The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Local Explanations for Reinforcement Learning

Ronny Luss*, Amit Dhurandhar* and Miao Liu

IBM Research, Yorktown Heights, NY
rluss@us.ibm.com, adhuran@us.ibm.com, miao.liu1@ibm.com

Abstract

Many works in explainable AI have focused on explaining black-box classification models. Explaining deep reinforcement learning (RL) policies in a manner that could be understood by domain users has received much less attention. In this paper, we propose a novel perspective to understanding RL policies based on identifying important states from automatically learned meta-states. The key conceptual difference between our approach and many previous ones is that we form meta-states based on locality governed by the expert policy dynamics rather than based on similarity of actions, and that we do not assume any particular knowledge of the underlying topology of the state space. Theoretically, we show that our algorithm to find meta-states converges and the objective that selects important states from each meta-state is submodular leading to efficient high quality greedy selection. Experiments on four domains (four rooms, door-key, minipacman, and pong) and a carefully conducted user study illustrate that our perspective leads to better understanding of the policy. We conjecture that this is a result of our meta-states being more intuitive in that the corresponding important states are strong indicators of tractable intermediate goals that are easier for humans to interpret and follow.

1 Introduction

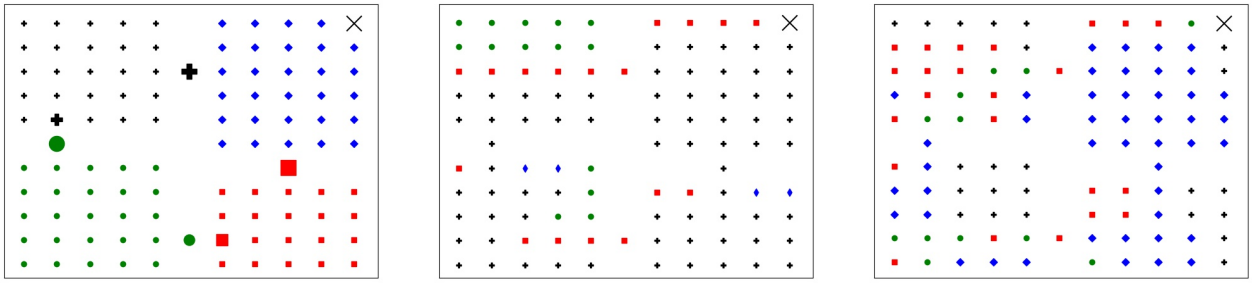
Deep reinforcement learning (RL) has seen stupendous success over the last decade with superhuman performance in games such as Go (Silver, Huang, and et al. 2016), Chess (Silver et al. 2018), and Atari benchmarks (Mnih, Kavukcuoglu, and et al. 2015). With increasing superior capabilities of automated (learning) systems, there is a strong push to understand the reasoning behind their decision making. One motivation is for (professional) humans to improve their performance in these games (Rensch 2021). An even deeper reason is for humans to be able to trust these systems if they are deployed in real life scenarios (Gunning 2017). The General Data Protection Regulation (Yannella and Kagan 2018) passed in Europe demands that explanations need to be provided for any automated decisions that affect humans. While various methods have been provided to explain classification models (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Lapuschkin et al. 2016; Dhurandhar et al. 2018) and be evaluated in an application-grounded manner (Doshi-Velez and Kim 2017; Dhurandhar et al. 2017), the exploration of different perspectives to explain RL policies has been limited and user study evaluations are rarely employed in this space.

In this paper, we provide a novel perspective to produce human understandable explanations with a task-oriented user study that evaluates which explanations help users predict the behavior of a policy better. Our approach involves two steps: 1) learning meta-states, i.e., clusters of states, based on the dynamics of the policy being explained, and 2) within each meta-state, identifying states that act as intermediate goals, which we refer to as *strategic states*. We call our method the Strategic State eXplanation (SSX) method. Such an approach has real-world applicability. Consider scenarios where businesses want to increase their loyalty base. Companies often train RL policies to recommend next-best actions in terms of promotions to offer. They try to maximize the *Lifetime Value* (Theocharous, Thomas, and Ghavamzadeh 2015). For such policies, SSX could identify strategic offers that lead to becoming loyalty customers based on a state space with features such as demographics, buying behavior, etc. Another example is robotics; consider the task of a robotic arm lifting a cup on a table which can be broken down to a sequence of stages, i.e., strategic states (subgoals) to aim for.

Contrary to the global nature of recent explainability works in RL (Topin and Veloso 2019; Sreedharan, Srivastava, and Kambhampati 2020; Amir and Amir 2018), our focus is on local explanations; given the current state, we explain the policy moving forward within a fixed distance from the current state. This key distinction lets us consider richer state spaces (i.e., with more features) because the locality restricts the size of the state space. It is also important to recognize the difference from bottlenecks (Menache, Mannor, and Shimkin 2002; Simsek and Barto 2004) which are *policy-independent* and learned by approximating the state space with randomly sampled trajectories; rather than help explain a policy, bottlenecks are used to *learn* efficient policies such as through hierarchical RL (Botvinick, Niv, and Barto 2008) or options frameworks (Ramesh, Tomar, and Ravindran 2019). Strategic states are rather learned with respect to a policy and identified without assuming access to the underlying topology.

