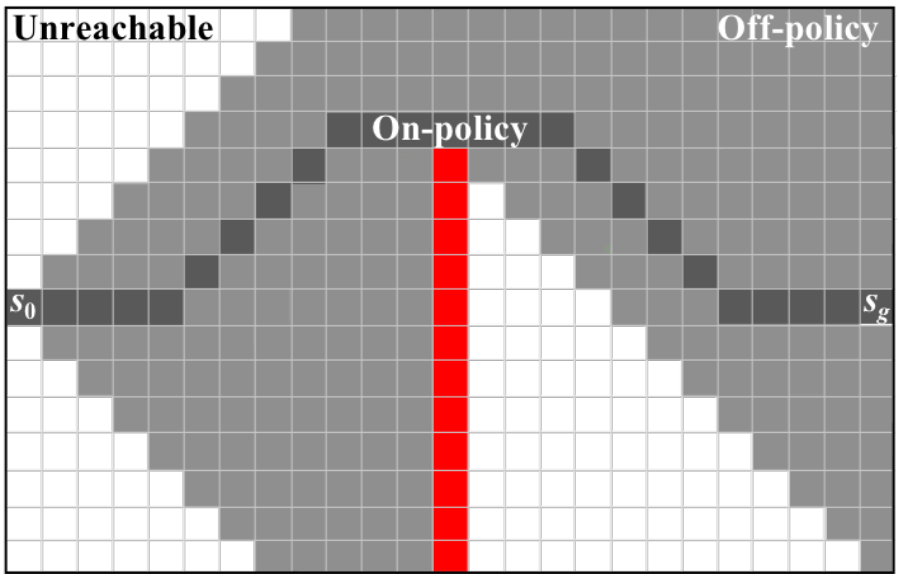


Figure 3.1: Examples of on-policy, off-policy, and unreachable states in GridWorld.

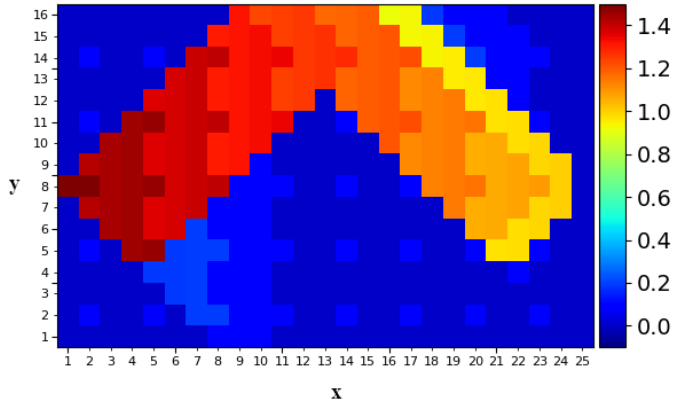


Figure 3.2: Optimal value function for the GridWorld example.

policy and, if the state space is large, it is completely impractical to identify the optimal policy from all states. Although the learned policy π is not optimal for all the states (see Fig. 3.4), it still learned the optimal path from the start state and thus fulfilled its main mission. Therefore, if you only consider the performance of an agent, this result appears adequate and measuring the performance of the learned

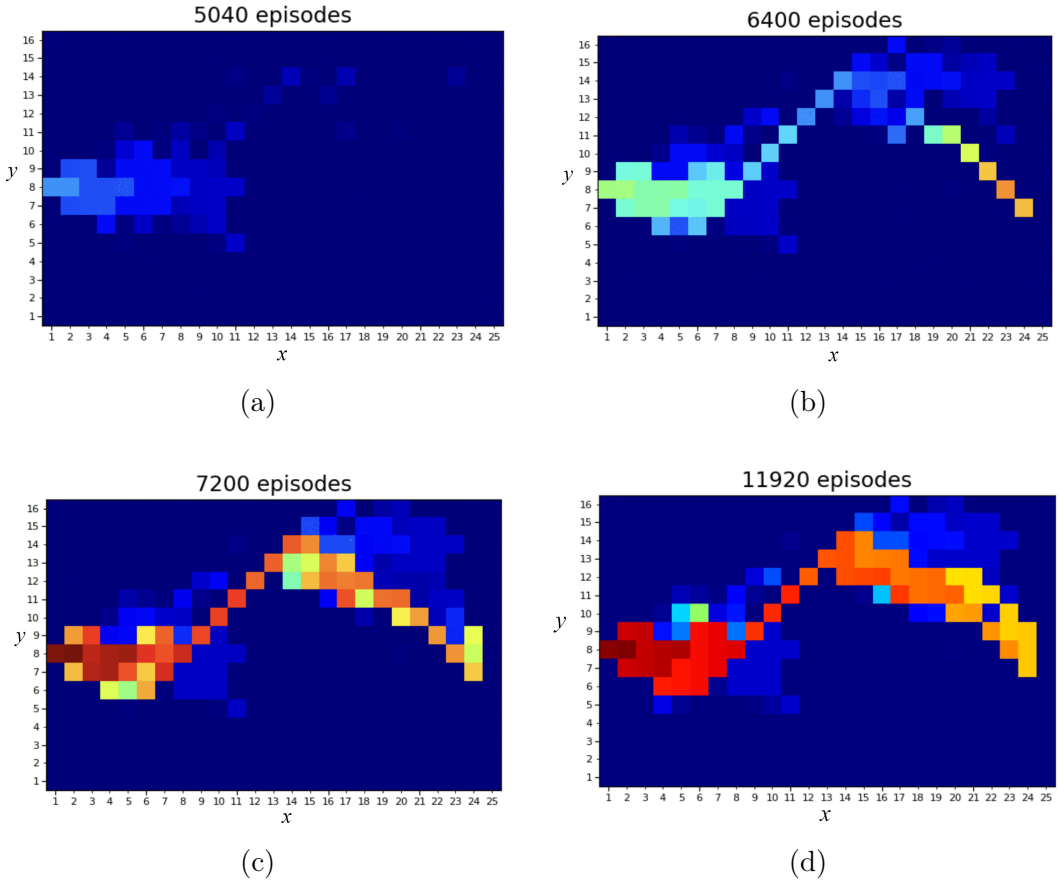


Figure 3.3: The progression of Q-learning in GridWorld (a) After 5040 episodes. (c) After 6400 episodes. (c) After 7200 episodes. (d) After 11920 episodes.

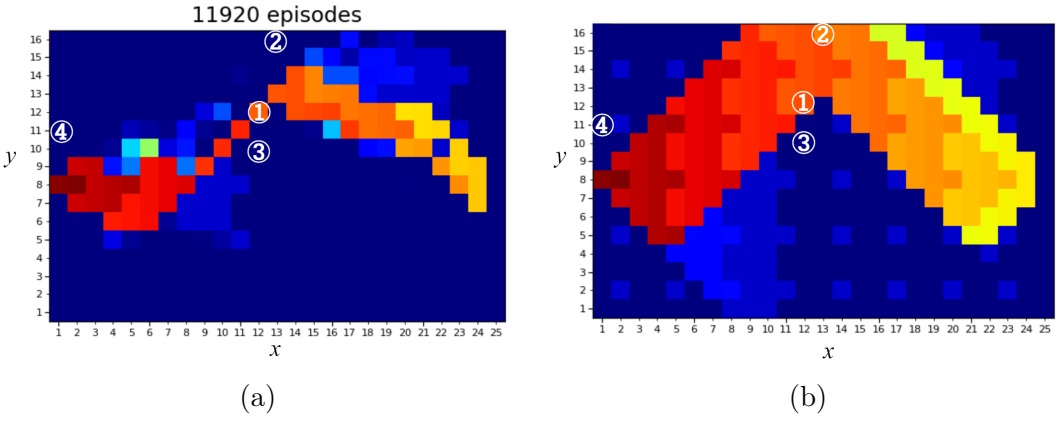


Figure 3.4: The Explanation of four states in the GridWorld (a) Values π learned by the tabular Q-learning method (b) Optimal values from the value iteration.

policy produces the same conclusion. However, the algorithm's behavior is a problem when counterfactual comparisons are made. Consider comparing the learned policy to the optimal policy for this grid at the four different states highlighted as from 1 through 4 in white circles in Fig. 3.4, and the result of attempting to explain the behavior of the agent in these states. In the case of the State 1, which is an on-policy state, both the learned and optimal policies output the same result. In contrast, if you use the learned policy, both State 2 and State 3, which are off-policy, will not reach the goal state (see Fig. 3.4 (a)). Using the optimal policy, only State 2 will reach the goal and State 3 will not (See Fig. 3.4 (b)). States 2 and 3 are semantically different, even though the learned policy treats them as highly similar. The difference can later be used to explain the effect of wall in the agent's decisions. However, the explanations that can be distilled from both policies differ drastically based on the learning progress. It is therefore necessary to capture this phenomenon in the testing phase, but, even there, it is hard to capture if we only use the performance measures on on-policy states or trajectories.

For unreachable states like State 4, the predicted values are the same for both policies. However, if we train on a broader set of start states that includes State 4, the results will be different and the explanations will also disagree. In real world scenarios, it is impossible to include all possible start states and the problem of this discrepancy can be an issue.

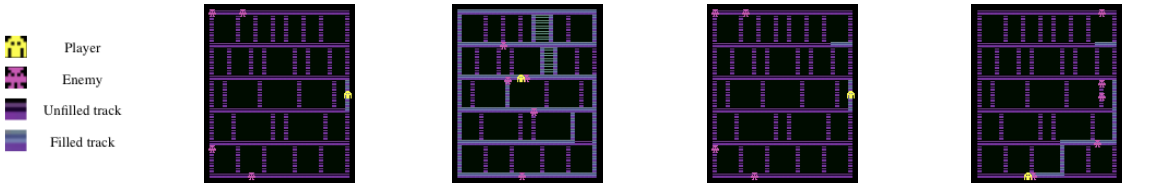
3.4 Case Study: Amidar and Intervenidar

Amidar (see Fig. 3.5) is a Pac-Man-like video game in which an agent moves a player around a two-dimensional maze, accumulating reward for each vertical and horizontal hallway segment visited. An episode terminates when the player makes contact with one of the five enemies that also move around in the grid. After three episodes, the game is over.

We demonstrate the state categorization described previously on Amidar, which has been used as a benchmark task for deep RL. We trained a suite of agents and evaluated them on a series of on-policy, off-policy, and unreachable Intervenidar states. Using our proposed partitioning of states and empirical methodology, we ran a series of experiments on these agents' performance to assess how well they generalize.

We used the standard Amidar MDP specification for state: a three-dimensional tensor composed of greyscale pixel values for the current, and three previous, frames during gameplay [61]. There are five movement actions. The transition function is deterministic. The reward function is the difference between successive scores, and is truncated such that positive differences in score result in a reward of 1. There are no negative rewards, and state transitions with no change in score result in a reward of 0.

We trained all agents using the state-of-the-art dueling network architecture, double Q-loss function, and prioritized experience replay [93, 98, 77]. All of the training sessions in this dissertation used the same hyperparameters as in Mnih et al.'s work [61] and we use the OpenAI's baseline implementation [16].



(a) Default start (b) Default death (c) Modified start (d) Modified death

Figure 3.5: Minor changes in Amidar game state can dramatically reduce a trained agent's reward.

3.4.1 Methodology

In this section, we describe specific techniques for producing off-policy states and a general methodology for producing unreachable states based on parameterized simulators and controlled experiments.

Off-Policy States

Stochasticity. One method for producing off-policy states is to introduce stochasticity into the agent's policy [56]. We present a representative method we call k *off-policy actions* (k-OPA), which causes the agent to execute some sequence of on-policy actions and then take k random actions to place the agent in an off-policy state. This method is scalable to large and complex environments, but careful consideration must be made to avoid overlap between states, as well as to ensure that the episode does not terminate before k actions are completed. It is easy to imagine other variations, where the k actions are not selected randomly but according to some other mechanism inconsistent with greedy-action selection.

Human Agents. The use of human agents has become a standard method in

evaluating the generalization capabilities of RL agents. The most common method is known as human starts (HS) and is defined as exposing the agent to a state recorded by a human user interacting with an interface to the MDP environment [61]. One could easily imagine desirable variations on human starts within this general category, such as passing control back and forth between an agent and a human user. Human agents differ from other alternative agents in that they may not be motivated by the explicit reward function specified in the MDP, instead focusing on novelty or entertainment.

Synthetic Agents. We present a state-sampling method method we call *agent swaps* (AS), where the agent is exposed to a state midway through an alternative agent's trajectory. This method has the potential to be significantly more scalable than human starts in large and complex environments since it can work using agents trained with different random seeds.

