Interpreting Reinforcement Learning with Shapley Values

Daniel Beechey, Thomas M. S. Smith, Özgür Şimşek
Department of Computer Science
University of Bath, United Kingdom
djeb20@bath.ac.uk

Abstract

For reinforcement learning systems to be widely adopted, their users must understand and trust them. We present a theoretical analysis of explaining reinforcement learning using Shapley values, following a principled approach from game theory for identifying the contribution of individual players to the outcome of a cooperative game. We call this general framework Shapley Values for Explaining Reinforcement Learning (SVERL). Our analysis exposes the limitations of earlier uses of Shapley values in reinforcement learning. We then develop an approach that uses Shapley values to explain agent performance. In a variety of domains, SVERL produces meaningful explanations that match and supplement human intuition.

1 Introduction

Reinforcement learning systems have potential for significant impact in real-world applications. To be widely adopted, it is useful for these systems to not only perform well but also be explainable.

Methods for explaining reinforcement learning can be categorised as *intrinsically interpretable* or *post-hoc*. Intrinsically interpretable approaches improve the transparency of models by substituting an opaque model with a more understandable one, such as a decision tree. This approach often leads to a reduction in representational power. In contrast, post-hoc methods hold no constraints on the complexity of the model, treating it as a black box. Reinforcement learning systems with the largest potential to positively benefit society depend on function approximators with large representational power, such as deep neural networks. We therefore focus on post-hoc explanation methods.

An established, post-hoc explanation method for supervised learning uses Shapley values [1], a principled approach from game theory for identifying the contribution of individual players to the outcome of a cooperative game. Shapley values are the result of a rigorous mathematical formulation that satisfies four axioms of fairness. In supervised learning, Shapley values explain a model by expressing the contribution of individual features to the predictions of the model.

We analyse, from first principles, how Shapley values can be used to explain reinforcement learning. We make three main contributions. First, we develop a theoretical framework for using Shapley values in the context of reinforcement learning, showing that earlier uses of Shapley values in reinforcement learning are incorrect or incomplete. Secondly, we consider which aspects of reinforcement learning are important to explain, arguing that explaining agent performance is an important and overlooked element. Thirdly, we develop a principled approach that identifies the contributions of state features to the performance of an agent.

We call this general framework Shapley Values for Explaining Reinforcement Learning (SVERL). In a variety of domains, SVERL produces meaningful explanations that match and supplement human intuition.[1]

[1]This paper contains substantially the same content as an earlier work by the authors, titled Explaining Reinforcement Learning with Shapley Values, presented at ICML 2023.

17th European Workshop on Reinforcement Learning (EWRL 2024).

2 Background

We model the interaction of an agent with its environment as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, p_0)$, where $\mathcal{S}$ denotes the set of states, $\mathcal{A}$ the set of actions, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, $\gamma \in [0, 1]$ the discount factor, and $p_0 : \mathcal{S} \rightarrow [0, 1]$ the initial state distribution. At decision stage t, $t \geq 0$, the agent observes the current state of the environment, $s_t \in \mathcal{S}$, and executes action $a_t \in \mathcal{A}(s_t)$. Consequently, the environment transitions to a new state, $s_{t+1} \sim p(\cdot|s_t, a_t)$, and returns reward r_{t+1} whose expected value is $r(s_t, a_t)$. The objective is to learn a policy π that maximises the expected return $\mathbb{E}_\pi[G_t]$, where $G_t = \sum_{k=t}^{\infty} \gamma^k r_{k+1}$. The policy can be stochastic, $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, or deterministic, $\pi : \mathcal{S} \rightarrow \mathcal{A}$. A state-value function, $V^\pi(s)$, gives the expected return from state s when following policy π, $V^\pi(s) = \mathbb{E}_\pi[G_t|s_t = s]$. A state-action value function, $Q^\pi(s, a)$, gives the expected return from state s if the agent executes action a and follows policy π thereafter, $Q^\pi(s, a) = \mathbb{E}_\pi[G_t|s_t = s, a_t = a]$. The optimal state value function is denoted by V^* and the optimal state-action value function by Q^*.

We assume that an environment has a set of n state features, $\mathcal{F} = \{0, \ldots, n-1\}$, where we can decompose the state space according to the state features, $\mathcal{S} = \mathcal{S}_0 \times \ldots \times \mathcal{S}_{n-1}$, and each state can be represented as an ordered set: $s = \{s_i | s_i \in \mathcal{S}_i\}_{i=0}^{n-1}$. For example, in a classic gridworld domain, a state could be the agent's location, with x and y coordinates as state features. Let $\mathrm{C} \subset \mathcal{F}$ be a set of observable state features. Then a partial observation of a state is the ordered set $s_\mathrm{C} = \{s_i | i \in \mathrm{C}\}$.

Shapley values assign the contributions of individual players to the outcome of a cooperative game [1]. They are the unique solution to a set of mathematical axioms that specify fair distribution of credit across players. A cooperative game is defined by a set $\mathcal{F}$ of players and a characteristic value function $v : 2^{|\mathcal{F}|} \rightarrow \mathbb{R}$, where $v(\mathrm{C})$ returns the outcome of the game when played by some coalition of players $\mathrm{C} \subseteq \mathcal{F}$, with $v(\emptyset) = 0$. The Shapley value of player i in the game $(\mathcal{F}, v)$ is:

$$\phi_i(v) = \sum_{\mathrm{C} \subseteq \mathcal{F} \setminus \{i\}} \frac{|\mathrm{C}|! \, (|\mathcal{F}| - |\mathrm{C}| - 1)!}{|\mathcal{F}|!} \cdot \delta(i, \mathrm{C}), \tag{1}$$

where $\delta(i, \mathrm{C}) = v(\mathrm{C} \cup \{i\}) - v(\mathrm{C})$ is the marginal gain in characteristic value when player i joins coalition C. As an example, the employees of a company can be modelled as players in a game where profit is the characteristic value function.