An example of this for the Four Rooms game is seen in Figure 1a, where an agent moves through a grid with walls (represented by lack of a marker) looking for the goal state (upper

*Equal contribution



(a) (b) (c)

Figure 1: Illustrations of our SSX (a), VIPER (b), and abstract states used for compression (c) methods based on an expert policy for the Four Rooms game with neither having information about the underlying topology of the state space. Colors/Shapes denote different meta-states/clusters. The black **X** in the upper right is the goal state. SSX clusters the four rooms exactly with strategic states denoted by larger markers, where the biggest marker implies the priority strategic state. SSX explains that the expert policy will head towards the open doors in each room preferring the door that leads to the room with the goal state. VIPER clusters states by action (black/plus=up, green/circle=down, blue/diamond=left, red/square=right) based on the full (discrete) state space, rather than samples, since it is tractable here. The compressed state space in (c) is also a function of the experts (conditional) action distribution. Clusters in (b) and (c) are scattered making it challenging for a human to understand any policy over clusters.

right corner). Each position is a state and a meta-state is a cluster of possible positions (states sharing a color/marker). Within each meta-state, we identify certain states as *strategic states* (shown with larger markers), which are intermediate states that moving towards will allow the agent to move to another meta-state and get closer to the goal state, which is the final state that the agent wants to get to. In Figure 1a, each room is (roughly) identified as a meta-state by our method with the corresponding doors being the respective strategic states. Topology refers to the graph connecting states to one another; our method only has access to the knowledge of which states are connected (through the policy), whereas reinforcement learning algorithms might have access to properties of the topology, e.g., the ability to access similar states using successor representations (Machado et al. 2018). In Figure 1, the topology is a graph connecting the different positions in each room or the doors connecting two rooms.

A key conceptual difference between our approach and others is that other methods aggregate insight (i.e. reduce dimension) as a function of actions (Bastani, Pu, and Solar-Lezama 2018) or formulas derived over factors of the state space (Sreedharan, Srivastava, and Kambhampati 2020) to output a policy summary, whereas we aggregate based on locality of the states determined by the expert policy dynamics and further identify strategic states based on these dynamics. Other summarization methods simply output simulated trajectories deemed important (Amir and Amir 2018; Huber et al. 2021) as judged by whether or not the action taken at some state matters. We use the term *policy dynamics* to refer to state transitions and high probability paths. We use the term dynamics because this notion contrasts other methods that use actions to explain what to do in a state or to identify important states; strategic states are selected according to the trajectories that lead to them, and these trajectories are implicitly determined by the policy.

The example in Figure 1 exposes the global view of our explanations when the state space is small because local approximations of the state space are not needed. We show that this perspective leads to more understandable explanations; aggregating based on actions, while precise, are too granular a view where the popular idiom *can't see the forest for the trees* comes to mind. We conjecture that the improved understanding is due to our grouping of states being more intuitive with strategic states indicating tractable intermediate goals that are easier to follow. An example of this is again seen in Figures 1b and 1c, where grouping based on actions for interpretability or for efficiency leads to less intuitive results (note that Figure 1c replicates Figure 4b from (Abel et al. 2019)). This scenario is further discussed in Section 4, where yet other domains have large state spaces and require strategic states to explain local scenarios. As such, our main contributions are two-fold:

1. We offer a novel framework for understanding RL policies, which to the best of our knowledge, differs greatly from other methods in this space which create explanations based on similarity of actions rather than policy dynamics. We demonstrate on four domains of increasing difficulty.
2. We conduct a task-oriented user study to evaluate effectiveness of our method. Task-oriented evaluations are one of the most thorough ways of evaluating explanation methods (Doshi-Velez and Kim 2017; Lipton 2016; Dhurandhar et al. 2017) as they assess simulatability, yet to our knowledge, have rarely been used in the RL space.

2 Related Work

While a plethora of methods are proposed in XAI (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Lapuschkin et al. 2016; Dhurandhar et al. 2018), we focus on works related to RL explainability and state abstraction, as they are most relevant to our work, and distinguish between global and local explainability methods as done in (Burkart and Huber 2021). Namely, global methods are *model explanation approaches* whereas local methods are *instance explanation approaches*. It is important to note that various

global methods described below, e.g. decision trees such as (Bastani, Pu, and Solar-Lezama 2018), can be used to explain individual instances, however this does not apply to all global methods, e.g. (Amir and Amir 2018). While global methods can explain a model without passing individual instances, i.e., by analyzing the splits of a decision tree, local methods only explain a model's performance on individual instances.

In the spirit of (Burkart and Huber 2021), most global RL methods summarize a policy using some variation of state abstraction where the explanation uses aggregated state variables that group actions (Bastani, Pu, and Solar-Lezama 2018; Liu et al. 2021) using decision trees or state features (Topin and Veloso 2019) using importance measures, or such that an ordering of formulas based on features is adhered to (Sreedharan, Srivastava, and Kambhampati 2020). These approaches all intend to provide a global summary of the policy. (Liu et al. 2021) is most recent and can be viewed complementary as well; the idea of using latent representations to increase interpretability could be adapted in our framework when visualizing results. Other summaries output trajectories deemed important according to importance measures (Amir and Amir 2018; Huber et al. 2021) or through imitation learning (Lage et al. 2019), or train finite state representations to summarize a policy with an explainable model (Danesh et al. 2019, 2021). Visualization techniques combined with saliency have been used to either aggregate states and view the policy from a different perspective (Zahavy, Zrihem, and Mannor 2016) or create a trajectory of saliency maps (Greydanus et al. 2018). Other works try to find state abstractions or simplify the policy (Abel et al. 2019; Paul, Vanbaar, and Roy-Chowdhury 2019; Liang et al. 2016), which should not be confused with works seeking explainability. State abstraction in these works is used for compression so that simpler policies can be used; the compressed state space is not interpretable as seen in Figure 1c.