Unreachable States

Unreachable states are unlike off-policy states, which can be produced using carefully selected action sequences. By definition, unreachable states require some modification to the environment such as removing or adding game entities.

Intervening on Latent State. We present two distinct classes of interventions on latent state: existential, adding or removing entities, and parameterized, varying the value of an input parameter for an entity.

To facilitate this kind of intervention on latent state, our colleagues implemented Intervenidar, an Amidar emulator. Intervenidar closely mimics the behavior of the Atari 2600 Amidar game, while allowing scientists to modify board configurations, sprite positions, enemy movement behavior, and other features of gameplay. Some manipulable features that we use in our experiments are:

Enemy existence and movement. The five enemies in Amidar move at a constant speed along a fixed track. By default, Intervenidar also has five enemies whose movement behavior is a time-based lookup table that mimics enemy position and speed in Amidar. Other distinct enemy movement behaviors include following the perimeter.

Line segment existence and predicates. A line segment is any hallway in the maze that intersects with another hallway at both endpoints. Line segments may be filled or unfilled; the player's objective is to fill all of them to advance to the next board. In Intervenidar, scientists may specify which of the 88 line segments are filled at any timestep. Furthermore, Intervenidar allows scientists to customize the quantity and position of line segments.

Player/enemy positions. Player and enemy entities always begin a game in the same start positions during Amidar, but they may be moved to arbitrary locations at any point in Intervenidar.

We included these features in the experiments because they cover what we believe to be the fundamental components of Amidar gameplay, avoiding death and navigating the

board to accumulate reward. The scale of these interventions were selected to constitute small changes from the original environment and hence minimal counterfactuals.

3.4.2 Experiment Setup

We trained a suite of agents and evaluated them on a series of on-policy, off-policy, and unreachable Intervenidar states. In this section, we discuss how we generated off-policy and unreachable states for the Amidar problem domain.

Amidar Agents. We explored three types of modifications on network architecture and training regimens in an attempt to produce agents that generalized well: (1) increasing dataset size by increasing training time; (2) broadening the support of the training data by increasing exploration at the start of each episode; and (3) reducing model capacity by decreasing network size and number of layers. To establish performance benchmarks for unreachable states, we trained an agent on each of the experimental unreachable configurations.

Training Time. To understand the effect of training-set size on generalization performance, we saved checkpoints of the parameters for the baseline DQN after 10, 20, 30, and 40 million training actions before the model's training reward converged at approximately 50 million actions. This process differs from increasing training dataset size in prediction tasks in that increasing the number of training episodes simulataneously changes the distribution of states in the agent's experience replay.

Exploring Starts. To increase the diversity of the agent's experience, we trained

agents with 30 and 50 random actions at the beginning of each training episode before returning to the agent's standard ϵ-greedy exploration strategy.

Model Capacity. To reduce the capacity of the Q-value function, we explored three architectural variations from the state-of-the-art dueling architecture: (1) reducing the size of the fully connected layers by half (256-HU), (2) reducing the number of channels in each of the three convolutional filters by half (HC), and (3) removing the last convolutional layer of the network (TL).

Off-policy States. We employed three strategies to generate off-policy states for an agent: human starts, agent swaps, and k-OPA. None of these methods require the Intervenidar system. In each case, we ran an agent nine times, for n steps, where $n \in \{100, 200, \ldots, 900\}$.

Human starts. Four individuals played 30 Intervenidar games each. We randomly selected 75 action sequences lasting more than 1000 steps and extracted 9 states, taken at each of the n time steps [65].

Agent swaps. We designated five of the trained agents as *alternative agents*: (1) the baseline agent, (2) the agent that starts with 50 random actions, (3) the agent with half of the convolutional channels as the original architecture, (4) the agent with only two convolutional layers, and (5) the agent with 256 hidden units. We chose these agents with the belief that their policies would be sufficienctly distinct from each other to provide some variation in off-policy states.[1]

[1]When evaluating any of the alternative agents, we only used states from the remaining four to generate off-policy states.

k-OPA. Unlike the previous two cases where states came from sources external to the agent, in this case we had every agent play the game for n steps before taking k random actions, where k was set to 10 and 20.

Unreachable States. With Intervenidar, we generated unreachable states, guaranteeing that the agent begins an episode in a state it has never encountered during training. All modifications to the board happen before the agent begins acting.

Modifications to enemies. We make one existential and one parameterized modification to enemies: We randomly remove between one and four enemies from the board (ER), and we shift one randomly selected enemy by n steps along its path, where n is drawn randomly between 1 and 20 (ES).

Modifications to line segments. We make one existential and one parameterized modification to line segments: We add one new vertical line segment to a random location on the board (ALS) and we randomly fill between one and four non-adjacent unfilled line segments (FLS).

Modification to player start position. We start the player in a randomly chosen unoccupied tile location that is at least one tile away from any enemies (PRS).

3.4.3 Results

Our experiments demonstrate that: (1) the state-of-the-art DQN has poor generalization performance for Amidar gameplay; (2) distance in the network's learned

representation is strongly anti-correlated with generalization performance; (3) modifications to training volume, model capacity, and exploration have minor and sometimes counterintuitive effects on generalization performance.

Poor Generalization Performance. Fig. 3.6 show that the fully trained state-of-the-art DQN dueling architecture produces a policy that is exceptionally sensitive to small contextual to environmental variations. The worst examples can be seen in Fig. 3.6, in the filling line segments (FLS) and player random starts (PRS) interventions. Examination of the action sequences preceding these states showed the agent predominantly remaining stationary, often terminating epsisodes without completing a single line segment.

Furthermore, Fig. 3.6 shows that VEE and EDR are very highly anti-correlated across the experiments, indicating that the agent's ability to select appropriate actions is related to its ability to correctly measure the value of a particular state. We observe that the model always overestimates the value of off-policy and unreachable states. In contrast, the agent's value estimates are much more accurate in the control condition.

Distance in Representation. By extracting the activations of the last layer of the DQN, we are able to observe the distance between training and evaluation states in terms of the network's learned representation. Fig. 3.7 depicts the the distribution of these distances. We find that the agent does not "recognize" the unreachable states where generalization is the worst, such as PRS and FLS, in the sense that the internal representations are quite far from what is observed during its

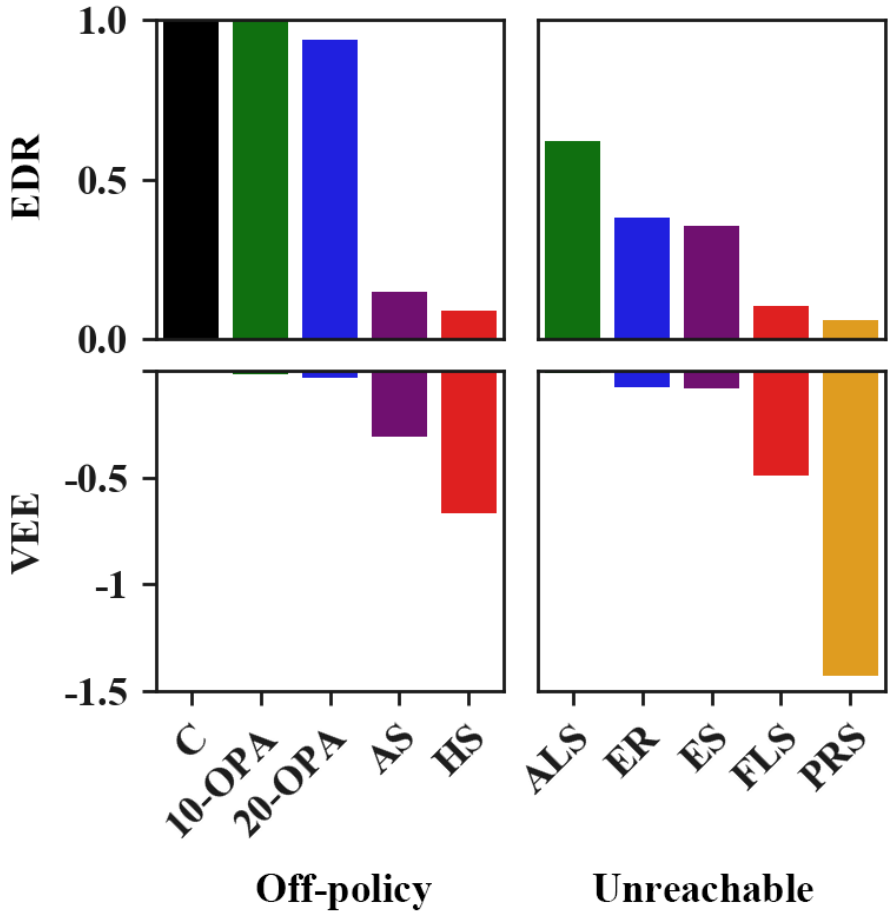


Figure 3.6: EDR and average VEE for control, off-policy, and unreachable experiments. The agent consistently overestimates the state value. EDR and VEE are strongly anti-correlated. All EDR bars are normalized by the EDR of the control condition. All VEE bars are normalized by their corresponding EDR.

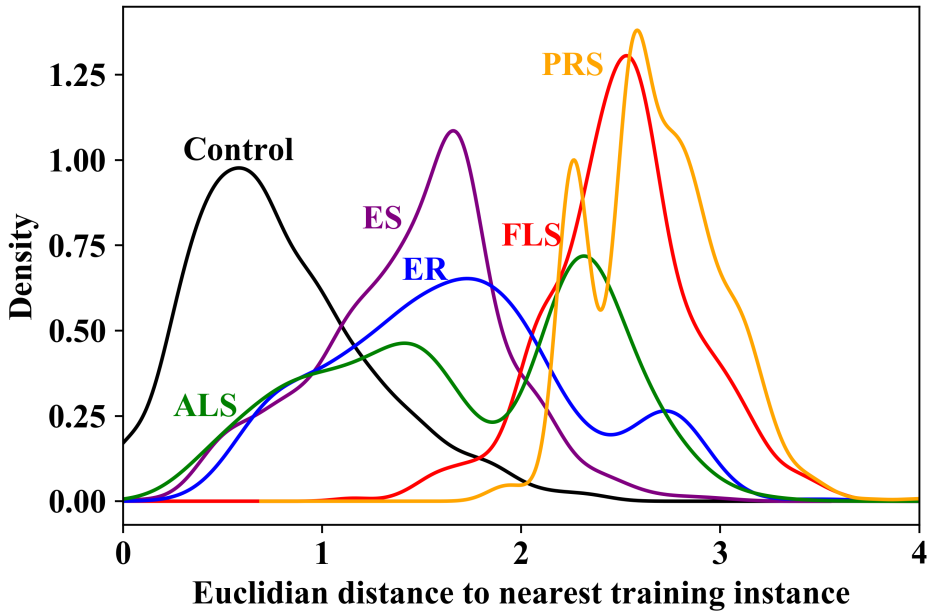


Figure 3.7: Smoothed empirical distributions of the distances between the test points of the unreachable experiments and the training data. More distant states are handled less well by the agents, suggesting they are unable to generalize counterfactually.

standard behavior. Alternatively, one could imagine a network that performs poorly by conflating states that are meaningfully different. Using the activations of the last layers, both the relative distances between each state's internal representation and its Q-value can be depicted in a two dimensional graph using t-SNE [55] as in Fig. 3.8. The perturbed states, such as FLS, ER, ALS, stayed close together in terms of their internal representations. However, each state's temporal correlation seems to play a more important role in combining each state's internal representation. That implies that the network is correctly keeping different states different, but is struggling to relate novel states to those it is more familiar with.