Shapley values have been adopted in machine learning to determine the contribution of features to the predictions of supervised learning models [2]. Let $f_\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ be a supervised learning model defined over a set of n features, $\mathcal{F} = \{0, \ldots, n-1\}$, such that $\mathcal{X} = \mathcal{X}_0 \times \ldots \times \mathcal{X}_{n-1}$ and each $\mathbf{x} \in \mathcal{X}$ can be represented as an ordered set, $\mathbf{x} = \{x_i | x_i \in \mathcal{X}_i\}_{i=0}^{n-1}$. Then Shapley values show the contribution of feature $x_i \in \mathbf{x}$ to the target $y = f_\mathcal{F}(\mathbf{x})$ for the single point $\mathbf{x}$. As an example, when predicting the quality of wine using features such as acidity, pH, and alcohol [3], the Shapley values show how much each feature contributes to the predicted quality of a specific wine. This is done by modelling the prediction at $\mathbf{x}$ as a game, where the features $\{x_0, \ldots, x_{n-1}\}$ are the players and the target prediction $y = f_\mathcal{F}(\mathbf{x})$ is the outcome of the game. Then the Shapley values $\phi_i(f, \mathbf{x})$, specifying the contribution of feature x_i to the prediction $y = f(\mathbf{x})$, are computed using the characteristic value function:

$$v^f(\mathrm{C}) \coloneqq f_\mathrm{C}(\mathbf{x}),$$

where $\mathrm{C} \subseteq \mathcal{F}$ and $f_\mathrm{C}(\mathbf{x})$ is the model's prediction for the ordered set $\mathbf{x}_\mathrm{C} = \{x_i | i \in \mathrm{C}\}$. The resulting Shapley values satisfy $f_\mathcal{F}(\mathbf{x}) = v^f(\emptyset) + \sum_{i \in \mathcal{F}} \phi_i(f, \mathbf{x})$.

Shapley values show each feature's contribution to the change in prediction when all features are known, $f_\mathcal{F}(\mathbf{x})$, compared to when no features are known, $f_\emptyset(\mathbf{x}) = v^f(\emptyset)$. In game theory, the value of a game with no players is zero. Hence $v(\emptyset) = 0$. In supervised learning, the prediction when no features are known is the expected model prediction over the data distribution. Hence $v^f(\emptyset) = \mathbb{E}_{p(\mathbf{x})}[f(\mathbf{x})]$, where $p(\mathbf{x})$ is the data distribution, the probability that a randomly sampled point from $\mathcal{X}$ equals $\mathbf{x}$.

Computing Shapley values requires predictions, $f_\mathrm{C}(\mathbf{x})$, to be made for all subsets of features, $\mathrm{C} \subseteq \mathcal{F}$. The original approach to approximating such predictions was to retrain the model for all $\mathrm{C} \subseteq \mathcal{F}$ [4]. With a large number of features, this is infeasible. An alternative method defines the prediction at $\mathbf{x}$ with subset of features C as:

$$f_\mathrm{C}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{x}')} \left[f_\mathcal{F}(\mathbf{x}_\mathrm{C} \cup \mathbf{x}'_{\bar{\mathrm{C}}}) \right], \tag{2}$$

where $p(\mathbf{x}')$ is the data distribution [5, 6]. Equation (2) can be approximated by marginalising over possible values for the unobserved features $\bar{\mathrm{C}} = \mathcal{F} \setminus \mathrm{C}$. Assuming independent features and sampling n data points,

$$f_{\mathrm{C}}(\mathbf{x}) = \lim_{n\to\infty} \frac{1}{n} \sum_{\mathbf{x}'\sim p(\mathbf{x}')} f_{\mathcal{F}}(\mathbf{x}_{\mathrm{C}} \cup \mathbf{x}'_{\bar{\mathrm{C}}}). \tag{3}$$

Using Equation (3), an unbiased approximation algorithm for calculating Shapley values samples a marginal gain:

$$\hat{\delta}(i, \mathrm{C}) = f_{\mathcal{F}}(\mathbf{x}_{\mathrm{C}\cup\{i\}} \cup \mathbf{x}'_{\overline{\mathrm{C}\cup\{i\}}}) - f_{\mathcal{F}}(\mathbf{x}_{\mathrm{C}} \cup \mathbf{x}'_{\bar{\mathrm{C}}}), \tag{4}$$

where the coalition $\mathrm{C} \subseteq \mathcal{F} \setminus \{i\}$ is sampled proportional to the multinomial term in Equation (1) and $\mathbf{x}' \sim p(\mathbf{x}')$. The mean of these samples is the Shapley value in the limit [5]. This algorithm does not require retraining the models. It is one of the approximations used in the popular python package SHAP [7], which calculates Shapley values for an arbitrary machine learning model. There are other approximations included in SHAP; they all approximate Equation (2) in some way.

Equation (2) is referred to as *off-manifold*. It makes the simplifying assumption that the features are independent. When features are correlated, this assumption samples points $\mathbf{x}_{\mathrm{C}} \cup \mathbf{x}'_{\bar{\mathrm{C}}}$ that may not lie on the data manifold. Without this simplifying assumption, the prediction at $\mathbf{x}$ with subset of features C becomes:

$$f_{\mathrm{C}}(\mathbf{x}) = \mathbb{E}_{p(\mathbf{x}'|\mathbf{x}_{\mathrm{C}})} \left[f_{\mathcal{F}}(\mathbf{x}') \right], \tag{5}$$

where the conditional data distribution $p(\mathbf{x}'|\mathbf{x}_{\mathrm{C}})$ takes into account the feature correlations [8]. An *on-manifold* sampling method that uses Equation (4) but now samples $x' \sim p(\mathbf{x}'|\mathbf{x}_{\mathrm{C}})$ can then be used to approximate Shapley values for models with correlated features.

Shapley values $\phi_i(f, \mathbf{x})$ provide the local contribution of features to a prediction. The local contributions can be combined to identify the global Shapley value for a feature, producing the mean contribution of a feature to a model's predictions: $\Phi_i(f) = \mathbb{E}_{p(\mathbf{x})} \left[\phi_i(f, \mathbf{x}) \right]$. If we consider a new characteristic value function, defined using a model's loss ℓ, $v^{\ell}(\mathrm{C}) := \ell\left(f_{\emptyset}(\mathbf{x}), y\right) - \ell\left(f_{\mathrm{C}}(\mathbf{x}), y\right)$, then global Shapley values can be interpreted as the contribution of feature i to the model's prediction accuracy [9]:

$$\Phi_i(f) = \mathbb{E}_{p(\mathbf{x})} \left[\phi_i(v^{\ell}, \mathbf{x}) \right].$$

In reinforcement learning, earlier work has directly applied the SHAP package to an agent's policy [10–15] or state-value function [16, 17] in an effort to explain reinforcement learning in specific applications. This earlier work implicitly assumes that the state features are independent because SHAP implements only off-manifold approximations. More importantly, this earlier work has not explored the theoretical basis for what the resulting Shapley values mean in the context of reinforcement learning.

In the following sections, we present a theoretical and empirical analysis of how Shapley values can be used to explain reinforcement learning, starting from first principles. We refer to this general framework as Shapley Values for Explaining Reinforcement Learning (SVERL).