Turning towards local explanation methods, some works focus on self-explaining models (Mott et al. 2019) where the policy has soft attention and so can indicate which (local) factors it is basing its decision on at different points in the state space. (Yau, Russell, and Hadfield 2020) learns a *belief map* concurrently during training which is used to explain locally by predicting the future trajectory. Interestingly, there are works which suggest that attention mechanisms should not be considered as explanations (Jain and Wallace 2019). These directions focus on learning an inherently explainable model rather than explaining a given model. Other works use local explanation methods to explain reasons for a certain action in a particular state (Olson et al. 2021; Madumal et al. 2020). These are primarily contrastive where side information such as access to the causal graph may be assumed. Our approach is both methodologically and conceptually different.

There are also program synthesis-type methods (Verma et al. 2018; Inala et al. 2020) that learn syntactical programs representing policies, which while more structured in their form, are typically not amenable to lay users. Methods in safe RL try to uncover failure points of a policy (Rupprecht, Ibrahim, and Pal 2020) by generating critical states. Another use of critical states is to establish trust in a system (Huang et al. 2018). There is also explainability work in the Markov decision processes literature focusing on filling templates according to different criteria such as frequency of state occurrences or domain knowledge (Khan, Poupart, and Black 2009; Elizalde et al. 2009). An elaborate discussion of these and other methods can be found in (Alharin, Doan, and Sartipi 2020), all of which unequivocally are different from ours.

3 Method

We now describe our algorithm, the Strategic State eXplanation (SSX) method, which involves computing shortest paths between states, identifying meta-states, and selecting their corresponding strategic states. Recall that all paths discussed below are based on transitions dictated by an expert policy we want to explain; bottlenecks however, are identified from paths generated as random walks through the state space and are meant to help *learn* policies rather than explain them.

Notations: Let $\mathcal{S}$ define the full state space and $s \in \mathcal{S}$ be a state in the full state space. Denote the expert policy by $\pi_E(\cdot,\cdot) : (\mathcal{A},\mathcal{S}) \rightarrow \mathbb{R}$ where $\mathcal{A}$ is the action space. The notation $\pi_E \in \mathbb{R}^{|\mathcal{A}|\times|\mathcal{S}|}$ is a matrix where each column is a distribution of actions to take given a state (i.e., the policy is stochastic). We assume a transition function $f_E(\cdot,\cdot) : (\mathcal{S},\mathcal{S}) \rightarrow \mathbb{R}$ that defines the likelihood of moving between states in one jump by following the expert policy.

Let $\mathcal{S}_\phi = \{\Phi_1, ..., \Phi_k\}$ denote a meta-state space of cardinality k. Denote m strategic states of meta-state Φ by $G^\Phi = \{g_1^\Phi, ..., g_m^\Phi\}$ where $g_i^\Phi \in \mathcal{S}\ \forall i \in \{1, ..., m\}$.

Maximum likelihood (expert) paths: One criterion used below is that two states in the same meta-state should not be far from each other. The distance we consider is the most likely path from state s to state s' under π_E. Consider a fully connected, directed graph where the states are vertices and an edge from s to s' has weight $-\log f_E(s,s')$. By this definition, the shortest path is also the maximum likelihood path from s to s'. Denote by $\gamma(s,s')$ the value of this maximum likelihood path and $\Gamma \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ a matrix containing the values of these paths for all pairs of states in the state space.

Counts of Out-paths: Another criterion used below for assigning states to meta-states is that if state s lies on many of the paths between one meta-state Φ_i and all other meta-states, then s should be assigned the meta-state Φ_i, i.e., $s \in \Phi_i$. We define below the number of shortest paths leaving Φ_i that a fixed state s lies on. Denote $T(s,s')$ as the set of states that lie on the maximum likelihood path between s and s', i.e., the set of states that define $\gamma(s,s')$. Then $1[s \in T(s',s'')]$ is the indicator of whether state s lies on the maximum likelihood path between s' and s'', and we compute the count of the number of such paths for state s and meta-state Φ via

$$C(s,\Phi) = \sum_{s' \neq s, s' \in \Phi} \sum_{s'' \notin \Phi} 1[s \in T(s',s'')]. \qquad (1)$$

One may also consider the likelihood (rather than count) of out-paths by replacing the indicator in eq. (1) with $\gamma(s',s'')$. $C(s,\phi(s))$ can be computed for all $s \in \mathcal{S}$ in $O(|\mathcal{S}|^2)$ by iteratively checking if predecessors of shortest paths from each node to every other node lie in the same meta-state as the first node on the path. Approximating $C(s,\phi(s))$ (through sampling) can lead to significant computational savings while

maintaining stability of the selected strategic states. The computation of out-paths in equation (1) involves searching over all paths between states in each meta-state with those states in other meta-states. See Appendix[1] B where stability is illustrated when randomly sampling a fixed fraction of the states in other meta-states (second summation in equation (1)).

3.1 Learning Meta-States

We seek to learn meta-states that balance the criteria of having high likelihood paths within the meta-state and having many out-paths from states within the meta-state. It is important to distinguish our goals from more classic cluster methods that are solely state-based; such clusterings would be independent of the expert policy that we want to explain and hence could lead to states connected by low likelihood paths as per the expert policy being in the same meta-state. Our meta-states account for the expert policy by minimizing the following objective for a suitable representation of s, which in our case is the eigen-decomposition of the Laplacian of Γ:

$$\underset{\mathcal{S}_\phi}{\operatorname{argmin}} \sum_{\Phi \in \mathcal{S}_\phi} \sum_{s \in \Phi} \left[(s - c_\Phi)^2 - \eta C(s, \Phi) \right] \tag{2}$$

where c_Φ denotes the centroid of the meta-state Φ and $\eta > 0$ balances the trade-off between the criteria. Note that we are optimizing $\mathcal{S}_\phi$ over all possible sets of meta-states. Other representations for s and functions for the first term could be used; our choice is motivated from the fact that such formulations are nostalgic of spectral clustering (Shi and Malik 2000) which is known to partition by identifying well-connected components, something we strongly desire. This representation connects the explanation to the policy because the matrix Γ is determined by the policy and provides intuitions. Specifically, in problem (2) when $\eta \to 0$, the meta-states will tend to be equi-sized where the likelihood of meta-state transitions will be minimized leading to (approximate) optimization of an NCut objective (von Luxburg 2007). For larger η, the likelihood of meta-state transitions is still kept small (which is desirable), with a tendency towards having a few large meta-states. We found our method to be stable for $\eta \in (0, 5]$.