Training Agents for Generalization. We take inspiration from well-established methods in supervised learning; increasing training set size, broadening the support

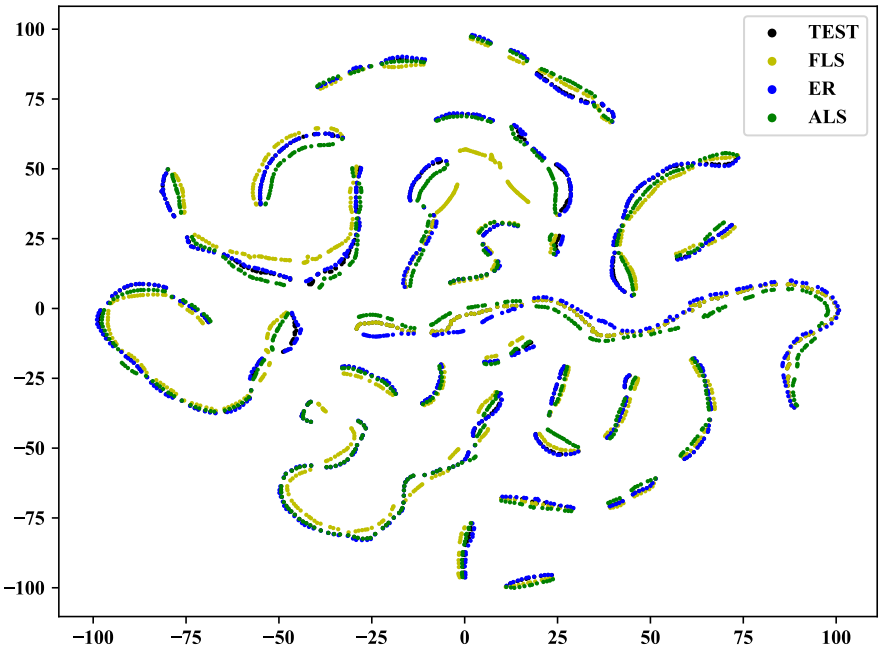


Figure 3.8: t-SNE plot of a single run of a trained agent. The plot shows the relationship between the relative distances of each input state's internal representation and each state's Q-value. Although perturbed states stay close together in terms of their internal representations, each state's temporal correlation seems to play a dominant role is determining the representation.

of the training distribution, and reducing model capacity. We propose the following analogs to each of these methods, respectively; increasing the number of training episodes, introducing additional exploration, and removing layers and nodes.

Our experiments indicate that: (1) naïvely increasing the number of training episodes until training set performance converges does not help generalization; (2) some reductions to model capacity induce improvements to generalization; and (3) increasing exploration and otherwise diversifying training experience results in more generalized policies. These results are shown in Fig. 3.9.

Training Episodes. While increasing training time clearly increases the expected

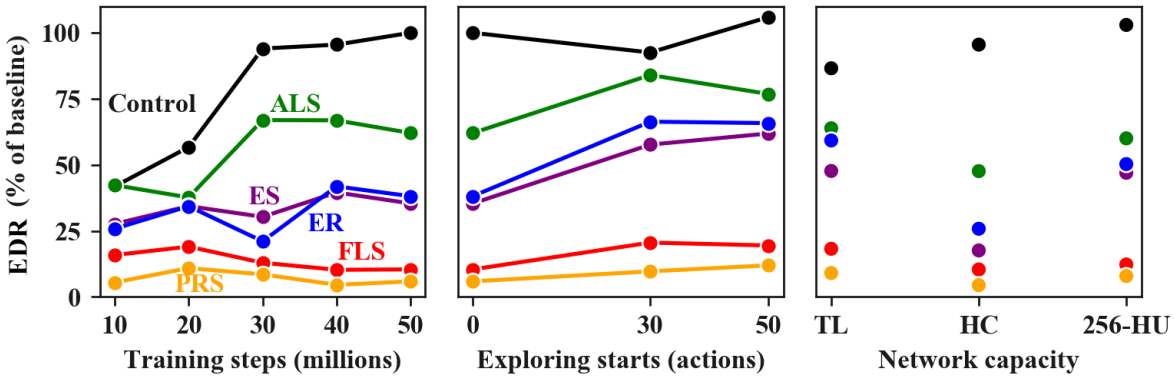


Figure 3.9: Average expected discounted returns (EDR) from various unreachable states for each of the trained agents.

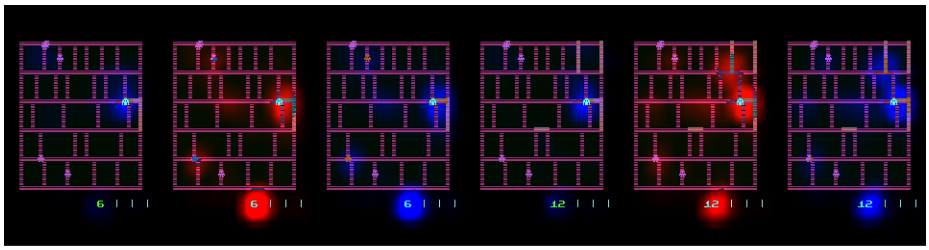
discounted return in the control condition, shorter training times appear to contribute to increased generalization ability. This increase is minimal, but it does illustrate that naïvely increasing training time until converge of training rewards may not be the best strategy for producing generalized agents. The mechanism for this finding is akin to overfitting as the network deploys more and more of its capacity to on-policy states.

Model Capacity. Of the reductions to model capacity, we find that shrinking the size of the fully-connected layers results in the greatest increase in generalization performance across perturbations. Reducing the number of convolutional layers also results in improvements in generalization performance, particularly for the enemy perturbation experiments.

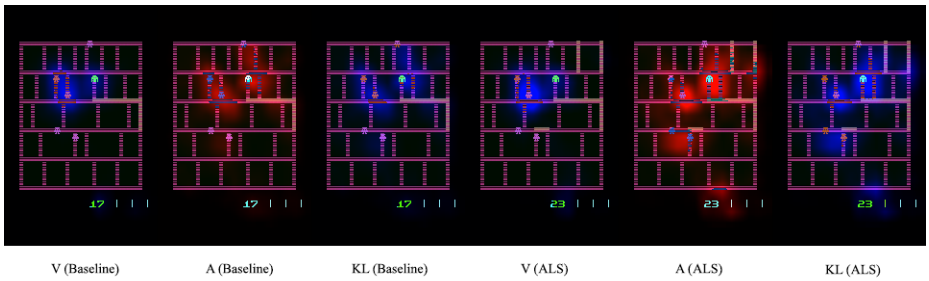
Exploration Starts. We find that increasing the diversity of training experience has the greatest effect on generalization performance, particularly for the agent with 50 random actions. This agent experiences almost a factor of two increase in expected discounted return for human starts and all of the unreachable-state experiments. This

agent outperforms the baseline agent in every condition. Of particular interest is the agent's performance on the enemy shift experiments, where the agents' expected discounted return approaches the reward achieved by an agent trained entirely in that scenario. While the agents with increased exploration demonstrate a clear improvement in generalization ability over the baseline, it is not consistent with their ability to achieve high reward with the alternative enemy-movement protocol after retraining.

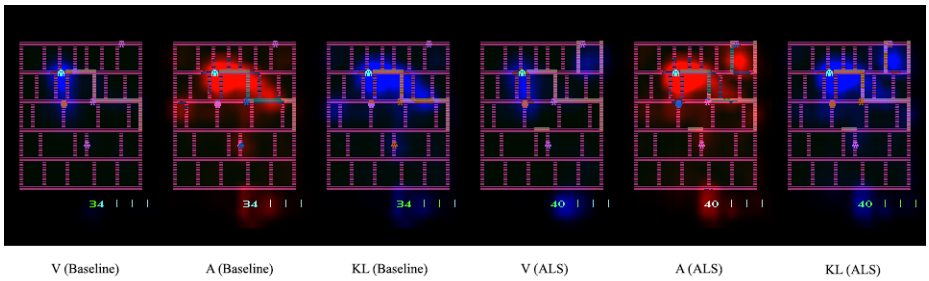
Saliency Maps. Using Greydanus et al. [26]'s method, the value function (V), advantage function (A) and KL divergence (KL) values are depicted in heatmaps. Baseline and ALS states are compared in Fig. 3.10. Note that the agent itself has high saliency, which is appropriate. However, in many cases the score appears more salient than the enemies, which is a clear sign of memorization.



(a) At frame 30.



(b) At frame 86.



(c) At frame 127.

Figure 3.10: Saliency maps in Amidar. The value function (V), advantage function (A) and KL divergence (KL) values are depicted in heatmaps. Baseline and ALS states are compared.

Chapter 4

Policy Comparison using Value Decomposition

In this chapter, I propose a novel method for decomposing the agent's value into the accumulated reward obtained before vs. after visiting a state and use this metric to select an important decision state where switching to a different policy would result in the most impact in the overall value of starting states (see Fig. 4.1). Based on this analysis, we can generate an explanation that makes use of the identity of the most influential state.

4.1 Background

This section provides some background information that my approach builds on.

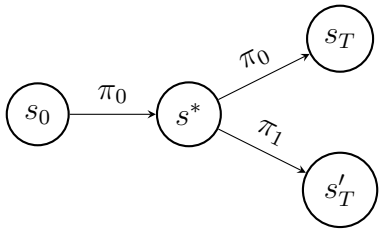


Figure 4.1: A diagram showing an agent's decision change at the intermediate state s^* starting from s_0.

4.1.1 Occupancy Frequency

Khan et al. [44] first developed a method to explain an RL agent in a MDP using the occupancy frequency concept. Their method distinguishes the most influential sets of states where each set has the same reward value. Templates for explanations can then be generated including an occupancy frequency for that set of states and factors of states that are generalized from the set of states.

Our approach is built upon this framework and adds an algorithm to compare two different policies and find a state such that switching to a different contrasting policy from this state most impacts the overall utility of an agent, $\mathbb{E}_{s_0 \in S_0}[V(s_0)]$.

Dodson et al. [17] also extends the work of Khan et al. [44] in the area of academic advising by making use of an MDP and the occupancy frequency concept to generate template-based natural language explanations.

4.1.2 Counterfactual Reasoning

According to Pearl and Mackenzie [69], causality plays an essential role in human explanation. In this dissertation, we situate an agent in a counterfactual situation

where it follows a policy π_0 until it reaches a state s^* and then we run a test to either switch to a different policy π_1 or to stay with the original policy. This switch allows us to compare two different policies, highlighting which state most affects the value after this policy change.

In term of comparing two different policies and finding an important decision point, Gottesman et al. [25] take a similar approach and introduce an influence function. However, their approach mostly concentrates on removing a single transition instead of the change of an entire policy after visiting a certain state.

4.2 Methods

This section describes our algorithmic approach.

4.2.1 Occupancy Frequency

Definition 4.2.1 (State Occupancy Function)**.** *The state occupancy of a state $s \in S$ starting from s_0 following a policy π is defined by*

$$K^{s_0,\pi}(s') = \sum_{t=0}^{\infty} \gamma^t P(s_{t+1} = s' \mid s_0, \pi).$$

This state occupancy function K can be interpreted as a sum of all probability values at all steps discounted by γ^t as it marches away from the start state s_0. An interesting aspect of this function is that similar to value and Q functions it can also be calculated by using dynamic programming. In other words, it can be computed by

iterating a recurrence relation.

Lemma 4.2.1 (State Occupancy Recurrence Relation)**.** *Let K be a state occupancy function following a policy π starting from a state s_0, then the following holds:*

$$K^{s_0,\pi}(s') = \sum_s T(s' \mid s, \pi) \left(P(s_0 = s \mid s_0) + \gamma K^{s_0,\pi}(s)\right).$$

Proof. We first start with the definition of the state occupancy function

$$K^{s_0,\pi}(s') = \sum_{t=0}^{\infty} \gamma^t P(s_{t+1} = s' \mid s_0, \pi).$$

Conditioning on a previous state using the Markov property,

$$\begin{aligned}
&= \sum_{t=0}^{\infty} \gamma^t \sum_s P(s_{t+1} = s' \mid s_t = s, s_0, \pi) P(s_t = s \mid s_0, \pi) \\
&= \sum_{t=0}^{\infty} \gamma^t \sum_s T(s' \mid s, \pi) P(s_t = s \mid s_0, \pi) \\
&= T(s' \mid s_0, \pi) P(s_0 \mid s_0) + \sum_{t=1}^{\infty} \gamma^t \sum_s T(s' \mid s, \pi) P(s_t = s \mid s_0, \pi).
\end{aligned}$$

By factoring out γ and reparameterizing the first summation,

$$\begin{aligned}
&= T(s'|s, \pi) P(s_0 \mid s_0) + \gamma \sum_{t=1}^{\infty} \gamma^{t-1} \sum_s P(s_t = s|s_0, \pi) T(s'|s, \pi) \\
&= T(s'|s, \pi) P(s_0 \mid s_0) + \gamma \sum_{t=0}^{\infty} \gamma^t \sum_s P(s_{t+1} = s|s_0, \pi) T(s'|s, \pi),
\end{aligned}$$

we can rearrange the summations,

$$= T(s'|s, \pi) P(s_0 \mid s_0) + \sum_s T(s'|s, \pi) \gamma \sum_{t=0}^{\infty} \gamma^t P(s_{t+1} = s|s_0, \pi).$$

Using the definition of K,

$$= T(s'|s,\pi)P(s_0 \,|\, s_0) + \sum_s T(s'|s,\pi)\gamma K^{s_0,\pi}(s)$$

$$= \sum_s T(s'|s,\pi)\left(P(s_0 = s \,|\, s_0) + \gamma K^{s_0,\pi}(s)\right).$$

Although trivially $P(s_0 \,|\, s_0) = 1$, we leave it in the proof to avoid confusion since it gets added again at the last step to the summation. □

Next, we show that this system of equations has a unique solution and that it can be approximated accurately via repeated application of a value-iteration-like operator.