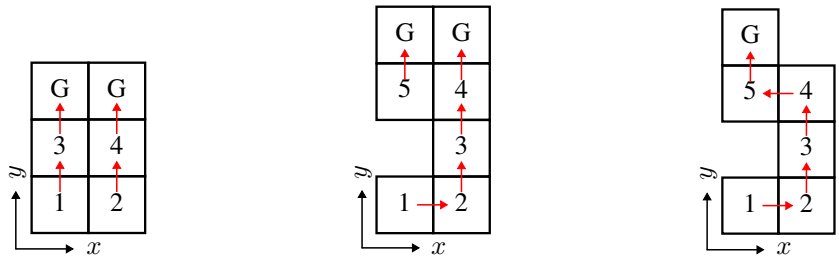
3 Using Shapley Values to Explain Reinforcement Learning

We start by exploring the use of Shapley values to explain the value function and the policy of an agent. Our analysis shows that (1) applying Shapley values to a value function produces explanations that have no relation to the performance or behaviour of an agent, and (2) applying Shapley values to policies explains the contribution of state features to an agent's decisions but not to its performance.

Shapley values applied to value functions. In order to use Shapley values to explain an agent's value function, we follow the theory of on-manifold Shapley values in supervised learning to propose the following characteristic value functions for V and Q:

$$v^{\hat{V}}(\mathrm{C}) := \hat{V}^{\pi}_{\mathrm{C}}(s) = \sum_{s'\in\mathcal{S}} p^{\pi}(s'|s_{\mathrm{C}}) \hat{V}^{\pi}(s') \tag{6}$$

$$v^{\hat{Q}}(\mathrm{C}) := \hat{Q}^{\pi}_{\mathrm{C}}(s, a) = \sum_{s'\in\mathcal{S}} p^{\pi}(s'|s_{\mathrm{C}}) \hat{Q}^{\pi}(s', a) \tag{7}$$



(a) Gridworld-A (b) Gridworld-B (c) Gridworld-C

Figure 1: Deterministic gridworlds, with actions North, South, East, and West. The numbers in each grid square show the state identifier. The initial state is either state 1 or state 2, with equal probability. The reward is -1 for each action and an additional $+10$ for transitions into a terminal state (G). The discount factor γ is 1. State features are the x and y coordinates. The red arrows show the optimal action in each state.

Equations (6) and (7) account for feature correlations by using the conditional limiting state occupancy distribution $p^\pi(s'|s_C)$, the probability of being in state s' given that s_C is observed and the agent is following policy π.

Shapley values resulting from Equation (6) satisfy $v^{\hat{V}}(\mathcal{F}) = v^{\hat{V}}(\emptyset) + \sum_{i\in\mathcal{F}} \phi_i(v^{\hat{V}}, s)$. They show each feature's contribution to the change in characteristic value when all state features are observed, $\hat{V}^\pi(s)$, compared to when no state features are observed, $\hat{V}^\pi_\emptyset(s)$. This observation also holds for Equation (7) and all other characteristic value functions for reinforcement learning presented in this paper.

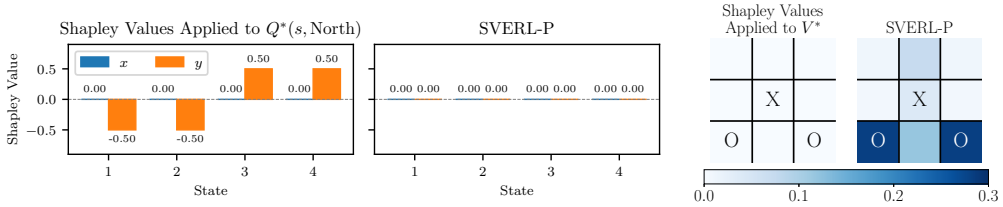
One might expect the Shapley values resulting from Equations (6) and (7) to relate to performance in some way, given that a value function represents an agent's prediction of how well its policy performs. However, these characteristic value functions refer to the expected return of the agent's *original* policy π. Not observing a state feature is likely to change an agent's policy, which in turn changes the expected return. By never evaluating any change in policy, the full consequences of removing state features are not being considered. Consequently, the resulting explanations do not meaningfully relate to performance or behaviour.

Instead, Shapley values applied to the value function explain the contribution of each feature to the *predictions* of the value function—but only under the assumption that all features will be observed by the agent when acting in the environment. This is a subtle but important point. Shapley values applied to the value function do not explain the agent's performance; they explain the value function as a predictor—but without considering the impact of features on behaviour.

We use two examples to illustrate the difference between explaining the value function as a predictor and explaining agent performance. We use Equation (7) to apply Shapley values to Q^* in Gridworld-A, shown in Figure 1a, and Equation (6) to apply Shapley values to V^* in Tic-Tac-Toe.

In Gridworld-A, the optimal action is North (N) in each state. Intuitively, if the optimal action is the same in all states, then the contribution of each state feature to performance should be zero. However, Shapley values applied to $Q^*(s, N)$, shown in Figure 2a (left), produce non-zero contributions for the y state feature.

To explore why, consider the contribution of y in state 1. If neither x nor y is known, the agent is equally likely to be in states 1, 2, 3, or 4 (we are ignoring terminal states), with $Q^*(s, N)$ values of 8, 8, 9, and 9, respectively. Consequently, the predicted return is 8.5. Now consider the marginal gain from observing y. If y is known to be 1 and x remains unknown, the agent is equally likely to be in states 1 and 2, with $Q^*(s, N) = 8$ for both states, yielding a predicted return of 8. Hence, the marginal gain in prediction from observing y is $8 - 8.5 = -0.5$. Similarly, if x is known to be 1 and y is unknown, the agent is equally likely to be in states 1 and 3, with $Q^*(s, N)$ values of 8 and 9, respectively, yielding a predicted return of 8.5. If y is also known, then predicted return is $Q^*(1, N) = 8$. Hence, the marginal gain in prediction when observing y is again $8 - 8.5 = -0.5$. Both marginal gains are -0.5, resulting in a Shapley value of -0.5 for y in state 1.



(a) Gridworld-A (b) Tic-Tac-Toe

Figure 2: (a) Shapley values applied to a state-action value function and SVERL-P, presented in Section 4, in Gridworld-A (Figure 1a). (b) Shapley values applied to a state-value function and SVERL-P in a Tic-Tac-Toe state. Shapley values are represented using a colour scale projected onto each cell.

In contrast, the *actual* return from state 1 is 8, whatever combination of features is observed, because the optimal policy selects North in every state. Human intuition therefore assigns a contribution of 0 because observing y does not change the agent's behaviour or expected return. Shapley values applied to the value function is not capable of capturing this relationship.