Our method for solving eq. (2) is given by algorithm 1 and can be viewed as a regularized version of spectral clustering. In addition to clustering a state with others that it is connected to, the regularization pushes a state to a cluster, even if there are only a few connections to the cluster, if the policy dictates that many paths starting in the cluster go through that state.

3.2 Identifying Strategic States

Next, strategic states are selected for each meta-state. Assume that $g_1^\Phi, ..., g_m^\Phi \in \mathcal{S}$ are m strategic states for a meta-state Φ that does not contain the target state. SSX finds strategic states by solving the following problem for some $\lambda > 0$:

$$G_\Phi^{(m)} = \underset{g_1^\Phi, ..., g_m^\Phi}{\operatorname{argmax}} \sum_{i=1}^{m} C(g_i^\Phi, \Phi) - \lambda \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \max\left(\gamma(g_i^\Phi, g_j^\Phi), \gamma(g_j^\Phi, g_i^\Phi)\right). \tag{3}$$

[1]Appendix can be found at https://arxiv.org/abs/2202.03597.

Algorithm 1: Meta-states $\mathsf{MS}(\mathcal{S}, \mathcal{A}, \pi_E, \Gamma, k, \epsilon_\phi, \eta)$

1) Get eigen representation of each state s from eigen decomposition of the Laplacian of Γ
2) Randomly assign states $s \in \mathcal{S}$ to a meta-state in $\mathcal{S}_\phi = \{\Phi_1, ..., \Phi_k\}$ and compute centroids $c_1, ..., c_k$ for meta-states
3) ξ^{cur} = current value of objective in eq. (2)
do
  4) $\xi^{\text{prev}} = \xi^{\text{cur}}$
  5) Reassign states s to the meta-states based on smallest value of $(s - c_\Phi)^2 - \eta C(s, \Phi)$
  6) Compute centroids $c_1, ..., c_k$ for meta-states based on current assignment
  7) ξ^{cur} = current value of objective in eq. (2)
while $|\xi^{cur} - \xi^{prev}| \geq \epsilon_\phi$;
Output: Meta-states $\{\Phi_1, ..., \Phi_k\}$

Algorithm 2: Strategic State function $\mathsf{SS}(\mathcal{S}_\phi, \Gamma, \epsilon_g)$. Finds Strategic States with Greedy Selection (w.l.o.g. assume meta-state Φ_k contains the goal state).

for $i = 1$ *to* $k - 1$ **do**
  1) Let $\xi^{\text{cur}} = 0$ and $G_{\Phi_i} = \emptyset$
  do
    2) $\xi^{\text{prev}} = \xi^{\text{cur}}$
    3) $G_{\Phi_i} = G_{\Phi_i} \cup g$ where g solves eq. (3) over states not in the set of strategic states G_{Φ_i}
    4) ξ^{cur} = evaluate eq. (3) with G_{Φ_i}
  while $|\xi^{cur} - \xi^{prev}| \geq \epsilon_g$;
end
5) G_{Φ_k} = g, where g is the expert policy's goal state
Output: Strategic states for each corresponding meta-state $\{G_{\Phi_1}, ..., G_{\Phi_k}\}$

The first term favors states that lie on many out-paths from the meta-state, while the second term favors states that are far from each other. The overall objective tries to pick states that explore different meta-states consistent with the expert policy, while balancing the selection of states to be diverse . The objective in eq. (3) is submodular as stated next (proof in Appendix A) and hence we employ greedy selection in algorithm 2. Note that for the meta-state that contains the target state, the target state itself is its only strategic state.

Proposition 1. *The objective to find strategic states in equation (3) is submodular.*

3.3 Strategic State eXplanation (SSX) Method

Our method is detailed as follows. First, the maximum likelihood path matrix Γ is computed. Then, algorithm 1 tries to find meta-states that are coherent w.r.t. the expert policy, in the sense that we group states into a meta-state if there is a high likelihood path between them. If many paths from states in a meta-state go through another state, then the state is biased to belong to this meta-state. Finally, algorithm 2 selects strategic states by optimizing a trade-off between being on many out-paths with having a diverse set of strategic states.

3.4 Scalability and Complexity

Given our general method, we now discuss important details for making our algorithm practical when applied to different domains. SSX is applied in Section 4 to games with state spaces ranging from small to exponential in size. SSX is straightforward for small state spaces as one can pass the full state space as input, however, neither finding meta-states nor strategic states would be tractable with an exponential state space. One approach could be to compress the state space using VAEs as in (Abel et al. 2019), but as shown in Figure 1c, interpretability of the state space can be lost as there is little control as to how states are grouped. Our approach is to use local approximations to the state space; given a starting position, SSX approximates the state space by the set of states within some $N > 0$ number of moves from the starting position. In this approach, Algorithms 1 and 2 are a function of N, i.e., increasing N increases the size of the approximate state space which is passed to both algorithms. One can contrast our approach of locally approximating the state space with that of VIPER (Bastani, Pu, and Solar-Lezama 2018) which uses full sample paths to train decision trees. While the number of states in such an approximation is M^N, where M is the number of possible agent actions, the actual number of states in a game such a pacman is much smaller in practice. Indeed, while pacman has 5 possible actions, growth of the state space in our approximation as N increases acts similar to a game with 2-3 actions per move because most states in the local approximation are duplicates due to both minipacman and the ghost going back and forth. See Figure 5 in Appendix B, where other practical considerations, including approximating $C(s, \Phi)$, tractability of Γ and the eigen decomposition of its Laplacian, are also discussed.