Theorem 4.2.1 (Contraction)**.** *If K is a function mapping states to values, define Z, an operator on such functions, as follows:*

$$ZK(s') = \sum_s T(s'|s,\pi)(P(s_0 = s|s_0) + \gamma K(s).$$

Then, for any occupancy frequency function K_1 or K_2, the operator Z is a contraction mapping in the total variation $\sum_s |ZK_1(s) - ZK_2(s)|$ with the index of contract γ. This implies, by Banach's fixed point theorem, that Z has a unique fixed point and that iterating Z converges to this fixed point at a rate of γ.

Proof. We only need to prove the Z is a contraction mapping and the rest follows

naturally:

$$\begin{aligned}\sum_{s'} |ZK_1(s') - ZK_2(s')| =& \sum_{s'} |(\sum_{s} T(s'|s,\pi)(P(s_0 = s|s_0) + \gamma K_1(s)) - \\ & (\sum_{s} T(s'|s,\pi)(P(s_0 = s|s_0) + \gamma K_2(s))| \\ \leq& \sum_{s'}\sum_{s} T(s'|s,\pi)\gamma|K_1(s) - K_2(s)| \\ =& \gamma \sum_{s} |K_1(s) - K_2(s)| \sum_{s'} T(s'|s,\pi) \\ =& \gamma \sum_{s} |K_1(s) - K_2(s)|.\end{aligned}$$

That is, the new total variation after applying the operator Z to K_1 and K_2 is no more than γ times the total variation between K_1 and K_2. □

We can define the overall occupancy frequency function over a set of start states S_0 by applying the expectation to the previous occupancy frequency function on each single start state $s_0 \in S_0$.

Definition 4.2.2 (Overall Occupancy Frequency)**.** *For any $s \in S$, an* overall occupancy frequency function *of a policy π can be defined as*

$$K^{\pi}(s) = \mathop{\mathbb{E}}_{s' \in S_0}\left[K^{s',\pi}(s)\right].$$

Lemma 4.2.2 (Overall Occupancy Frequency Recurrence Relation)**.** *Let K be an overall state occupancy function following a policy π start from a set of states S_0. Then, the following holds:*

$$K^{\pi}(s') = \sum_{s} T(s' \,|\, s,\pi)\left(P(s_0 = s) + \gamma K^{\pi}(s)\right).$$

Proof.

$$
\begin{aligned}
K^{\pi}(s') &= \mathop{\mathbb{E}}_{s''\in S_0}\left[K^{s'',\pi}(s')\right] \\
&= \sum_{s''\in S_0} P(s_0 = s'')K^{s'',\pi}(s') \\
&= \sum_{s''\in S_0} P(s_0 = s'')\left(\sum_{s} T(s' \mid s,\pi)\left(P(s_0 = s \mid s'',\pi) + \gamma K^{s'',\pi}(s)\right)\right) \\
&= \sum_{s} T(s' \mid s,\pi) \\
&\quad \left(\sum_{s''\in S_0} P(s_0 = s'' \mid \pi)P(s_0 = s \mid s'',\pi) + \gamma \sum_{s''\in S_0} P(s_0 = s'' \mid \pi)K^{s'',\pi}(s)\right) \\
&= \sum_{s} T(s' \mid s,\pi) \\
&\quad \left(\sum_{s''\in S_0} P(s_0 = s'' \mid \pi)P(s_0 = s \mid s'',\pi) + \gamma K^{\pi}(s)\right).
\end{aligned}
$$

The start probability $P(s_0 = s \mid s'', \pi)$ is 1 if $s'' = s$ and 0 otherwise. If $s \notin S_0$ the first term is always 0:

$$
= \sum_{s} T(s' \mid s,\pi)\left(P(s_0 = s \mid \pi) + \gamma K^{\pi}(s)\right).
$$

□

It is trivial to prove that the overall occupancy frequency recurrence relation is also a contraction mapping. Therefore, the function always has a fixed point and converges. The iteration algorithm in Alg. 1, which is similar to the value iteration algorithm, works on the occupancy frequency function both on a single start state s_0 and a set of start states S_0 since its main structure is the same.

One key benefit in using the occupancy frequency function is that the value of

Algorithm 1 Overall Occupancy Frequency Iteration Algorithm

Input: a policy π, transition function T, distribution of start states, $P(s_0 = s)$, $s \in S_0$, and a threshold θ which determines the accuracy of the approximation.
Output: $K \approx K_*$

```
procedure OccupancyFunction(π, T, P(s_0 = s), θ)
    repeat
        Initialize all K(s)s with 0
        Δ ← 0
        for all s ∈ S do
            k ← K(s)
            K(s) ← Σ_s T(s' | s, π) (P(s_0 = s) + γK(s))
            Δ = max(Δ, |k − K(s)|)
        end for
    until Δ < θ
    return K
end procedure
```

a start state can be represented as a product of reward and occupancy frequency functions. For simplicity, we can define the reward function to map a state to a value and as defined as the single value received at the time an agent enters that state. In this dissertation, we use $R(s)$ notation to represent this function compared to $R(s, a, s')$ notation in which a reward function maps state, action, and next action triples to a reward value.

Lemma 4.2.3. *The value of a start state s_0 is a sum of the product of the reward R and occupancy frequency function $K^{s_0,\pi}$ over all states:*

$$V^{\pi}(s_0) = \sum_{s \in S} \left[R(s) \cdot K^{s_0,\pi}(s) \right].$$

An expected value on a set of start states S_0 is also a sum over all states of the product

of a reward function R and an overall occupancy frequency function K^π.

$$\mathop{\mathbb{E}}_{s\in S_0}\left[V^\pi(s)\right] = \sum_{s\in S}\left[R(s)\cdot K^\pi(s)\right].$$

Proof. For the value of a start state s_0,

$$\begin{aligned}
V^\pi(s_0) &= \mathbb{E}\left[\sum_t^\infty \gamma^t r_t \,|\, s_0, \pi\right] \\
&= \sum_t^\infty \gamma^t \,\mathbb{E}\left[r_t \,|\, s_0, \pi\right] \\
&= \sum_t^\infty \gamma^t \sum_s \left(R(s)\cdot P(s_{t+1}=s \,|\, s_0,\pi)\right) \\
&= \sum_s \left(R(s)\cdot \sum_t^\infty \gamma^t P(s_{t+1}=s \,|\, s_0,\pi)\right) \\
&= \sum_s \left(R(s)\cdot K^{s_0,\pi}(s)\right).
\end{aligned}$$

For the overall value of a set of start states S_0, using the above proof,

$$\begin{aligned}
\mathop{\mathbb{E}}_{s\in S_0}\left[V^\pi(s)\right] &= \mathop{\mathbb{E}}_{s\in S_0}\left[\sum_s \left(R(s)\cdot K^{s_0,\pi}(s)\right)\right] \\
&= \sum_{s'\in S_0} P(s_0=s')\left(\sum_s \left(R(s)\cdot K^{s_0,\pi}(s)\right)\right] \\
&= \sum_s \left(R(s)\cdot\left(\sum_{s'\in S_0} P(s_0=s')\cdot K^{s_0,\pi}(s)\right)\right) \\
&= \sum_s \left(R(s)\cdot K^\pi(s)\right).
\end{aligned}$$

□

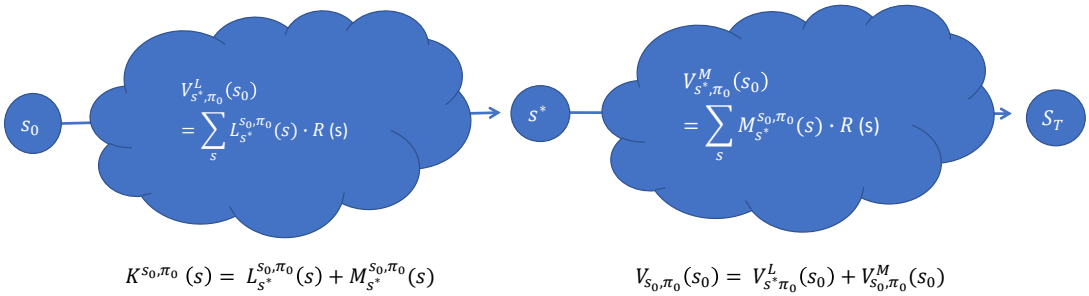


Figure 4.2: The value of a start state s_0 can be decomposed into the anterior and posterior values with respect to a state s^*.

4.2.2 Value Decomposition

We let $\mathcal{D}^\pi$ be the set of all trajectories that can be produced by the policy π. A trajectory τ consists of a series of state, action, and next-state pairs (s, a, s'), $\tau = \{(s_i^\tau, a_i^\tau, s_i'^\tau\}_{i=0}^L$. If, for any trajectory τ and a state s^* , there is a smallest i such that $s_i^\tau = s^*$ then that i is defined as $\mathrm{i}^\tau(s^*)$. If there is no such i, then $\mathrm{i}^\tau(s^*)$ is ∞. If we do not specify τ, it is implied that such τ is sampled by applying the policy π to an environment.

Definition 4.2.3 (Anterior Occupancy Function)**.** *The anterior state occupancy of a state $s \in S$ with respect to an intermediary state s^* is defined by*

$$L^{s_0,\pi}_{s^*}(s) = \sum_{t=0}^{\infty} \gamma^t P_\pi(s_{t+1} = s, t < \mathrm{i}(s^*) \mid s_0).$$

Lemma 4.2.4 (Anterior Occupancy Recurrence Relation)**.** *Let L be an anterior state occupancy function following a policy π with respect to a starting state s_0 and an*

intermediary state s^*, then the following holds

$$L_{s^*}^{s_0,\pi}(s') = \sum_{s \in S \setminus \{s^*\}} T(s' \mid s, \pi) \left(P(s_0 = s \mid s_0) + \gamma L_{s^*}^{s_0,\pi}(s) \right).$$

Proof.

$$\begin{aligned} L_{s^*}^{s_0,\pi}(s') &= \sum_{t=0}^{\infty} \gamma^t P_\pi(s_t = s', t < \mathrm{i}(s^*) \mid s_0) \\ &= \sum_{t=0}^{\infty} \gamma^t \sum_{s \in S \setminus \{s^*\}} P_\pi(s_{t+1} = s' \mid s_t = s, t < \mathrm{i}(s^*), s_0) P_\pi(s_t = s, t < \mathrm{i}(s^*) \mid s_0). \end{aligned}$$

Given that the transition happens before reaching s^*, the transition itself is not dependent on the fact that it happens before s^* and it starts with s_0. Therefore, $P_\pi(s_{t+1} = s' \mid s_t = s, t < \mathrm{i}(s^*), s_0) = T(s'|s, \pi)$. Returning to the derivation, $L_{s^*}^{s_0,\pi}(s')$

$$= \sum_{t=0}^{\infty} \gamma^t \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) P_\pi(s_t = s, t < \mathrm{i}(s^*) \mid s_0).$$

Since the probability of a starting state does not depend on the policy, $P(s_0 = s, t < \mathrm{i}(s^*) \mid s_0) = P(s_0 = s \mid s_0)$ and the summation is not counted on s^*,

$$\begin{aligned} &= \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) P(s_0 = s \mid s_0) + \sum_{t=1}^{\infty} \gamma^t \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) P_\pi(s_t = s, t < \mathrm{i}(s^*) \mid s_0) \\ &= \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) P(s_0 = s \mid s_0) + \sum_{t=1}^{\infty} \gamma^t \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) P_\pi(s_t = s, t < \mathrm{i}(s^*) \mid s_0) \\ &= \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) P(s_0 = s \mid s_0) + \sum_{s \in S \setminus \{s^*\}} T(s'|s, \pi) \sum_{t=1}^{\infty} \gamma^t P_\pi(s_t = s, t < \mathrm{i}(s^*) \mid s_0) \end{aligned}$$

$$= \sum_{s \in S \setminus \{s^*\}} T(s'|s,\pi) P(s_0 = s \mid s_0) + \sum_{s \in S \setminus \{s^*\}} T(s'|s,\pi) \gamma \sum_{t=0}^{\infty} \gamma^t P_\pi(s_{t+1} = s, t < \mathrm{i}(s^*) \mid s_0).$$

By the definition of L,

$$= \sum_{s \in S \setminus \{s^*\}} T(s'|s,\pi) P(s_0 = s \mid s_0) + \sum_{s \in S \setminus \{s^*\}} T(s'|s,\pi) \gamma L_{s^*}^{s_0,\pi}(s')$$

$$= \sum_{s \in S \setminus \{s^*\}} T(s' \mid s,\pi) \left(P(s_0 = s \mid s_0) + \gamma L_{s^*}^{s_0,\pi}(s) \right).$$