In Tic-Tac-Toe, there are 9 features, corresponding to each board position, with possible values `X`, `O` or `empty`. Consider an agent (`X`) that uses V^* to play optimally against an opponent (`O`) that follows a Minimax policy [18]. The reward is -1 for losing and 0 for drawing, the only possible outcomes when playing against Minimax. In the state shown in Figure 2b, the two squares marked by the opponent inform the agent that it needs to make a blocking move. Intuitively, we would expect the corresponding two state features to impact the performance of the agent. However, the feature contributions identified by applying Shapley values to V^* are zero for every state feature. The reason is that an optimal agent always draws, hence the optimal value function always predicts a return of zero, independently of which state features are observed. These Shapley values explain the value function as a static predictor. They do not consider that the value function depends on the policy, which would change in the absence of some state features.

Shapley values applied to policies. We follow the theory of on-manifold Shapley values in supervised learning to propose the following characteristic value function for a policy $\tilde{\pi} : \mathcal{S} \to \mathcal{A}$ that outputs actions:

$$v^{\tilde{\pi}}(\mathrm{C}) := \tilde{\pi}_{\mathrm{C}}(s) = \sum_{s' \in \mathcal{S}} p^{\tilde{\pi}}(s'|s_{\mathrm{C}})\tilde{\pi}(s'), \tag{8}$$

and the following characteristic value function for a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ that outputs action probabilities:

$$v^{\pi}(\mathrm{C}) := \pi_{\mathrm{C}}(a|s) = \sum_{s' \in \mathcal{S}} p^{\pi}(s'|s_{\mathrm{C}})\pi(a|s'). \tag{9}$$

Equations (8) and (9) account for feature correlations by using the conditional limiting state occupancy distribution $p^{\pi}(s'|s_{\mathrm{C}})$, as in Equations (6) and (7). We note that Equation (8) is not valid in discrete action spaces because it is not meaningful to sum discrete actions.

The characteristic value functions in Equations (8) and (9) produce Shapley values that show the contribution of state features, respectively, to the action selected by an agent and to the probability of selecting action a. Both values provide information on the contributions of state features to the decision made by the agent. This insight is valuable but we argue that there is more to be understood and explained about the decision. Specifically, these Shapley values reveal no insight into the importance of state features for an agent's performance.

As an illustrative example, imagine an agent planning the shortest route through a city. The agent arrives at a junction where turning left and turning right both result in an optimal route. Assume that the agent's policy is to turn left if it observes a road sign (a state feature), and to turn right otherwise. Shapley values applied to the agent's policy would assign a large contribution to the road sign, which is justified and improves our understanding of the agent's behaviour. The sign was indeed instrumental in the agent's decision to turn left. However, one would be incorrect to then conclude that the sign is important for the agent to perform well. On the contrary, because turning left and turning right are both optimal, the sign contributes nothing to the agent's performance. This insight can be gained only by considering the effect of removing state features on the agent's performance.

Therefore, we make a distinction between explaining why the agent acted in a specific way and explaining how features impact agent performance.

The contributions of state features to the value function or to the policy do not reveal insight into contributions to agent performance. These two approaches consider either the contributions to predicting expected return independent of behaviour or the contributions to behaviour independent of expected return. We have highlighted the limitations of both approaches. Next we propose an approach to explaining reinforcement learning by identifying contributions of state features to agent performance.

4 Explaining Agent Performance

Here we provide a formulation of Shapley values to explain the performance of a reinforcement learning agent. We present two methods that explain either the local or the global contributions of state features to performance. Each approach reveals unique insight that improves understanding. In both approaches, state features are removed from an agent's observation for certain states, then the performance of the resulting policy is evaluated using expected return. We call this approach SVERL-Performance (SVERL-P).

Local explanations. Local SVERL-P explains the contributions of state features to performance from state s by considering removing state features from an agent's observation of state s. For some policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ to be explained, the local SVERL-P characteristic value function is given by:

$$v^{\text{local}}(\text{C}) := \mathbb{E}_{\hat{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right], \text{ where } \hat{\pi}(a_t|s_t) = \begin{cases} \pi_{\text{C}}(a_t|s_t) & \text{if } s_t = s, \\ \pi(a_t|s_t) & \text{otherwise.} \end{cases} \tag{10}$$

Shapley values resulting from Equation (10) show the contribution of each feature to the change in performance when all state features are observed in state s, $v^{\text{local}}(\mathcal{F})$, compared to when no state features are observed in state s, $v^{\text{local}}(\emptyset)$.

In most problems, state features are not independent, so we use the theory for on-manifold Shapley values to propose sampling a from the agent's policy given that it observes s_{C}:

$$\pi_{\text{C}}(a|s) = \mathbb{E}_{p^{\pi}(s'|s_{\text{C}})}\left[\pi(a|s')\right], \tag{11}$$

where we suggest the conditional data distribution in Equation (5) becomes the conditional limiting state occupancy distribution $p^{\pi}(s'|s_{\text{C}})$.

The resulting explanations are specific to the policy used, which can be any possible policy, including a suboptimal policy. One can interpret π_{C} as the policy that best tries to match the behaviour of the original policy π given that features are missing. Policy π_{C} will not usually be able to perfectly mimic the behaviour of policy π. It is exactly this difference in behaviour that causes the change in performance.

Global explanations. Local SVERL-P considers the contributions of state features to performance from a single state. In addition to such local contributions, one may wish to understand the contributions of state features to performance globally. For example, in autonomous driving, a user may wish to understand which parts of an autonomous vehicle's observations are most important for driving performance, to focus resources on improving those parts of the road system. Some state features might contribute substantially to performance in certain states, such as breaking when observing a human or pulling over when an ambulance approaches, while road markings may be globally important by contributing to agent performance in many states.

To quantify the global impact of state features on agent performance, we consider the effect of removing state features from every state in an environment. The corresponding (global) SVERL-P characteristic value function is as follows:

$$v^{\text{global}}(\text{C}) := \mathbb{E}_{\pi_{\text{C}}}\left[\sum_{t=0}^{\infty} \gamma^t r_{t+1} | s_0 = s\right]. \tag{12}$$

Equation (12) produces Shapley values that show the contribution of state features to performance in state s and all future states that follow. These Shapley values are still conditioned on state and

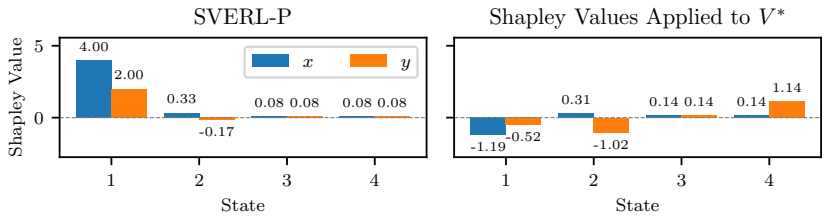


Figure 3: SVERL-P and Shapley values applied to a value function for x and y features in Gridworld-B (Figure 1b).

therefore not a truly global explanation method. To produce a fully global explanation, one can marginalise over the state space using the limiting state occupancy distribution, producing global SVERL-P:

$$\Phi_i(v^{\text{global}}) = \mathbb{E}_{p^\pi(s)}\left[\phi_i\left(v^{\text{global}}, s\right)\right]. \tag{13}$$

Equation (13) gives the contribution of a state feature to the performance of the agent in its environment. An alternative is to marginalise over the initial state distribution p_0, which would place undue attention on the initial states and is therefore less useful in infinite-horizon problems.