4 Experiments

This section illustrates the Strategic State eXplanation (SSX) method on three domains: four rooms, door-key, and minipacman. These domains represent different reinforcement learning (RL) regimes, namely, 1) non-adversarial RL with a small state space and tabular representation for the policy, 2) non-adversarial RL, and 3) adversarial RL, the latter two both with a large state space and a deep neural network for the policy. These examples illustrate how strategic states can aid in understanding RL policies. A fourth domain, pong, represents adversarial RL where the environment has no access to the adversary and is in Appendix C. Lack of access to the adversary means that the maximum likelihood path matrix Γ requires simulation. Experiments were performed with 1 GPU and up to 16 GB RAM. The number of strategic states was chosen such that additional strategic states increased the objective value by at least 10%. The number of meta-states was selected as would be done in practice, through cross-validation to satisfy human understanding. Experiments demonstrating stability of strategic states to changes in the initial state, i.e. robustness of SSX to the initial state, as well as how sensitive strategic states are to the size of the local approximation, using measures of stability and faithfulness, are in Appendix E. Details about environments are in Appendix F.

Four Rooms: The objective of Four Rooms is to move through a grid and get to the goal state (upper right corner). The lack of a marker in a position represents a wall. Our grid size is 11×11. The state space consists of the player's current position and the policy is learned as a tabular representation, since the state space is not large, using Value Iteration (Martino and Mostofsky 2016).

SSX is displayed in Figure 1a with settings that learn four meta-states. Clustering the states using algorithm 1 according to the policy dynamics (i.e. maximum likelihood path matrix Γ) results in an (almost) perfect clustering of states according to the rooms. Larger markers denote strategic states learned in each meta-state, with the larger strategic state in each room corresponding to the first strategic state found. Clearly either door in each room could lead to the goal state in the upper right corner (black X), but it is important to note that higher valued strategic states in the red and black rooms are those that lead directly to the blue room containing the goal state.

Figure 1b illustrates the results of VIPER (Bastani, Pu, and Solar-Lezama 2018). The explanation is illustrated using different colors per action which effectively offers decision tree rules. While an explanation based on rules can be informative in continuous state spaces (as demonstrated in (Bastani, Pu, and Solar-Lezama 2018)), such rules applied to a discrete state space as done here may lead to confusion, e.g., groups of reds states are split by black states in the lower left room and allow for an optimal policy but it is not clear how to describe the cluster of states in which to take each action. Figure 1c illustrates the difference between explainability and compression (Abel et al. 2019) where one wants to learn abstract states upon which a proxy policy replicating the expert policy can be efficiently learned on the full state space. The lack of interpretability of the abstract states is not of concern in that context.

Door-Key: Door-Key is another non-adversarial game, but differs from Four Rooms because the state space is exponential in the size of the board. The policy is learned as a convolutional neural network with three convolutional and two linear layers. In this game, one must navigate from one room through a door to the next room and find the goal location to get a reward. Policies are trained under two scenarios. In the first scenario, a key in the first room must be picked up and used to unlock the door before passing through. In the second scenario, the door is closed but unlocked, so one does not need to first pick up the key to open the door.

SSX is run with local approximations to the state space with the maximum number of steps set to 6 as discussed in Section 3.4. Results are shown in Figure 2. The state space is a 7×7 grid reflecting the forward facing perspective of the agent. Walls are light gray and empty space visible to the agent is dark gray. Grid positions blocked from view by walls are black. The scenes in Figure 2 are exactly what a user sees. To better understand why scenes do not appear easily connected, consider the first two states in the first row - the only difference is that the agent changed directions. When facing the wall, the agent's view only includes the three positions to the right and one position to the left. Positions on

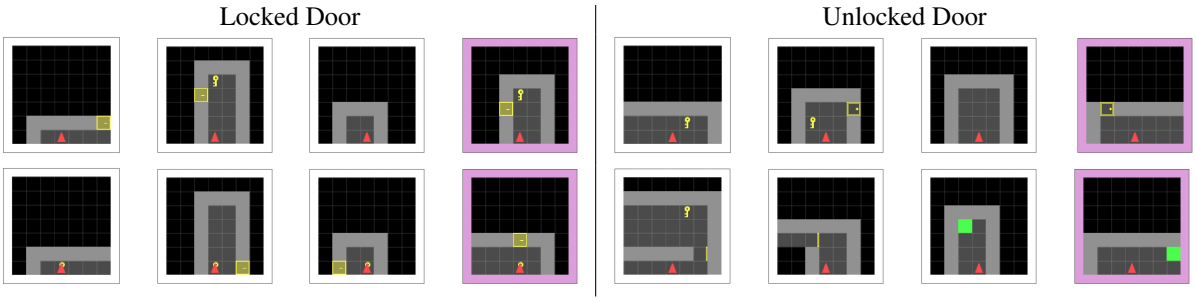


Figure 2: Illustration of our SSX method on Door-Key. Policies were trained on two different environments: Locked Door and Unlocked Door. Each row corresponds to a meta-state and strategic state (outlined in pink) from running SSX starting at a different number of moves into the same path (one path for completing the task in each of the two environments).

the other side of the wall are not visible to the agent, which is depicted as black. When the agent changed directions, many more positions in the room become visible to the agent.