□

Definition 4.2.4 (Posterior Occupancy Function)**.** *The posterior state occupancy of a state $s \in S$ given the start state s_0 and the intermediary state s^* is defined by*

$$M_{s^*}^{s_0,\pi}(s) = \sum_{t=0}^{\infty} \gamma^t P_\pi(s_{t+1} = s, t \geq \mathrm{i}(s^*) \mid s_0).$$

Corollary 4.2.1. *Given the anterior and posterior state occupancy with respect to the start state s_0 and the intermediary state s^*, the following holds:*

$$K^{s_0,\pi}(s') = L_{s^*}^{s_0,\pi}(s') + M_{s^*}^{s_0,\pi}(s').$$

Proof.

$$\begin{aligned} L_{s^*}^{s_0,\pi}(s') + M_{s^*}^{s_0,\pi}(s') &= \sum_{t=0}^{\infty} \gamma^t P_\pi(s_{t+1} = s, t < \mathrm{i}(s^*) \mid s_0) + \sum_{t=0}^{\infty} \gamma^t P_\pi(s_{t+1} = s, t \geq \mathrm{i}(s^*) \mid s_0) \\ &= \sum_{t=0}^{\infty} \gamma^t \left(P_\pi(s_{t+1} = s, t < \mathrm{i}(s^*) \mid s_0) + P_\pi(s_{t+1} = s, t \geq \mathrm{i}(s^*) \mid s_0) \right) \end{aligned}$$

Since $P(A, B) + P(A, \neg B) = P(A)$,

$$= \sum_{t=0}^{\infty} \gamma^t P_\pi(s_{t+1} = s \mid s_0)$$

$$= K^{s_0,\pi}(s').$$

□

Lemma 4.2.5 (Posterior Occupancy Recurrence Relation)**.** *Let M be a posterior state occupancy function following a policy π with respect to a starting state s_0 and an intermediary state s^*, then the following two equations hold:*

If any intermediary state is not in S_0, then

$$M_{s^*}^{s_0,\pi}(s') = \gamma T(s' \mid s^*, \pi) L_{s^*}^{s_0,\pi}(s^*) + \gamma \sum_s T(s' \mid s, \pi) M_{s^*}^{s_0,\pi}(s).$$

For any intermediary state in S, then

$$M_{s^*}^{s_0,\pi}(s') = T(s' \mid s^*, \pi) \left(P(s_0 = s' \mid s_0) + \gamma L_{s^*}^{s_0,\pi}(s^*)\right) + \gamma \sum_s T(s' \mid s, \pi) M_{s^*}^{s_0,\pi}(s).$$

Proof. From Corollary 4.2.1,

$$M_{s^*}^{s_0,\pi}(s') = K^{s_0,\pi}(s') - L_{s^*}^{s_0,\pi}(s').$$

By the definitions of both K and L,

$$\begin{aligned}
&= \sum_s T(s' \mid s, \pi)(P(s_0 = s \mid s_0, \pi) + \gamma K^{s_0,\pi}(s)) \\
&\quad - \sum_{s \in S \setminus \{s^*\}} T(s' \mid s, \pi)(P(s_0 = s \mid s_0, \pi) + \gamma L_{s^*}^{s_0,\pi}(s)) \\
&= T(s' \mid s^*, \pi)(P(s_0 = s^* \mid s_0, \pi) + \gamma K^{s_0,\pi}(s^*)) \\
&\quad + \sum_{s \in S \setminus \{s^*\}} T(s' \mid s, \pi)\gamma(K^{s_0,\pi}(s) - L_{s^*}^{s_0,\pi}(s)) \\
&= T(s' \mid s^*, \pi)(P(s_0 = s^* \mid s_0, \pi) + \gamma K^{s_0,\pi}(s^*)) \\
&\quad + \sum_{s \in S \setminus \{s^*\}} T(s' \mid s, \pi)\gamma M_{s^*}^{s_0,\pi}(s) \\
&= T(s' \mid s^*, \pi)(P(s_0 = s^* \mid s_0, \pi) + \gamma L_{s^*}^{s_0,\pi}(s^*) + \gamma(K^{s_0,\pi}(s^*) - L_{s^*}^{s_0,\pi}(s^*))) \\
&\quad + \sum_{s \in S \setminus \{s^*\}} T(s' \mid s, \pi)\gamma M_{s^*}^{s_0,\pi}(s) \\
&= T(s' \mid s^*, \pi)(P(s_0 = s^* \mid s_0, \pi) + \gamma L_{s^*}^{s_0,\pi}(s^*)) + \sum_{s \in S} T(s' \mid s, \pi)\gamma M_{s^*}^{s_0,\pi}(s).
\end{aligned}$$

□

Overall frequency functions can be defined for anterior and posterior functions.

Definition 4.2.5 (Overall Anterior and Posterior Occupancy Frequency)**.** *For any $s \in S$ and $s^* \in S$, an* overall anterior occupancy frequency function *of a policy π can*

Algorithm 2 Overall Anterior Occupancy Frequency Iteration Algorithm

Input: a policy π, the transition function T, the distribution of start states, $P(s_0 = s)$, $s \in S_0$, an intermediary state s^*, and a threshold θ that determines the accuracy of the approximation.

Output: $L \approx L_*$

```
procedure AnteriorOccupancy(π, T, P(s_0 = s), s*, θ)
    repeat
        Initialize all L(s)s with 0
        Δ ← 0
        for all s ∈ S \ {s*} do
            l ← L(s)
            L(s) ← Σ_s T(s' | s, π) (P(s_0 = s) + γL(s))
            Δ = max(Δ, |l − L(s)|)
        end for
    until Δ < θ
    return L
end procedure
```

be defined as

$$L^{\pi}_{s^*}(s) = \mathop{\mathbb{E}}_{s' \in S_0} \left[L^{s',\pi}_{s^*}(s) \right].$$

For any $s \in S$ and $s^* \in S$, an overall posterior occupancy frequency function *of a policy π can be defined as*

$$M^{\pi}_{s^*}(s) = \mathop{\mathbb{E}}_{s' \in S_0} \left[M^{s',\pi}_{s^*}(s) \right].$$

Lemma 4.2.6 (Overall Anterior and Posterior Occupancy Frequency Recurrence Relation)**.** *Let L be an overall anterior occupancy frequency function following a policy π starting from a set of states S_0 with respect to an intermediary state s^*. Then, the following holds:*

$$L^{\pi}_{s^*}(s') = \sum_{s \in S \setminus \{s^*\}} T(s' \mid s, \pi) \left(P(s_0 = s) + \gamma L^{\pi}_{s^*}(s) \right).$$

Algorithm 3 Overall Posterior Occupancy Frequency Iteration Algorithm

Input: a policy π, the transition function T, the distribution of start states, $P(s_0 = s)$, $s \in S_0$, an intermediary state s^*, an anterior occupancy frequency function w.r.t. s^* $L^{\pi}_{s^*}$, and a threshold θ which determines the accuracy of the approximation.

Output: $M \approx M_*$

procedure PosteriorOccupancy($\pi, T, P(s_0 = s), s^*, L^{\pi}_{s^*}, \theta$)
  repeat
    Initialize all $M(s)$s with 0
    $\Delta \leftarrow 0$
    for all $s \in S$ **do**
      $m \leftarrow M(s)$
      $M(s) \leftarrow T(s' \mid s^*, \pi)\,(P(s_0 = s^*) + L^{\pi}_{s^*}(s^*)) + \sum_s T(s' \mid s, \pi)\gamma M(s)$
      $\Delta = \max(\Delta, |m - M(s)|)$
    end for
  until $\Delta < \theta$
  return M
end procedure

Let M be an overall posterior occupancy frequency function following a policy π starting from a set of states S_0 with respect to an intermediary state s^*. Then, the following holds:

If any intermediary state is or is not in S_0 then,

$$M^{\pi}_{s^*}(s') = T(s' \mid s^*, \pi)\,(P(s_0 = s^*) + \gamma L^{\pi}_{s^*}(s^*)) + \gamma \sum_s T(s' \mid s, \pi) M^{\pi}_{s^*}(s).$$

If any intermediary state is not in S_0 then,

$$M^{\pi}_{s^*}(s') = \gamma T(s' \mid s^*, \pi) L^{\pi}_{s^*}(s^*) + \gamma \sum_s T(s' \mid s, \pi) M^{\pi}_{s^*}(s).$$

Proof. Proofs are similar to the proof of Lemma 4.2.2. □

The algorithms for the above recurrence relations are in Alg. 2 and Alg. 3, respectively.

Before defining the anterior and posterior values, we revisit the meaning of the equation $\mathbb{1}_{t<1^{\tau}(s^*)}$. It is 1 if the current step t precedes reaching s^* or if s^* is not included in τ. Otherwise, it is 0.

Definition 4.2.6 (Anterior and Posterior Values)**.** *Given a start state s_0 and intermediary state s^*, the expected accumulated reward before reaching s^* is defined as the* anterior value*:*

$$V^L_{s^*,\pi}(s_0) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\mathrm{i}^{\tau}(s^*)} \gamma^t r^\tau_t \mid s_0, \pi\right].$$

In other words, using the above $\mathbb{1}$ notation,

$$V^L_{s^*,\pi}(s_0) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\infty} \gamma^t r^\tau_t \cdot \mathbb{1}_{t\leq 1^{\tau}(s^*)} \mid s_0, \pi\right].$$

Since the posterior value is counting rewards in the exactly opposite situation, it is defined as

$$V^M_{s^*,\pi}(s_0) = \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\infty} \gamma^t r^\tau_t \cdot \mathbb{1}_{t>1^{\tau}(s^*)} \mid s_0, \pi\right].$$

Corollary 4.2.2 (Value Decomposition)**.** *Using the above definitions, we can easily conclude that*

$$V_\pi(s_0) = V^{L,s^*}_\pi(s_0) + V^{M,s^*}_\pi(s_0).$$

Proof.

$$V^L_{s^*,\pi}(s_0) + V^{M,s^*}_\pi(s_0)$$

$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t^\tau \cdot \mathbb{1}_{t\leq 1^\tau(s^*)} \mid s_0,\pi\right] + \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t^\tau \cdot \mathbb{1}_{t> 1^\tau(s^*)} \mid s_0,\pi\right]$$

Since $\mathbb{1}_{t>1^\tau(s^*)} = 1 - \mathbb{1}_{t\leq 1^\tau(s^*)}$,

$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t^\tau \cdot \left(\mathbb{1}_{t\leq 1^\tau(s^*)} + (1 - \mathbb{1}_{t\leq 1^\tau(s^*)})\right) \mid s_0,\pi\right]$$
$$= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t^\tau \cdot \mid s_0,\pi\right] = V_\pi(s_0).$$

□

Lemma 4.2.7. *Given anterior and posterior occupancy frequency functions L and M with respect to a start state s_0 and an intermediary state s^*, an anterior and posterior value with respect to s_0 and s^* can be represented as*

$$V^L_{s^*,\pi}(s_0) = \sum_{s\in S}\left[R(s)\cdot L^{\pi,s_0}_{s^*}(s)\right],$$

$$V^M_{s^*,\pi}(s_0) = \sum_{s\in S}\left[R(s)\cdot M^{\pi,s_0}_{s^*}(s)\right].$$

The overall anterior and posterior values can also be given as

$$\mathop{\mathbb{E}}_{s\in S_0}\left[V^L_{s^*,\pi}(s_0)\right] = \sum_{s\in S}\left[R(s)\cdot L^{\pi}_{s^*}(s)\right],$$

$$\mathop{\mathbb{E}}_{s\in S_0}\left[V^M_{s^*,\pi}(s_0)\right] = \sum_{s\in S}\left[R(s)\cdot M^{\pi}_{s^*}(s)\right].$$

Proof.

$$
\begin{aligned}
V^{L}_{s^*,\pi}(s_0) &= \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^{\pi}}\left[\sum_{t=0}^{\infty}\gamma^t r_t^{\tau}\cdot \mathbb{1}_{t\leq 1^{\tau}(s^*)} \,|\, s_0,\pi\right] \\
&= \sum_{t=0}^{\infty}\gamma^t \mathop{\mathbb{E}}_{\tau\sim\mathcal{D}^{\pi}}\left[r_t^{\tau}\cdot \mathbb{1}_{t\leq 1^{\tau}(s^*)} \,|\, s_0,\pi\right] \\
&= \sum_{t=0}^{\infty}\gamma^t \sum_{s} R(s)\cdot P_{\pi}(s_{t+1}=s, t<\mathrm{i}(s^*) \,|\, s_0) \\
&= \sum_{s}\left[R(s)\cdot \sum_{t=0}^{\infty}\gamma^t P_{\pi}(s_{t+1}=s, t<\mathrm{i}(s^*) \,|\, s_0)\right] \\
&= \sum_{s}\left[R(s)\cdot L^{\pi,s_0}_{s^*}(s)\right].
\end{aligned}
$$