5 Experiments

We present experimental results in a variety of domains. We contrast SVERL-P with applying Shapley values to policies and to value functions in Appendix A.2, demonstrating the limitations of the latter approaches. All Shapley values are calculated exactly, as described in Appendix A.4. The domains are fully described in Appendix A.1.

Gridworld-B. We first consider Gridworld-B, shown in Figure 1b. Imagine an agent acting optimally: choosing East (E) in state 1 and North (N) in every other state.

Consider local explanations for specific states. Whatever the state, if neither x nor y is known, the agent cannot know the optimal action with certainty but it knows that the optimal action is either N or E and that N is more likely than E.

In states 3 and 4, either the x or the y feature is sufficient for the agent to take the optimal action N; in other words, x and y features make an equal contribution to agent performance. Furthermore, this contribution is rather small because, if neither feature is known, N is still the likely optimal action.

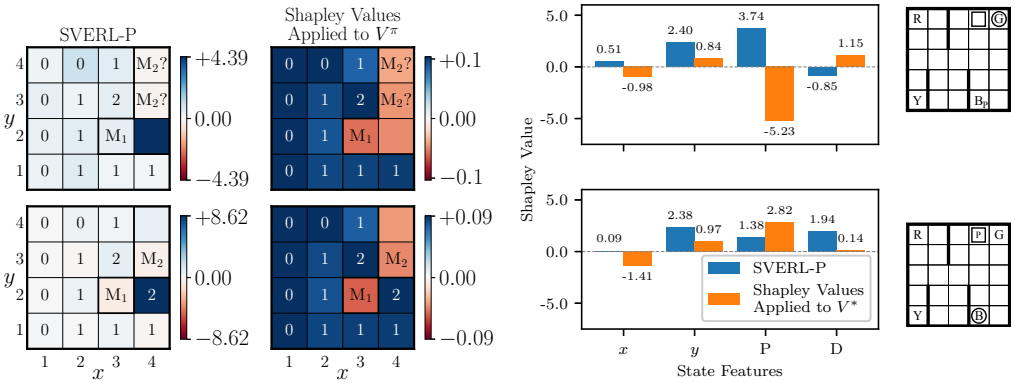
In state 1, the x feature is sufficient for the agent to take the optimal action E (because an optimal agent is never in state 5). The y feature also improves the agent's performance, but by a smaller amount, because it increases the probability of the agent selecting the optimal action E. In sum, the x and y features contribute positively to agent performance, with the x feature contributing more.

In state 2, the x feature is sufficient for the agent to take the optimal action N while the y feature actually decreases the probability of selecting the optimal action N (it increases the probability of selecting the suboptimal action E). The x feature therefore makes a positive contribution to agent performance while the y feature makes a negative contribution.

Local SVERL-P contributions are shown on the left of Figure 3. SVERL-P values align with our intuitive analysis of the domain. As expected, in states 3 and 4, both x and y contribute a small, equal amount to agent performance. Also as expected, in state 1, x contributes more to performance than y. And we can now quantify the difference precisely: x contributes exactly twice as much as y. In state 2, Shapley values once again mirror our expectations, with x contributing positively to performance and y contributing negatively—a reminder that a little bit of knowledge can be a dangerous thing.

Next, consider the global contribution of the two features. Based on the discussion above, the x feature positively contributes larger amounts, more often, to the agent's performance than the y feature. Therefore, we expect the global contribution of the x feature to be larger than that of the y feature. These expectations align with global SVERL-P contributions: 1.43 for x and 0.64 for y.

SVERL-P has correctly and precisely expressed the local and global contribution of the features x and y to performance. It has done so in more detail and precision than our intuitive expectations, demonstrating the value of SVERL-P even in such a simple domain.



(a) Minesweeper. (b) Taxi.

Figure 4: (a) SVERL-P and Shapley values applied to a value function for two successive states (top to bottom) of Minesweeper, represented as the colour of each cell. The domain contains two mines, hidden from the agent. In the top state, the state features reveal the exact location of one mine and two potential locations for the second mine, marked for reference as "M_1" and "M_2?" respectively. The exact location of the second mine is then revealed in the second state, marked as "M_2" for reference. (b) SVERL-P contributions contrasted with Shapley values applied to a value function for two states in the Taxi domain. State features are the x and y coordinates of the taxi, passenger location (P), and destination location (D). The taxi location is marked with a rectangle, the passenger location is marked with a p and the destination location is circled. In the top state, the passenger is at location B and the destination is location G. In the bottom state, the passenger is in the taxi and the destination is location B.

Minesweeper. This is a relatively large domain, with approximately 175,000 states, where it can be difficult to identify how individual state features contribute to performance by reasoning alone. By using SVERL-P, we find local explanations of performance that reveal novel insight into the two successive Minesweeper states shown in Figure 4a.

The features in this domain are the 16 grid squares, with possible values 0, 1, 2, or unopened. Figure 4a shows that one feature in particular ($x = 4, y = 2$) contributes substantially to performance in both states, with all other features contributing relatively little in comparison. On further inspection, we see that the feature $(4, 2)$ is the *only* feature that can exactly determine the location of M_2. On the other hand, many features reveal the exact location of M_1. To act optimally, the agent must determine the exact location of M_2 so the feature $(4, 2)$ is the most important one for completing the episode successfully.

Notice the negative SVERL-P contributions for the squares with possible mines. These are discussed in detail in Appendix A.3.

Taxi. In the taxi domain [19], the agent picks up a passenger and drops them off at their destination. Rewards are -1 for all actions, an additional $+20$ for dropping a passenger at the correct destination, and an additional -10 for attempting to pick up or drop off the passenger at an inappropriate location. We examine the two states shown in Figure 4b.

In the state shown on the top panel, to successfully complete the episode, the agent must first pick up the passenger. Knowledge of the passenger location is therefore vital and we expect this feature to contribute a large amount to performance. This is captured by SVERL-P, as shown in Figure 4b. Conversely, until the passenger has been collected, we do not expect the destination location to contribute positively to performance. Surprisingly, SVERL-P shows that observing the destination location actually reduces the agent performance. Upon closer review, we see that, in this state, observing the destination location without the passenger location increases the probability of navigating towards the destination, which is a suboptimal action.