In Figure 2, a sample path was generated using each policy. SSX was run at three different states along these paths, and one meta-state and corresponding strategic state (outlined in pink) from each SSX explanation is displayed. The two strategic states for the locked door environment correspond to the agent looking for the key (row 1) and getting the key (row 2). The two strategic states for the unlocked door environment correspond to the agent looking for the door (row 1) and making it through the door (row 2). An additional scenario can be found in Appendix G.

For intuition on how a human would use these explanations, consider the cluster in row 1 for the Locked Door. Comparing the first three states in the cluster to the strategic state, a human sees that the policy is suggesting to face the key and move closer to it. As this is a local explanation, it is limited by the initial state being explained as to how close one get to the key. The cluster in row 1 for the Unlocked Door shows that the policy at these states is to face the door. Facing the door within a certain distance seems how the policy breaks down the ultimate strategy. While one might wonder why the strategy is not to get closer to the door (e.g., move up from the second column), recall that the strategic state is explaining the policy and not human intuition.

Minipacman: Minipacman is a small version of the classic Pacman game. This game differs from Door-Key with the addition of an adversary - the ghost. The state space is again exponential in the size of the board and the policy is learned as a convolutional neural network with two convolutional and two linear layers. Two policies are trained with different scenarios. The first scenario, denoted EAT, is for minipacman to eat all the food with no reward for eating the ghost. The second scenario, denoted HUNT, is for minipacman to hunt the ghost with no reward for eating food.

SSX is again run with local approximations to the state space with the maximum number of steps set to 8. The state space is a 10×7 grid reflecting where food, pacman, a ghost, and the pill are located. Figure 3 displays one sample scenario under both the EAT and HUNT policies, with two meta-states and corresponding strategic states highlighted in pink. In order to interpret the figures, one needs to consider black vs blue pixels. The two strategic states of EAT Scenario 1 show pacman eating the food (row 1), i.e. columns 2/3 show blue pixels to the right of the pill meaning those pixels were not yet eaten before the strategic state is reached, but then avoiding the ghost and ignoring the pill (row 2). In HUNT Scenario 1, pacman is either directly moving towards the ghost after having eaten the pill (row 1) or heading away from the pill while the ghost is near it (row 2), i.e. going back to pixels already visited when waiting out the ghost near the pill. Additional scenarios and an experiment with a baseline motivated by (Amir and Amir 2018) appear in Appendix H and D, respectively.

5 User Study

We designed a user study to evaluate the utility of our approach relative to the more standard approach of explaining based on grouping actions. While SSX has thus far been used to give users local explanations about particular scenarios, we use it here to gain insight as to the general goal of a policy because the relevant explanations to compare with are global; as previously discussed, other local literature is about learning inherently explainable models rather than explaining a fixed model or learning contrastive explanations which should be used complementary to our methods. The global applicability of SSX can also be seen as another advantage. As with Four Rooms, we again compare with VIPER – a state-of-the-art explanation method for reinforcement learning policies – but use a visual output tailored for the discrete state space and label it Viper-D. We do not compare with methods that output trajectories (Amir and Amir 2018) as they require estimating Q-values to determine state importance; while this measure can successfully be used to select important trajectories that give users an idea of what a policy is doing, such important states are not necessarily good representatives of states that one should aim for, as is the goal of strategic states in SSX (see Appendix D for further discussion and related experiments). Among explanation methods, VIPER makes for the best comparison as it requires a similar amount of human analysis of the explanation (by observing states), and while meant for global explainability, also gives local intuitions, as opposed to other global methods. The utility of each approach is measured through a task posed to study participants: users must guess the intent of the expert policy based on provided explanations which are either output by SSX or VIPER. Such

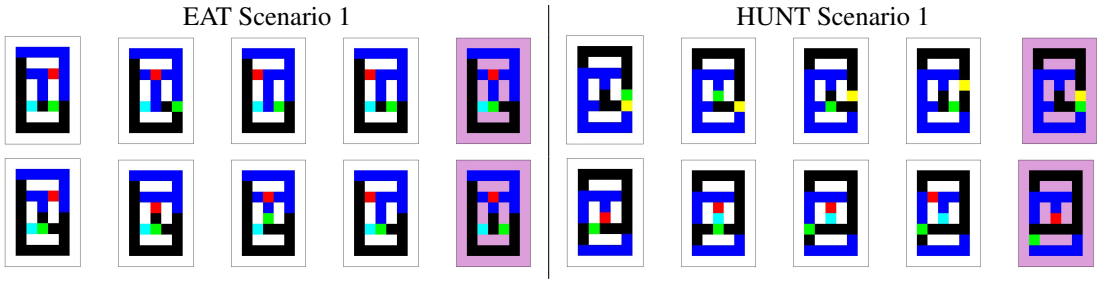


Figure 3: Illustration of our SSX method on minipacman. Two policies, EAT and HUNT, are displayed. Two clusters, one per row, are shown as part of the SSX result. The last board with pink background is a strategic state for each cluster. The color scheme is as follows: green = pacman, red = ghost, yellow = edible ghost, cyan = pill, blue = food, black = food eaten, white/pink=wall.

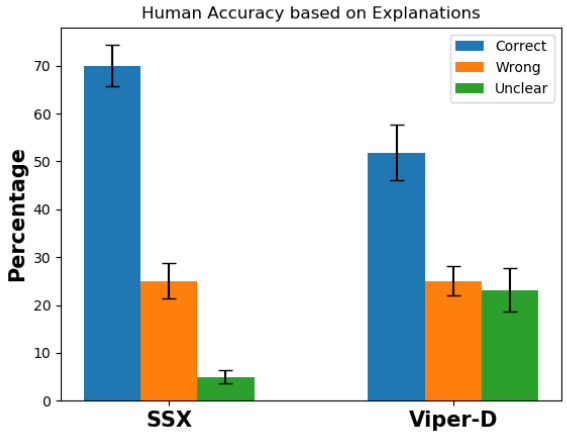


Figure 4: Above we see the percentage (human) accuracy in predicting if the expert policy is Eat or Hunt based on SSX and Viper-D. Performance difference is statistically significant (paired t-test p-value=0.01). Error bars are 1 std error.

a task oriented setup for evaluation is heavily encouraged in seminal works on XAI (Doshi-Velez and Kim 2017; Lipton 2016; Dhurandhar et al. 2017).