The proof for $V^{M}_{s^*,\pi}(s_0)$ is similar to the above. The overall anterior and posterior value proofs are also similar to lemma 4.2.3. □

The different M value can be calculated for a counterfactual policy change from π_0 to π_1 when the state s^* is reached if we use the L from the policy π_0:

$$
M^{s_0,\pi_0,\pi_1}_{s^*}(s') = \gamma T(s' \,|\, s^*,\pi_1(s^*))L^{s_0,\pi_0}_{s^*}(s^*) + \gamma\sum_{s} T(s' \,|\, s,\pi_1(s))M^{s_0,\pi_0,\pi_1}_{s^*}(s). \quad (4.1)
$$

4.2.3 Contrasting Different Policies

Our approach to explaining RL agents contrasts a policy π_0 with another policy π_1. Specifically, we want to answer the question of *why* an agent should choose its preferred policy π_0 over a contrasting policy π_1. The comparison of two polices is achieved by situating an agent in a counterfactual situation. We run an agent from a start state or a set of start states and let it follow its preferred policy π_0 until a significant state

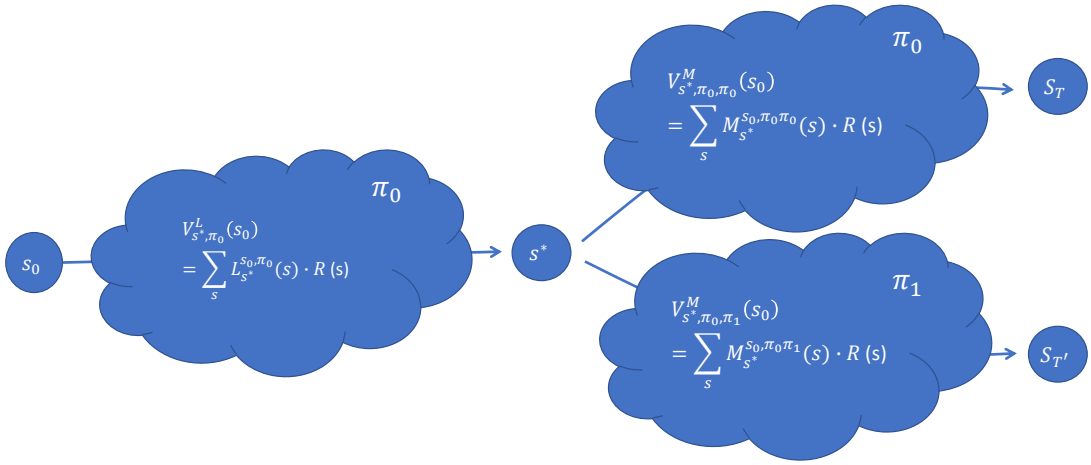


Figure 4.3: Contrasting two different policies via the value decomposition with respect to a state s^*.

s^* is reached. We compare the behavior of an agent that continues with its π_0 policy to one that switches at that point to the contrasting policy π_1.

If we revisit the *do* operator in Chapter 2 as in Fig. 2.1, the controlled variable X can be set to $X = \pi_0 \to \pi_0$ or $X = \pi_0 \to \pi_1$. Then, we can set Y to be $V^M_{s^*,\pi_0,\pi_0}$ or $V^M_{s^*,\pi_0,\pi_1}$ and observe $\mathbb{E}[Y \,|\, do(X = \pi_0 \to \pi_0)]$ and $\mathbb{E}[Y \,|\, do(X = \pi_0 \to \pi_1)]$ to determine the cause of possible outcomes.

The value decomposition approach from the previous section can efficiently compare the value of the original and counterfactual policy. Both policies' expected values can be calculated via:

$$V_{s^*,\pi_0,\pi_0}(s_0) = V^L_{s^*,\pi_0}(s_0) + V^M_{s^*,\pi_0,\pi_0}(s_0), \tag{4.2}$$

$$V_{s^*,\pi_0,\pi_1}(s_0) = V^L_{s^*,\pi_0}(s_0) + V^M_{s^*,\pi_0,\pi_1}(s_0). \tag{4.3}$$

Looking into these value differences, we are comparing the *impact* on the start state value of switching policies, $I_{s^*,\pi_0,\pi_1}(s_0)$, after an agent first visits the intermediate state s^*. This impact is measured by

$$I_{s^*,\pi_0,\pi_1}(s_0) = V_{s^*,\pi_0,\pi_0}(s_0) - V_{s^*,\pi_0,\pi_1}(s_0) = V^M_{s^*,\pi_0,\pi_0}(s_0) - V^M_{s^*,\pi_0,\pi_1}(s_0), \tag{4.4}$$

$$\mathop{\mathbb{E}}_{s_0 \in S_0}\left[I_{s^*,\pi_0,\pi_1}(s_0)\right] = \mathop{\mathbb{E}}_{s_0 \in S_0}\left[V^M_{s^*,\pi_0,\pi_0}(s_0) - V^M_{s^*,\pi_0,\pi_1}(s_0)\right]. \tag{4.5}$$

The expectation of the impact over all the start states measures the overall impacts of switching policies with respect to a intermediate state. To generate a useful explanation, we want to find the state s^* that maximizes this impact, $s^* = \text{argmax}_s \, \mathbb{E}_{s_0 \in S_0}\left[I_{s,\pi_0,\pi_1}(s_0)\right]$. The state is the one that, if an agent changes from π_0 to an alternative policy π_1 upon reaching that state, it will have the most impact on the overall performance of an agent.

Algorithm 4 Impact Algorithm

Input: a base policy π_0, a contrastive policy π_1, the transition function T, the distribution of start states, $P(s_0 = s)$, $s \in S_0$, and a threshold θ that determines the accuracy of the approximation.

Output: $s^*, \mathbb{E}\left[I_{s^*,\pi_0,\pi_1}\right]$

procedure IMPACT($\pi_0, \pi_1, T, P(s_0 = s), \theta$)
  $m \leftarrow -\infty$
  for all $s \in S$ **do**
    $L_s^{\pi_0} \leftarrow$ ANTERIOROCCUPANCY($\pi_0, T, P(s_0 = s), s, \theta$)
    $M_s^{\pi_0,\pi_0} \leftarrow$ POSTERIOROCCUPANCY($\pi_0, T, P(s_0 = s), s, L_s^{\pi_0}, \theta$)
    $M_s^{\pi_0,\pi_1} \leftarrow$ POSTERIOROCCUPANCY($\pi_1, T, P(s_0 = s), s, L_s^{\pi_0}, \theta$)
    $\mathbb{E}\left[I_{s,\pi_0,\pi_1}\right] \leftarrow \sum_{s'} \left(R(s') \cdot M_s^{\pi_0,\pi_0}(s')\right) - \sum_{s'} \left(R(s') \cdot M_s^{\pi_0,\pi_1}(s')\right)$
    if $m < \mathbb{E}\left[I_{s,\pi_0,\pi_1}\right]$ **then**
      $m \leftarrow \mathbb{E}\left[I_{s,\pi_0,\pi_1}\right]$
      $s^* \leftarrow s$
    end if
  end for
  return s^*, m
end procedure

Chapter 5

Experiment and Evaluation

This chapter demonstrates the proposed method using an experiment on the 4×3 GridWorld domain introduced in Russell and Norvig's book [74] and the game of Blackjack. In these domains, we look at three different methods for choosing an influential state to use in our explanation.

5.1 Metrics

The choice of metric is key for selecting a meaningful state to build an explanation around. Here are several options.

5.1.1 Maximum Q-value Difference

The state with the maximum Q-value difference between the two policies π_0 and π_1 is:

$$s_q = \underset{s}{\operatorname{argmax}} \left| Q_{\pi_0}(s, \pi_0(s)) - Q_{\pi_0}(s, \pi_1(s)) \right|.$$

This method answers the question: Which state would have its value change the most if the agent switched from following π_0 to following the action proposed by π_1 for one step at that state? As Q-values encode a simple kind of counter-factual, and Q-values are produced in the context of many RL algorithms, this method is a very natural one to consider. A drawback of this method, however, is it does not take into account where the agent actually starts: s_0 (or a distribution over such states). A state could have very different Q-values, but be so unlikely to be reached that its hypothetical difference is moot.

5.1.2 Maximum Value Difference

The state with the maximum value difference is:

$$s_v = \underset{s}{\operatorname{argmax}} \left| V_{\pi_0}(s) - V_{\pi_1}(s) \right|.$$

This method chooses the state where the two policies differ the most in terms of their state values V. The value of a state $V_\pi(s)$ is the expected discounted return starting from s and following π. The maximum value difference is a direct and simple way to select a state that is very different for the two different policies—it's the place in the state space where following the policies leads to the most extreme difference in

value. It is like the maximum Q-value difference except it considers switching behavior indefinitely, instead of for the single step used in Q values. As such, it provides a stronger contrast and a more meaningful counter-factual. Like the maximum Q-value difference, however, it does not consider the likelihood that the state is reached, resulting in a potentially misleading choice of state.

5.1.3 Impact Using Anterior and Posterior Occupancy Frequencies

The state with the most impact on the start-state value is:

$$s^* = \operatorname*{argmax}_s \mathop{\mathbb{E}}_{s_0 \in S_0} \left[I_{s,\pi_0,\pi_1}(s_0) \right].$$

Our proposed approach evaluates the difference resulting from following π_0 until s is reached, and then following π_1 after that point. Although marginally more computationally complex than the prior two methods, the main advantage of this impact measure is that it accounts for the overall value difference at the start state, and not just what happens when starting at the intermediate state. In this sense, it is a much better choice as the answer to the *why* question of how changing policies impacts the results.

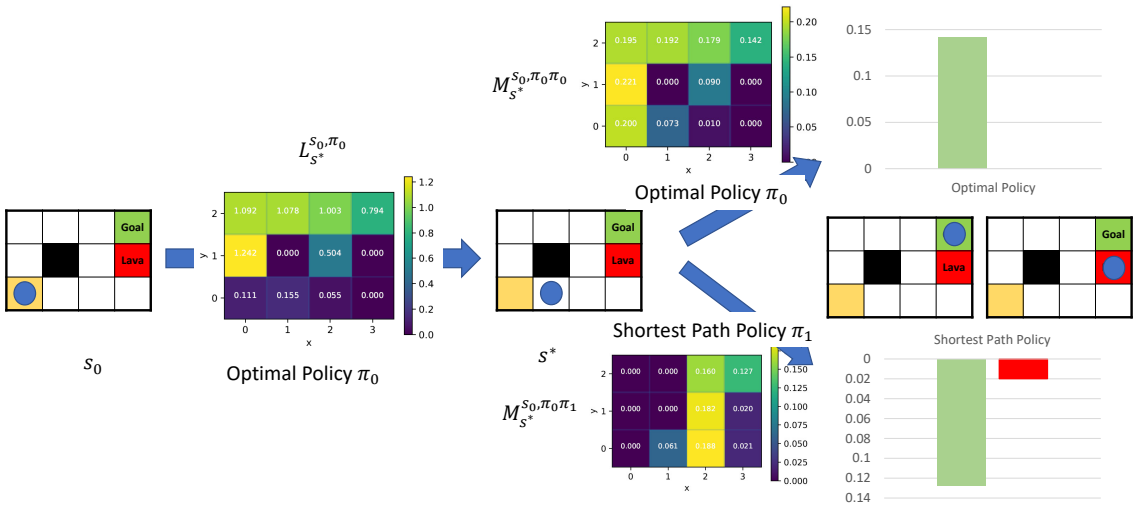


Figure 5.1: GridWorld result. The diagram first shows the status of the states, s_0, s^* and two terminal states, by showing the grid and the agents' locations. It shows the occupancy frequency grids before and after the state s^* and two contrasting result graphs in terms of the probability of ending in two different states, *goal* (green) and *lava* (red).