In the state shown on the lower panel in Figure 4b, the passenger is in the taxi, to be dropped off at location B. The optimal policy navigates to the drop-off location with the passenger in the taxi. Intuitively, both the passenger and the destination location are important, as shown by the SVERL-P contributions.

6 Discussion

We presented a theoretical and empirical analysis of using Shapley values for explaining reinforcement learning (SVERL), starting from first principles, and demonstrated the limitations of existing work. We then developed SVERL-P, a method that uses Shapley values to explain agent performance. SVERL-P considers the consequences of removing features by explicitly deriving an agent's policy and quantifying the change in performance. Our results show that SVERL-P produces meaningful explanations in a variety of reinforcement learning problems, matching and supplementing human intuition.

In most real-world applications, it is computationally expensive to calculate the SVERL-P characteristic value functions exactly. So the characteristic value functions, and hence the Shapley values, must be approximated. Here we outline an approximation algorithm for local SVERL-P based on the on-manifold sampling approach from Shapley values in supervised learning, which has been proven to converge to the Shapley value in the limit [5, 8]. Analogous to the sample in Equation 4, each sample in the algorithm is a marginal gain:

$$\mathbb{E}_{\pi_1}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}|s_0=s\right]-\mathbb{E}_{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t r_{t+1}|s_0=s\right],$$

$$\text{where}\quad \pi_1(a_t|s_t)=\begin{cases}\pi(a_t|s') & \text{if } s_t=s,\\ \pi(a_t|s_t) & \text{otherwise,}\end{cases}\quad\text{and}\quad \pi_2(a_t|s_t)=\begin{cases}\pi(a_t|s'') & \text{if } s_t=s,\\ \pi(a_t|s_t) & \text{otherwise.}\end{cases}$$

A new s' is sampled from $p^\pi(\cdot|s_{\mathrm{C}\cup\{i\}})$, and a new s'' is sampled from $p^\pi(\cdot|s_{\mathrm{C}})$, whenever $s_t=s$. Each coalition $\mathrm{C}\subseteq\mathcal{F}\setminus\{i\}$ is sampled proportional to the multinomial term in the Shapley value calculation. The expected returns can be evaluated using a standard reinforcement learning method, such as Monte Carlo rollouts. This sampling method requires the learning of state occupancy distributions $p^\pi(\cdot|s_{\mathrm{C}})$ for all $\mathrm{C}\subseteq\mathcal{F}$, which is not trivial. We suggest taking inspiration from one of the on-manifold sampling methods proposed by Frye et al. [8]. Importantly, it is likely that these distributions do not need to be learnt exactly because optimal policies usually visit only a small subset of states in large domains.

SVERL is a direct application of Shapley values using specific characteristic value functions suitable for reinforcement learning. All the theoretical guarantees of Shapley values apply to SVERL. Similarly, any advancements in applying Shapley values to supervised learning will apply directly to SVERL. For example, SVERL might be difficult to interpret in domains with thousands of features, such as robotics or vision. However, a method such as *groupShapley* [20], which finds the contribution of groups of features and was developed for supervised learning, could be applied to SVERL, offering computational advantages and simplifying interpretation.

As with any feature-based explanation method, there is further work, often psychological and sociological, to derive useful explanations which improve a user's understanding. It is naturally human to interpret Shapley values subjectively, often developing beliefs and understanding that extend beyond the quantitative information that they provide. These interpretations will likely become more challenging and subjective as the number of features increases. When one proceeds to develop this extended understanding, before acting on it, they must first evaluate whether it is well founded. For example, SVERL-P values allow us to say "this feature contributed x amount to an agent's performance". One can hypothesise on why that feature contributed x but such hypotheses must be tested. These tests depend on the task, explanation and hypothesis. We suggest that future research focuses on (1) the presentation, interpretation and explanatory use of feature attribution techniques such as Shapley values, and (2) methods for evaluating the conclusions drawn from such interpretations. We provide an example in Appendix A.3.

Acknowledgments and Disclosure of Funding

This work was supported by the UKRI Centre for Doctoral Training in Accountable, Responsible and Transparent AI (ART-AI) [EP/S023437/1], the EPSRC Centre for Doctoral Training in Digital Entertainment (CDE) [EP/L016540/1] and the University of Bath. This research made use of Hex, the GPU Cloud in the Department of Computer Science at the University of Bath. We thank our reviewers for a constructive process and the members of the Bath Reinforcement Learning Laboratory for their feedback. We thank Scarllette Ellis for her Minesweeper implementation.

References

[1] Lloyd S Shapley. A value for n-person games. 1953.

[2] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[3] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[4] Erik Strumbelj, Igor Kononenko, and M Robnik Sikonja. Explaining instance classifications with interactions of subsets of feature values. *Data & Knowledge Engineering*, 68(10):886–904, 2009.

[5] Erik Strumbelj and Igor Kononenko. An efficient explanation of individual classifications using game theory. *The Journal of Machine Learning Research*, 11:1–18, 2010.

[6] Erik Strumbelj and Igor Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41:647–665, 2014.

[7] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 2017.

[8] Christopher Frye, Damien de Mijolla, Tom Begley, Laurence Cowton, Megan Stanley, and Ilya Feige. Shapley explainability on the data manifold. In *International Conference on Learning Representations*, 2020.

[9] Ian Covert, Scott M Lundberg, and Su-In Lee. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems*, 33: 17212–17223, 2020.

[10] Stefano Giovanni Rizzo, Giovanna Vantini, and Sanjay Chawla. Reinforcement learning with explainability for traffic signal control. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3567–3572. IEEE, 2019.

[11] Yuyao Wang, Masayoshi Mase, and Masashi Egi. Attribution-based salience method towards interpretable reinforcement learning. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering (1)*, 2020.

[12] Lei He, Nabil Aouf, and Bifeng Song. Explainable deep reinforcement learning for UAV autonomous path planning. *Aerospace Science and Technology*, 118:107052, 2021.

[13] Sindre Benjamin Remman, Inga Strümke, and Anastasios M Lekkas. Causal versus marginal Shapley values for robotic lever manipulation controlled using deep reinforcement learning. In *2022 American Control Conference (ACC)*, pages 2683–2690. IEEE, 2022.

[14] Jakob Løver, Vilde B Gjærum, and Anastasios M Lekkas. Explainable AI methods on a deep reinforcement learning agent for automatic docking. *IFAC-PapersOnLine*, 54(16):146–152, 2021.

[15] Roman Liessner, Jan Dohmen, and Marco A Wiering. Explainable reinforcement learning for longitudinal control. In *ICAART (2)*, pages 874–881, 2021.

[16] Ke Zhang, Peidong Xu, and Jun Zhang. Explainable AI in deep reinforcement learning models: A SHAP method applied in power system emergency control. In *2020 IEEE 4th Conference on Energy Internet and Energy System Integration (EI2)*, pages 711–716. IEEE, 2020.