Setup: We use the minipacman framework with the EAT and HUNT policies trained above and each question shows either an SSX explanation or Viper-D explanation and asks the user "Which method is the explanation of type A (or B) explaining?" to which they must select from the choices Hunt, Eat, or Unclear. Methods are anonymized (as A or B) and questions for each explanation type are randomized. Ten questions (five from both the EAT and HUNT policies) are asked for each explanation type giving a total of twenty questions to each participant. At the end of the study, we ask users to rate each explanation type based on a 5-point Likert scale for four qualitative metrics - completeness, sufficiency, satisfaction and understandability - as has been done in previous studies on explainable RL (Madumal et al. 2020). For users to familiarize themselves with the two types of explanations we also provided training examples at the start of the survey, one for each type.

To be fair to VIPER explanations, rather than just displaying rules in text which may not be aesthetically pleasing, we created a visualization which not only displayed the (five) rules to the user, but also three boards, one each for pacman, the ghost, and the pill, highlighting their possible locations as output by the rule. This visualization, which we call Viper-D, is beyond the typical decision tree offered by VIPER and better renders explanations in our discrete setting. Screenshots of sample visualizations along with the instruction page and optional user feedback can be found in Appendix I.

The study was implemented using Google Forms and we received 37 responses from people with quantitative/technical backgrounds, but not necessarily AI experts. We removed 5 responses as they were likely due to users pressing the submit button multiple times as we twice received multiple answers within 30 seconds that were identical.

Observations: Figure 4 displays user accuracy on the task for method SSX and Viper-D. Users were able to better distinguish between the EAT and HUNT policies given explanations from SSX rather than Viper-D and the difference in percentage correct is statistically significant (paired t-test p-value is 0.01). Another interesting note is that less than 5% of SSX explanations were found to be Unclear whereas more than 25% of Viper-D explanations were labeled Unclear, meaning that, right or wrong, users felt more comfortable that they could extract information from SSX explanations. See Appendix I for results of qualitative questions to which users scored SSX higher than VIPER.

6 Discussion

We have seen in this work that our novel approach of identifying strategic states leads to more complete, satisfying and understandable explanations, while also conveying enough information needed to perform well on a task. Moreover, it applies to single agent as well as multi-agent adversarial games with large state spaces. Further insight could be distilled from our strategic states by taking the difference between the variables in some particular state and the corresponding strategic state and conveying cumulative actions an agent should take to reach those strategic states (viz. go 2 steps up and 3 steps right to reach a door in Four Rooms). This would cover some information conveyed by typical action-based explanations while possibly enjoying benefits of both perspectives. Other future directions include seeing if strategic states could be used as intermediate goals for efficiently training new policies and extending our idea to continuous state spaces.

References

Abel, D.; Arumugam, D.; Asadi, K.; Jinnai, Y.; Littman, M. L.; and Wong, L. L. 2019. State Abstraction as Compression in Apprenticeship Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Alharin, A.; Doan, T.-N.; and Sartipi, M. 2020. Reinforcement Learning Interpretation Methods: A Survey. *IEEE Access*, 8: 171058–171077.

Amir, D.; and Amir, O. 2018. HIGHLIGHTS: Summarizing Agent Behavior to People. In *AAMAS*.

Bastani, O.; Pu, Y.; and Solar-Lezama, A. 2018. Verifiable Reinforcement Learning via Policy Extraction. In *Advances in Neural Information Processing Systems*.

Botvinick, M. M.; Niv, Y.; and Barto, A. C. 2008. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognitive*, 3(113): 262–280.

Burkart, N.; and Huber, M. C. 2021. A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research*, 70: 245–317.

Danesh, M. H.; Koul, A.; Fern, A.; and Khorram, S. 2019. Learning Finite State Representations of Recurrent Policy Networks. In *Intl. Conference on Learning Representaitons*.

Danesh, M. H.; Koul, A.; Fern, A.; and Khorram, S. 2021. Re-understanding Finite State Representations of Recurrent Policy Networks. In *Intl. Conference on Machine Learning*.

Dhurandhar, A.; Chen, P.-Y.; Luss, R.; Tu, C.-C.; Ting, P.; Shanmugam, K.; and Das, P. 2018. Explanations based on the Missing: Towards Contrastive Explanations with Pertinent Negatives. In *Advances in Neural Information Processing Systems*, 592–603.

Dhurandhar, A.; Iyengar, V.; Luss, R.; and Shanmugam, K. 2017. TIP: Typifying the Interpretability of Procedures. arXiv:1706.02952.

Doshi-Velez, F.; and Kim, B. 2017. Towards A Rigorous Science of Interpretable Machine Learning. arXiv:1702.08608v2.

Elizalde, F.; Sucar, E.; Noguez, J.; and Reyes, A. 2009. Generating Explanations based on Markov Decision Processes. In *Mexican Intl. Conference on Artificial Intelligence*.

Greydanus, S.; Koul, A.; Dodge, J.; and Fern, A. 2018. Visualizing and Understanding Atari Agents. In *International Conference on Machine Learning*.

Gunning, D. 2017. Explainable Artificial Intelligence (XAI). In *Defense Advanced Research Projects Agency*.

Huang, S. H.; Bhatia, K.; Abbeel, P.; and Dragan, A. D. 2018. Establishing Appropriate Trust via Critical States. In *Intl. Conference on Intelligent Robots and Systems*.