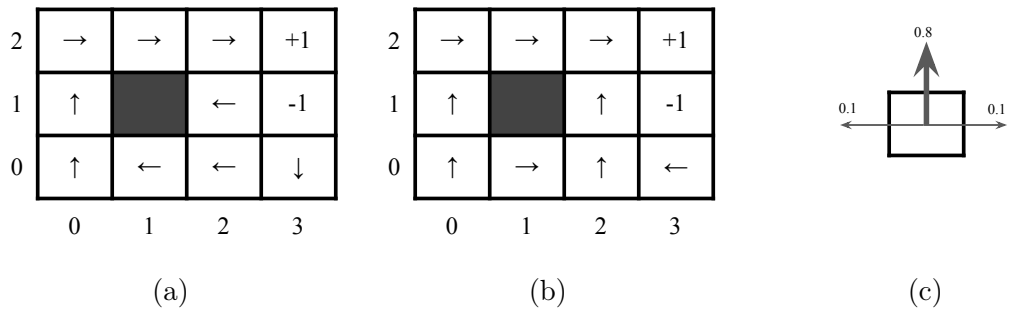
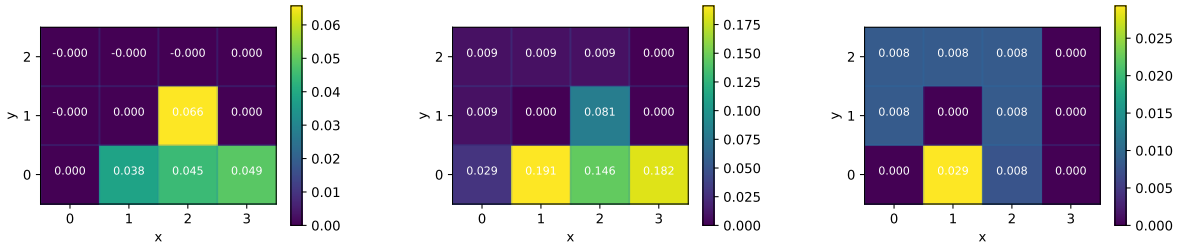


Figure 5.2: GridWorld policies. (a) An optimal policy π_0 with $R(\cdot,\cdot,s) = -0.04, \forall s \in S, s \notin S_T$. (b) Shortest path policy π_1 with the same R. (c) A diagram showing the slip probability of an action.

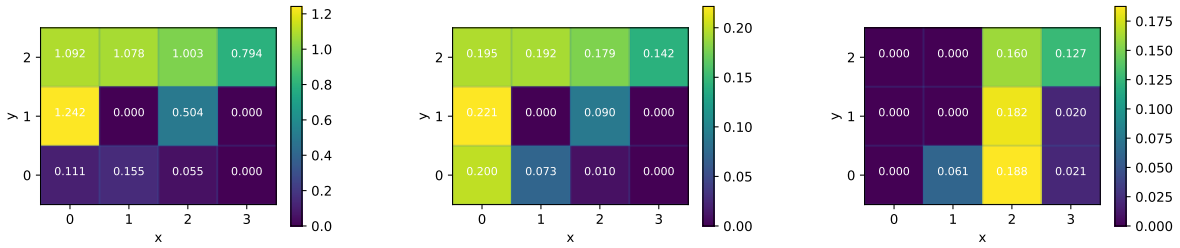
5.2 GridWorld

The GridWorld is fully observable and has four actions, *Up*, *Down*, *Left*, and *Right*. For each action, an agent will go to an intended state with the probability 0.8 and will move at right angles to the original direction for the rest. Taking each step at



(a) (b) (c)

Figure 5.3: A comparison of three evaluation metrics. (a) Q-value difference. (b) Value difference. (c) Impact using anterior and posterior occupancy frequencies.



(a) (b) (c)

Figure 5.4: The anterior and posterior occupancy frequencies of the state $s^* = (1, 0)$ (a) $L_{s^*}^{s_0,\pi_0}(s)$. (b) $M_{s^*}^{s_0,\pi_0,\pi_1}(s)$. (c) $M_{s^*}^{s_0,\pi_0,\pi_1}(s)$.

non-terminal states, the agent receives the reward -0.04 and, at two terminal states, the goal and lava, it receives $+1$ or -1, respectively as in Fig. 5.2 (c).

The polices for the comparison are depicted in Fig. 5.2 (a) and (b). The optimal policy is generated from the value iteration method and the shortest path policy is generated to minimize the total number of steps before reaching the goal state with the reward $+1$. The occupancy frequencies L and M can also be generated using Algorithm 2 and 3 and the calculated L and M for the state $(1, 0)$ are depicted in Fig. 5.4.

The results for the three metrics are shown in Fig. 5.3. Metrics (A) and (B) choose

the same state, $(2, 1)$ and the *impact* metric chooses a different state. For (A), we can explain the result as:

> If an agent starts at $(2, 1)$, the difference of expected returns between the optimal policy and that shortest path policy is maximized with a difference of 0.066.

For (B), we can explain the result as:

> If an agent starts at $(2, 1)$, the difference of expected returns between taking the action *Left* suggested by the optimal policy and taking the action *Up* suggested by the shortest path policy (then continuing with the optimal policy) is maximized with a difference of 0.081.

The proposed *Impact* metric chooses a different state for its explanation, $(1, 0)$. We can generate the following explanation using our occupancy frequencies and factoring states according by [44]:

> If an agent switches from the optimal policy to the shortest path policy at the state $(1, 0)$, the difference in value at the start state $(0, 0)$ is maximized with a difference of 0.029.

Furthermore, details about the likelihood of the different terminal states provide additional insight. See Fig. 5.1.

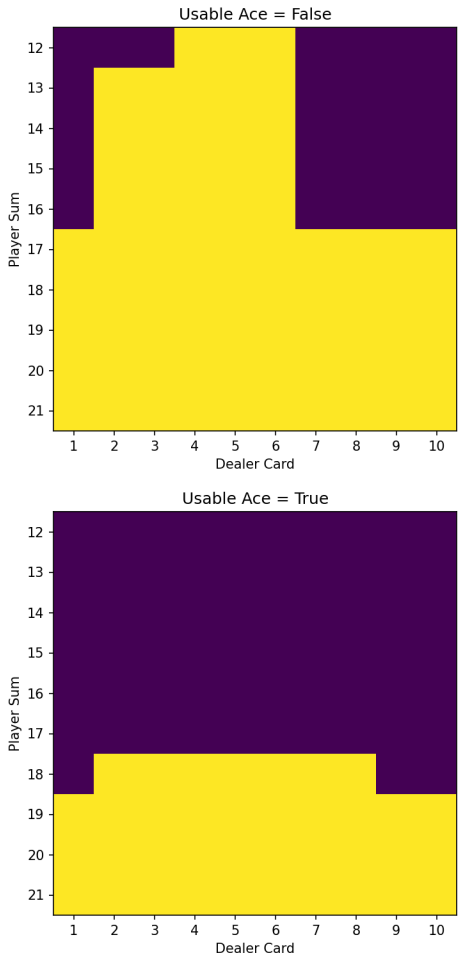


Figure 5.5: Blackjack optimal policy. Yellow color indicates *stick* action and violet color indicates *hit* action.

5.3 Blackjack

Blackjack is a card game played in casinos. The main goal of the game is to obtain a point total higher than the dealer without exceeding the total of 21. Ace cards can be worth either 1 or 10 when summing up the total points, whichever is most beneficial. At each step, the player can either choose *hit* or *stand*. A player that chooses to hit receives an additional card. A player choosing to stand receives no more cards and

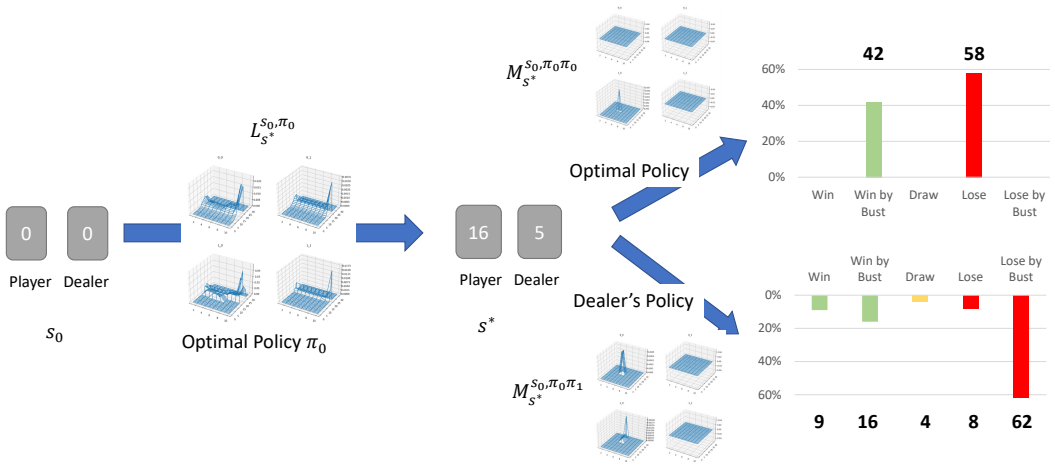


Figure 5.6: Blackjack result. The diagram first shows the status of the states, $s_0(0,0), s^*(16,5)$ and two terminal states, by showing the player's and dealer's sums. It shows the occupancy frequency graphs before and after the state s^* and two contrasting result graphs in terms of the probability of ending in five different states.

the dealer plays until the total sum is at least 17. For this experiment, we compared the optimal policy (see Fig. 5.5) computed by value iteration to what would happen if the player adopted the dealer's policy. The highest impact state for this pair of policies is $(16, 10)$, where the player has the sum 16 and the dealer has the sum 10.

Below is an explanation generated for this situation.

> If an agent switches from the optimal policy to the dealer policy at the state $(16, 5)$, the difference in value at the start state $(0, 0)$ is maximized with a difference of 0.029.

It is also possible to provide more visual representation of an explanation by showing the graphs with more data as in Fig. 5.6.

Chapter 6

Conclusion

The two different parts of the work presented in this dissertation seek to explain the behavior of reinforcement-learning agents in very different ways. However, the methods do share one main theme: counterfactual reasoning. Our main experimental setting is depicted in Fig. 6.1. The first part of my work mostly focused on experimenting on changing the environmental features $\Phi(s)$ to see how their outcomes $\mathbb{E}\left[V \mid do(\phi(s))\right]$ differ. The second part of the work delves more into the effect of varying the policy. It formally assesses different policies Π and finds a state that supports an important explanation: whether a policy change at that state can be the *cause* of a different outcome, $\mathbb{E}\left[V \mid do(\pi)\right]$. This final chapter elaborates the implications and limitations of these two methods as well as potential research questions that arise based on these approaches.

The first part of this dissertation delves into the problem of generalization in

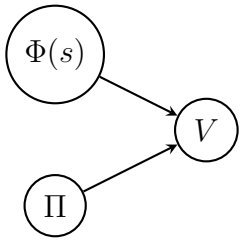


Figure 6.1: Two different main causes in an agent's decision making. Environmental features $\Phi(s)$ and policies Π can affect the overall outcome V of an agent.

reinforcement learning. It started with a belief that the high-performing learned agents using the current state-of-the-art algorithms have a general understanding of their environments and the ability to adapt to different situations and therefore behave appropriately in counterfactual situations. In such a setting, we can explain an agent's decision in counterfactual situations by contrasting outcomes. That is because such agents are expected to behave in a way that makes outcomes maximally beneficial to the agents. However, our results suggest that such agents perform poorly in unfamiliar environments, especially when they are not only pixel-wise different, but also contextually very different.

To better distinguish different sets of states that are more often or seldom visited during learning, we defined a new taxonomy classifying *reachable*, *unreachable*, *on-policy*, and *off-policy* states. We proposed a set of novel practical methods for evaluating generalization based on these new sets of states as a new benchmark task for deep RL agents. These results bring us to a question: What is the extent to which deep RL agents learn generalized representations? Our conclusion is that deep RL agents need *not* learn generalized representations, even when they perform well overall. To

support the claim that agents can learn general representations, more experimentation and analysis is necessary.

Subsequent to this work, researchers have attempted to improve the testing of generalization in reinforcement-learning agents and high-level representation learning. My colleagues Tosch et al. [91] built a framework to also provide contextual variations in two more games, Space Invaders and Breakout, in the Atari environment. Cobbe et al. [11]'s work further explored generalization of deep RL agents using the traditional train-and-test performance measures. Although they showed learned agents can adapt to different stage designs, it still remains unproven that its learned representations are high level enough to be adapted to contextual changes in their environments. Badia et al. [4] trained an agent with one single policy to play all 57 Atari games. One might think that it may have learned to play video games in general, but they have not conducted a test that it can play a completely different video game beyond the 57 Atari games used in training. We still need an architecture for which its inner components can represent high-level human-readable notions of its environment and visualizations of how it uses such notions in its architecture to chose an action. Research on human-readable state abstractions can help achieve such a goal. I hope that the taxonomy of differently trainable sets of states provided in this dissertation can help better test different architectural approaches for designing better policy-approximation methods.