[17] Ke Zhang, Jun Zhang, Pei-Dong Xu, Tianlu Gao, and David Wenzhong Gao. Explainable AI in deep reinforcement learning models for power system emergency control. *IEEE Transactions on Computational Social Systems*, 9(2):419–427, 2021.

[18] E. Polak. Basics of Minimax algorithms. In *Nonsmooth Optimization and Related Topics*, pages 343–369. Springer, 1989.

[19] Thomas G Dietterich. The MAXQ method for hierarchical reinforcement learning. In *ICML*, volume 98, pages 118–126, 1998.

[20] Martin Jullum, Annabelle Redelmeier, and Kjersti Aas. groupShapley: Efficient prediction explanation with Shapley values for feature groups. *arXiv preprint arXiv:2106.12228*, 2021.

[21] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

A Appendix / supplemental material

A.1 Domains

Gridworld-A, shown in Figure 1a, is a deterministic gridworld. The MDP state represents the grid square occupied by the agent and is described by two features, (x, y), the x and y coordinates of the agent on the grid. There are six states, $\mathcal{S} = \{(1,1),(1,2),(1,3),(2,1),(2,2),(2,3)\}$, two of which are goal states, $G = \{(1,3),(2,3)\}$. The initial state is sampled randomly from the southernmost squares, $\{(1,1),(2,1)\}$. The actions are North, East, South, and West. Reward is -1 for every action taken and an additional $+10$ for transitioning into a goal state, producing a shortest path problem. Actions that attempt to transition an agent out of the grid do not change the state. **Gridworld-B**, shown in Figure 1b, and **Gridworld-C**, shown in Figure 1c, are identical to Gridworld-A in all aspects other than the grid layout and the identity of the goal states.

Gridworld-D is a deterministic 10×10 gridworld, containing 20 grid positions that are impassable blocks, selected uniformly randomly from among all grid positions. There is a single goal state, selected randomly, and fixed across episodes. The initial state is selected randomly from among grid squares that are not impassable blocks or the goal. The domain is identical to Gridworld-A in all other aspects.

Tic-Tac-Toe is a classic game played on a 3×3 grid, where two players take turns to place noughts (`O`) and crosses (`X`). When a player places three noughts or three crosses such that a straight line can be drawn through them, the game ends with a win for the corresponding player. If the grid is full with no winner, the game is a draw. The state has nine features, with each feature representing a specific grid position, taking on values `X`, `O`, or `empty`. The agent plays as `X` and the opponent as `O`. The players have equal probability of playing first. The opponent's policy is the Minimax algorithm [18]. Optimal play against this opponent ends in a draw.

Taxi is a classic reinforcement learning domain by Dietterich [19]. We used the implementation by OpenAI Gym [21]. The domain has a grid with four locations, marked R(ed), G(reen), B(lue) and Y(ellow). There are four state features: $x \in \{1,2,3,4,5\}$, $y \in \{1,2,3,4,5\}$, passenger-location $\in \{\text{R}, \text{G}, \text{B}, \text{Y}, \text{in-taxi}\}$, and destination $\in \{\text{R}, \text{G}, \text{B}, \text{Y}\}$. State features x and y represent the taxi's location. Initial taxi location and destination are selected uniformly randomly. For an episode to terminate successfully, the taxi must navigate to the passenger location, pick-up the passenger, navigate to the destination, and drop-off the passenger. At the beginning of an episode, the passenger location is randomly selected among R, G, B, and Y. Once the passenger has been collected, passenger location becomes in-taxi. The actions are north, south, east, west, pick-up, and drop-off. Pick-up action successfully picks up the passenger only when the taxi and the passenger is at the same grid location. Similarly, drop-off action successfully drops off the passenger when the passenger is in the taxi and the taxi is at the destination. The reward is -1 for each action, an additional $+20$ for delivering the passenger at the destination, and -10 for unsuccessful execution of the pickup or the drop-off action.

Minesweeper is an implementation of the classic game on a 4×4 grid. Each episode resets a grid that contains two hidden mines, each placed randomly. The state has 16 features, with each feature respresenting a specific grid square, taking on values 0, 1, 2 or unopened. Initially, all grid squares are unopened. At each decision stage, the agent selects an unopened square to reveal what is underneath. If it happens to be a number, that number represents the total number of mines in the (up to eight) squares directly surrounding the newly opened square. If the number is zero, all surrounding grid squares are recursively revealed to reveal an area of zeros bordered by strictly positive numbers. The game ends when the agent opens a square with a mine or all squares that do not contain a mine are

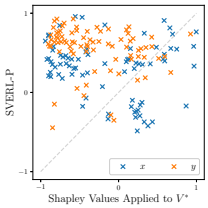


Figure 5: SVERL-P for every state of Gridworld-D compared to Shapley values applied to a value function. Shapley values were normalised to fall between -1 and 1. Each blue cross denotes the x feature for a particular state and each orange cross the y feature.

opened. There is only one reward signal: -20 whenever the agent reveals a mine. Therefore the highest return possible is 0. There is no incentive for the agent to complete a game in minimal time.

A.2 SVERL-P Comparisons

Here we demonstrate and discuss the key differences between SVERL-P and applying Shapley values to policies and the value function.

SVERL-P compared with Shapley values applied to value functions. The domains Gridworld-A and Tic-Tac-Toe were used in Section 3 to demonstrate that applying Shapley values to an agent's value function does not explain agent performance. In contrast, local SVERL-P contributions in these domains, shown in Figures 2a and 2b, match our intuitive understanding of the contribution of state features to performance.

As a result of purposely choosing simple, illustrative examples, the examples in these two domains used either a constant policy or a constant value function. MDPs with these particular properties are uncommon. Our arguments, however, are valid for any MDP. As an example, Figures 3, 4a and 4b show that, in all domains tested, SVERL-P gives different results than applying Shapley values to the value function. They include domains with varying policies and value functions. In Figure 5, we compare SVERL-P and Shapley values applied to V^* in every state of a randomly-constructed gridworld with 80 states (Gridworld-D). The results show a persistent difference between these two approaches.

SVERL-P compared with Shapley values applied to policies. In Section 3, we introduced Shapley values applied to an agent's policy. We argued that they provided insight which improved understanding of a decision but that further insight could be drawn by also considering the effect of state features on performance. We now illustrate our viewpoint by comparing local SVERL-P to Shapley values applied to a policy.

Consider Gridworld-C, shown in Figure 1c. In this domain, if no state feature is known, the agent cannot know the optimal action with certainty but it knows that (1) it is either North, East or West, and (2) North is more likely than East or West. In states 2 and 5, neither observing x nor observing y reveals the optimal action. We have no natural intuition on the importance of state features and must rely on Shapley values.