Huber, T.; Weitz, K.; André, E.; and Amir, O. 2021. Local and Global Explanations of Agent Behavior: Integrating Strategy Summaries with Saliency Maps. In *arXiv:2101.12446*.

Inala, J. P.; Bastani, O.; Tavares, Z.; and Solar-Lezama, A. 2020. Synthesizing Programmatic Policies that Inductively Generalize. In *International Conference on Learning Representations*.

Jain, S.; and Wallace, B. C. 2019. Attention is not Explanation. In *NAACL*.

Khan, O. Z.; Poupart, P.; and Black, J. P. 2009. Minimal Sufficient Explanations for Factored Markov Decision Processes. In *Proceedings of the Nineteenth Intl. Conference on Automated Planning and Scheduling*.

Lage, I.; Lifschitz, D.; Doshi-Velez, F.; and Amir, O. 2019. Exploring Computational User Models for Agent Policy Summarization. In *Intl. Joint Conference on Artificial Intelligence*.

Lapuschkin, S.; Binder, A.; Montavon, G.; Müller, K.-R.; and Samek, W. 2016. The LRP Toolbox for Artificial Neural Networks. *Journal of Machine Learning Research*, 17(114): 1–5.

Liang, Y.; Machado, M. C.; Talvitie, E.; and Bowling, M. 2016. State of the Art Control of Atari Games Using Shallow Reinforcement Learning. In *AAMAS*.

Lipton, Z. C. 2016. The mythos of model interpretability. ICML Workshop on Human Interpretability in Machine Learning.

Liu, G.; Sun, X.; Schulte, O.; and Poupart, P. 2021. Learning Tree Interpretation from Object Representation for Deep Reinforcement Learning. *NeurIPS*.

Lundberg, S. M.; and Lee, S.-I. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, 4765–4774.

Machado, M. C.; Rosenbaum, C.; Guo, X.; Liu, M.; Tesauro, G.; and Campbell, M. 2018. Eigenoption Discovery Through the Deep Successor Representation. In *International Conference on Learning Representations (ICLR)*.

Madumal, P.; Miller, T.; Sonenberg, L.; and Vetere, F. 2020. Explainable Reinforcement Learning Through a Causal Lens. In *AAAI*.

Martino, A. D.; and Mostofsky, S. 2016. ABIDE. Http://fcon_1000.projects.nitrc.org/indi/abide/abide_I.html.

Menache, I.; Mannor, S.; and Shimkin, N. 2002. Q-Cut - Dynamic Discovery of Sub-goals in Reinforcement Learning. In *Proceedings of the European Conference on Machine Learning*.

Mnih, V.; Kavukcuoglu, S., K.; and et al., D. 2015. Human-level control through deep reinforcement learning. *Nature*, 518: 529–533.

Mott, A.; Zoran, D.; Chrzanowski, M.; Wierstra, D.; and Jimenez Rezende, D. 2019. Towards Interpretable Reinforcement Learning Using Attention Augmented Agents. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Olson, M. L.; Khanna, R.; Neal, L.; Li, F.; and Wong, W.-K. 2021. Counterfactual State Explanations for Reinforcement Learning Agents via Generative Deep Learning. In *arXiv:2101.12446*.

Paul, S.; Vanbaar, J.; and Roy-Chowdhury, A. 2019. Learning from Trajectories via Subgoal Discovery. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Ramesh, R.; Tomar, M.; and Ravindran, B. 2019. Successor Options: An Option Discovery Framework for Reinforcement Learning. In *Intl. Joint Conference on Artificial Intelligence*.

Rensch, D. 2021. Learn From The Best: AlphaZero. https://www.chess.com/lessons/play-like-alphazero. Accessed: 2020-06-20.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1135–1144.

Rupprecht, C.; Ibrahim, C.; and Pal, C. J. 2020. Finding and Visualizing Weaknesses of Deep Reinforcement Learning Agents. In *International Conference on Learning Representations*.

Shi, J.; and Malik, J. 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on PAMI*.

Silver, D.; Huang, M., A.; and et al., C. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529: 484–489.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; Lillicrap, T.; Simonyan, K.; and Hassabis, D. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Simsek, O.; and Barto, A. G. 2004. Hierarchical Interpretations for Neural Network Predictions. In *Intl. Conference on Machine Learning*.

Sreedharan, S.; Srivastava, S.; and Kambhampati, S. 2020. TLdR: Policy Summarization for Factored SSP Problems Using Temporal Abstractions. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Theocharous, G.; Thomas, P. S.; and Ghavamzadeh, M. 2015. Personalized Ad Recommendation Systems for Life-Time Value Optimization with Guarantees. In *IJCAI*.

Topin, N.; and Veloso, M. 2019. Generation of Policy-Level Explanations for Reinforcement Learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Verma, A.; Murali, V.; Singh, R.; Kohli, P.; and Chaudhuri, S. 2018. Programmatically Interpretable Reinforcement Learning. In *International Conference on Machine Learning*.

von Luxburg, U. 2007. A Tutorial of Spectral Clustering. In *Statistics and Computing*, volume 17.

Yannella, P. N.; and Kagan, O. 2018. Analysis: Article 29 Working Party Guidelines on Automated Decision Making Under GDPR. Accessed: 2020-06-20.

Yau, H.; Russell, C.; and Hadfield, S. 2020. What Did You Think Would Happen? Explaining Agent Behaviour through Intended Outcomes. In *Advances in Neural Information Processing Systems*.

Zahavy, T.; Zrihem, N. B.; and Mannor, S. 2016. Graying the black gox: Understanding DQNs. In *Intl. Conference on Machine Learning*.