For the second part of my approach, I presented a framework for choosing a state that has the most impact in overall value and provided detailed explanations based

on the calculated anterior and posterior occupancy functions. This approach also leverages the power of reasoning counterfactually, since we can assume the same precondition and reason about the outcome based on the decision a designer can make by continuing or switching to a different policy. This work basically extends the previous work of Khan et al. [44] in terms of providing explanations using occupancy frequency. In addition, my work also provides formal derivations showing that all three occupancy frequency functions can be calculated via numerical iterations and values of start states can be decomposed into two parts with respect to a state.

The proposed method is currently limited to be used only in finite state domains. However, the formalization should be extensible to continuous-state domains. One approach would be to use a human-designed discretization of states combined with a sampling method. Another would be to make distribution approximations extended via importance sampling.

Applying the notion of explanation to the occupancy frequency functions themselves might also be very interesting. They contain rich contextual information on where an agent has visited before and after reaching a certain state. Since such a state is significant in decision-making before and after changing the policy, the whereabouts of an agent across two different sets of time can be informative to a human user.

My work is based on the use of factored-state MDPs. Factorizations of MDP the state space is based on classifying states based on their rewards. If human-readable abstract states capturing the occupancy frequency functions could be added to this

explanation process, it should be even more informative.

In conclusion, in the first part of my dissertation I focused mostly on *what* is necessary to provide correct explanations and in the second part, I worked on *how* to provide such explanations. This combination could be leveraged to make reinforcement-learning agents more effective, more transparent, and more trustworthy.

Bibliography

[1] Dan Amir and Ofra Amir. Highlights: Summarizing agent behavior to people. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1168–1176, 2018.

[2] Andrew Anderson, Jonathan Dodge, Amrita Sadarangani, Zoe Juozapaitis, Evan Newman, Jed Irvine, Souti Chattopadhyay, Alan Fern, and Margaret Burnett. Explaining reinforcement learning to mere mortals: An empirical study. *arXiv preprint arXiv:1903.09708*, 2019.

[3] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743*, 2019.

[4] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International Conference on Machine Learning*, pages 507–517. PMLR, 2020.

[5] Elias Bareinboim. Causal reinforcement learning, 2015. URL `https://www.youtube.com/watch?v=bwz3NpVfz6k`.

[6] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, Jun 2013.

[7] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[8] Tom Bewley and Jonathan Lawry. Tripletree: A versatile interpretable representation of black box agents and their environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11415–11422, 2021.

[9] Or Biran and Courtenay Cotton. Explanation and justification in machine learning: A survey. In *IJCAI-17 workshop on explainable AI (XAI)*, volume 8, page 1, 2017.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[11] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[12] Francisco Cruz, Richard Dazeley, and Peter Vamplew. Memory-based explainable reinforcement learning. In *Australasian Joint Conference on Artificial Intelligence*, pages 66–77. Springer, 2019.

[13] Francisco Cruz, Richard Dazeley, Peter Vamplew, and Ithan Moreira. Explainable robotic systems: Understanding goal-driven actions in a reinforcement learning scenario. *Neural Computing and Applications*, pages 1–18, 2021.

[14] Mohamad H Danesh, Anurag Koul, Alan Fern, and Saeed Khorram. Re-understanding finite-state representations of recurrent policy networks. In *International Conference on Machine Learning*, pages 2388–2397. PMLR, 2021.

[15] Giang Dao, Indrajeet Mishra, and Minwoo Lee. Deep reinforcement learning monitor for snapshot recording. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 591–598. IEEE, 2018.

[16] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. `https://github.com/openai/baselines`, 2017.

[17] Thomas Dodson, Nicholas Mattei, Joshua T Guerin, and Judy Goldsmith. An english-language argumentation interface for explanation generation with markov decision processes in the domain of academic advising. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 3(3):1–30, 2013.

[18] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[19] Upol Ehsan and Mark O Riedl. Human-centered explainable ai: Towards a reflective sociotechnical approach. In *International Conference on Human-Computer Interaction*, pages 449–466. Springer, 2020.

[20] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[21] Lasse Espeholt, Raphaël Marinier, Piotr Stanczyk, Ke Wang, and Marcin Michalski. Seed rl: Scalable and efficient deep-rl with accelerated central inference. *arXiv preprint arXiv:1910.06591*, 2019.

[22] Gregory Farquhar, Tim Rocktäschel, Maximilian Igl, and Shimon Whiteson. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. *arXiv preprint arXiv:1710.11417*, 2017.

[23] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *IET Conference Proceedings*, pages 850–855(5), January 1999.

[24] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of

machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[25] Omer Gottesman, Joseph Futoma, Yao Liu, Sonali Parbhoo, Leo Celi, Emma Brunskill, and Finale Doshi-Velez. Interpretable off-policy evaluation in reinforcement learning by highlighting influential transitions. In *International Conference on Machine Learning*, pages 3658–3667. PMLR, 2020.

[26] Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. *arXiv preprint arXiv:1711.00138*, 2017.

[27] LEE Haeyeon, Yasuhiro Ota, Cynthia Breazeal, and Jun Ki Lee. Methods of robot behavior generation and robots utilizing the same, June 10 2014. US Patent 8,751,042.

[28] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

[29] Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 56 (4):843–887, 2005.

[30] Mohammadhosein Hasanbeig, Natasha Yogananda Jeppu, Alessandro Abate,

Tom Melham, and Daniel Kroening. Deepsynth: Automata synthesis for automatic task segmentation in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7647–7656, 2021.

[31] Matthew J Hausknecht and Peter Stone. The impact of determinism on learning atari 2600 games. In *AAAI Workshop: Learning for General Competency in Video Games*, 2015.

[32] Bradley Hayes and Julie A Shah. Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–312. IEEE, 2017.

[33] Daniel Hein, Alexander Hentschel, Thomas Runkler, and Steffen Udluft. Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence*, 65:87–98, 2017.

[34] Daniel Hein, Steffen Udluft, and Thomas A Runkler. Interpretable policies for reinforcement learning by genetic programming. *Engineering Applications of Artificial Intelligence*, 76:158–169, 2018.

[35] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[36] Andreas Holzinger, Georg Langs, Helmut Denk, Kurt Zatloukal, and Heimo Müller. Causability and explainabilty of artificial intelligence in medicine. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, page e1312, 2019.

[37] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Interpretable text-to-image synthesis with hierarchical semantic layout generation. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 77–95. Springer, 2019.

[38] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado van Hasselt, and David Silver. Distributed prioritized experience replay. In *International Conference on Learning Representations*, 2018.

[39] Sandy H Huang, Kush Bhatia, Pieter Abbeel, and Anca D Dragan. Establishing appropriate trust via critical states. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3929–3936. IEEE, 2018.

[40] David Hume. An enquiry concerning human understanding. In *Seven masterpieces of philosophy*. Routledge, 2016.

[41] Aman Jhunjhunwala. Policy extraction via online q-value distillation. Master's thesis, University of Waterloo, 2019.

[42] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski,

Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[43] Sham M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003.

[44] Omar Khan, Pascal Poupart, and James Black. Minimal sufficient explanations for factored markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 19, pages 194–200, 2009.

[45] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*, 2017.

[46] Anurag Koul, Sam Greydanus, and Alan Fern. Learning finite state representations of recurrent policy networks. *arXiv preprint arXiv:1811.12530*, 2018.

[47] Isaac Lage, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir. Exploring computational user models for agent policy summarization. In *IJCAI: proceedings of the conference*, volume 28, page 1401. NIH Public Access, 2019.

[48] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering

symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*, pages 5979–5989. PMLR, 2021.

[49] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

[50] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521 (7553):436, 2015.

[51] David K Lewis. Causal explanation. *Philosophical Papers 2*, pages 214–240, 1986.

[52] Erik Lindholm, John Nickolls, Stuart Oberman, and John Montrym. Nvidia tesla: A unified graphics and computing architecture. *IEEE micro*, 28(2):39–55, 2008.

[53] Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.

[54] Tania Lombrozo. The structure and function of explanations. *Trends in cognitive sciences*, 10(10):464–470, 2006.

[55] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[56] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *arXiv preprint arXiv:1709.06009*, 2017.

[57] Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable reinforcement learning through a causal lens. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 2493–2500, 2020.

[58] Stephanie Milani, Nicholay Topin, Manuela Veloso, and Fei Fang. A survey of explainable reinforcement learning. *arXiv preprint arXiv:2202.08434*, 2022.

[59] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[60] Brent Mittelstadt, Chris Russell, and Sandra Wachter. Explaining explanations in ai. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 279–288. ACM, 2019.

[61] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[62] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous

methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[63] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[64] W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[65] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, 2015.

[66] Anh Nguyen, Jason Yosinski, and Jeff Clune. Understanding neural networks via feature visualization: A survey. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 55–76. Springer, 2019.

[67] Ali Nouri, Michael L Littman, Lihong Li, Ronald Parr, Christopher Painter-Wakefield, and Gavin Taylor. A novel benchmark methodology and data repository for real-life reinforcement learning. In *Proceedings of the 26th international conference on machine learning*, 2009.

[68] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In *Advances in Neural Information Processing Systems*, pages 6118–6128, 2017.

[69] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[70] Martin L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[71] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[72] Ribana Roscher, Bastian Bohn, Marco F Duarte, and Jochen Garcke. Explainable machine learning for scientific insights and discoveries. *arXiv preprint arXiv:1905.08883*, 2019.

[73] Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

[74] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3 edition, 2010.

[75] Wojciech Samek. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.

[76] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[77] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016.

[78] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

[79] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[80] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[81] Pedro Sequeira, Eric Yeh, and Melinda T Gervasio. Interestingness elements for explainable reinforcement learning through introspection. In *IUI workshops*, volume 1, 2019.

[82] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, pages 1855–1865. PMLR, 2020.

[83] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[84] David Silver, Hado van Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3191–3199. JMLR. org, 2017.

[85] Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Tldr: Policy summarization for factored ssp problems using temporal abstractions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 272–280, 2020.

[86] Walter Dan Stiehl, Cynthia Breazeal, Jun Ki Lee, Allan Z Maymin, Heather Knight, Robert L Toscano, and Iris M Cheung. Interactive systems employing robotic companions, December 9 2014. US Patent 8,909,370.

[87] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[88] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[89] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[90] Nicholay Topin and Manuela Veloso. Generation of policy-level explanations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2514–2521, 2019.

[91] Emma Tosch, Kaleigh Clary, John Foley, and David Jensen. Toybox: A suite of environments for experimental evaluation of deep reinforcement learning. *arXiv preprint arXiv:1905.02825*, 2019.

[92] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[93] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[94] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

[95] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[96] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[97] Xinzhi Wang, Shengcheng Yuan, Hui Zhang, Michael Lewis, and Katia Sycara. Verbal explanations for deep reinforcement learning neural networks with attention on extracted features. In *2019 28th IEEE International Conference on*

Robot and Human Interactive Communication (RO-MAN), pages 1–7. IEEE, 2019.

[98] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[99] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*, 2016.

[100] Laurens Weitkamp, Elise van der Pol, and Zeynep Akata. Visual rationalizations in deep reinforcement learning for atari games. In *Benelux Conference on Artificial Intelligence*, pages 151–165. Springer, 2018.

[101] Shimon Whiteson, Brian Tanner, Matthew E Taylor, and Peter Stone. Protecting against evaluation overfitting in empirical reinforcement learning. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011 IEEE Symposium on*, pages 120–127. IEEE, 2011.

[102] Sam Witty, Jun Ki Lee, Emma Tosch, Akanksha Atrey, Michael Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. *arXiv preprint arXiv:1812.02868*, 2018.

[103] Sam Witty, Jun Ki Lee, Emma Tosch, Akanksha Atrey, Kaleigh Clary, Michael L

Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. *Applied AI Letters*, 2(4):e45, 2021.

[104] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

[105] Herman Yau, Chris Russell, and Simon Hadfield. What did you think would happen? explaining agent behaviour through intended outcomes. *Advances in Neural Information Processing Systems*, 33:18375–18386, 2020.

[106] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.

[107] Hengzhe Zhang, Aimin Zhou, and Xin Lin. Interpretable policy derivation for reinforcement learning based on evolutionary feature synthesis. *Complex & Intelligent Systems*, 6(3):741–753, 2020.

[108] Qiyuan Zhang, Xiaoteng Ma, Yiqin Yang, Chenghao Li, Jun Yang, Yu Liu, and Bin Liang. Learning to discover task-relevant features for interpretable reinforcement learning. *IEEE Robotics and Automation Letters*, 6(4):6601–6607, 2021.

[109] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred–rule extraction from deep neural networks. In *International Conference on Discovery Science*, pages 457–473. Springer, 2016.