Shapley values applied to the optimal policy in every state are shown in Figure 6. For each state, the Shapley values are presented for the optimal action, a^*. In state 5, x contributes more than y to the probability of choosing the optimal action (N). One might assume that x is therefore more important than y for an agent to act optimally. However, this would be incorrect. The local SVERL-P contributions, shown in the top panel of Figure 6, reveal that in fact the x and y features contribute equally to performance. The reason for this difference is that, in state 5, x also contributes towards the likelihood of selecting the worst action (E). Similarly, in state 2, Shapley values applied to the policy show that both x and y contribute equally to the probability of selecting the optimal action (N). However, local SVERL-P contributions reveal that y actually contributes more than x to performance. In this state, observing x but not y increases the probability of selecting the worst action (W).

By applying Shapley values to policies without considering the consequence on performance, one would draw incorrect or incomplete conclusions about the importance of state features. By considering

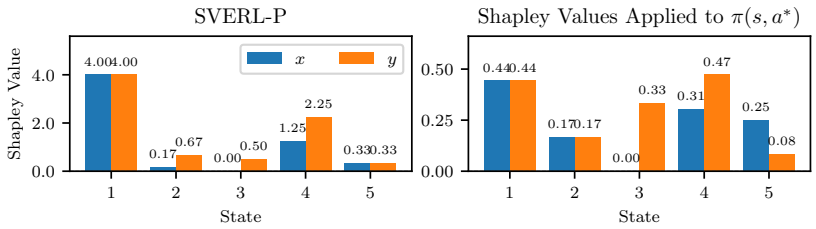


Figure 6: SVERL-P compared to Shapley values applied to a policy in Gridworld-C (Figure 1c). The plots show the Shapley values of the x and y state features for all states. SVERL-P gives the contribution of state features towards performance while Shapley values applied to a policy give the contributions of state features towards the likelihood of selecting the optimal action in each state.

the contribution of state features towards performance, SVERL-P provides additional insight into agent behaviour.

A.3 Extended Analysis in Minesweeper

In the minesweeper example of Figure 4a, SVERL-P contributions are negative for two unopened squares (M_1 and M_2) in the second state. The implication is that observing either state feature makes a negative contribution to the expected return. We hypothesise that, by becoming observable, these features increase the probability that the agent clicks on the corresponding squares. Such an action would reveal the underlying mine and terminate the game with a large negative reward.

In Section 6, we suggested that humans are likely to naturally over-interpret SVERL-P contributions, developing hypotheses that must be tested. This is one such example. The validity of our hypothesis can be tested by examining Shapley values applied to a policy that outputs action probabilities, introduced in Section 3. Figure 7 shows that the Shapley values for the probability of selecting each unopened feature are positive, showing that, on average, observing that a square is unopened positively contributes towards the probability of selecting it.

Note the non-negative SVERL-P contribution of M_1 in state 1 even though observing that a square is unopened increases the probability of opening it. On closer inspection, Figure 7 reveals that observing that square $(3, 2)$ is unopened increases the probability of opening square $(4, 2)$ (the optimal action) much more than it increases the probability of opening $(3, 2)$.

SVERL-P contributions revealed insight into *how* features contributed to performance but further analysis was required to investigate *why* features contributed to performance.

A.4 Computing Shapley Values

This work presented four applications of Shapley values in reinforcement learning, under the SVERL framework: Shapley values applied to value functions, Shapley values applied to policies, local SVERL-P and global SVERL-P. Each of the different Shapley values are computed using Equation (1), with their respective characteristic value functions computed using Equations (6) to (10) and (12). All of these characteristic value functions require the conditional limiting state occupancy distributions, $p^\pi(s'|s_C)$, for every $C \subset \mathcal{F}$. We calculate each $p^\pi(s'|s_C)$ using Bayes's rule:

$$p^\pi(s'|s_C) = \frac{p(s_C|s')p^\pi(s')}{p^\pi(s_C)} = \frac{p(s_C|s')p^\pi(s')}{\sum_{s' \in \mathcal{S}} p(s_C|s')p^\pi(s')}, \tag{14}$$

where the limiting state occupancy distribution $p^\pi(s')$ is approximated through interaction with the environment. Additionally, if s_C is a possible observation of s', then $p(s_C|s') = 1$, else $p(s_C|s') = 0$. For example, in Gridworld-B, $s_C = \{x = 1\}$ is a possible observation of $s' = \{x = 1, y = 3\}$, whereas $s_C = \{x = 2\}$ is not.

After computing the conditional limiting state occupancy distributions using Equation (14), the characteristic value functions for Shapley values applied to policies and Shapley values applied to value functions can be calculated directly using Equations (6) to (9). For the local and global SVERL-P characteristic values in Equations (10) and (12), first $\pi_C(a|s)$ must be computed using

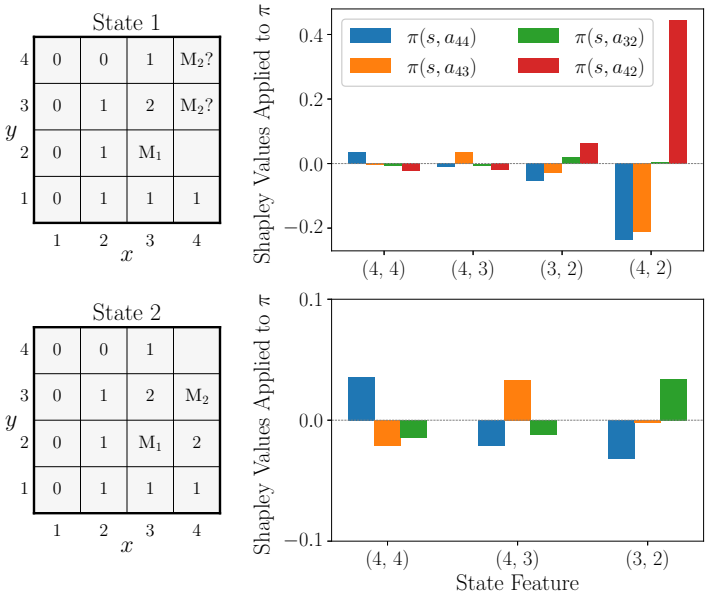


Figure 7: Shapley values applied to a policy in two states of Minesweeper. Action a_{xy} denotes the action that opens grid square (x, y). The plots show, for each available action, the Shapley values of the state features that correspond to unopened squares.

Equation (11). Then the characteristic values, which are expected returns, can be computed using any standard reinforcement learning algorithm. We used Monte Carlo roll outs.

A.5 Code

Code is available at `https://github.com/bath-reinforcement-learning-lab/SVERL_icml_2023`.