Reinforcement Learning Explainability via Model Transforms (Student Abstract)

Mira Finkelstein[1], **Lucy Liu**[2], **Yoav Kolumbus**[1],
David C. Parkes[2], **Jeffrey S. Rosenshein**[1], **Sarah Keren**[3]
[1] The Hebrew University of Jerusalem, Benin School of Computer Science and Engineering
[2] Harvard University School of Engineering And Applied Sciences
[3] Technion - Israel Institute of Technology, Taub Faculty of Computer Science
mirafinkel@gmail.com, lliu@college.harvard.edu, yoav.kolumbus@mail.huji.ac.il,
parkes@eecs.harvard.edu, jeff@cs.huji.ac.il, sarahk@cs.technion.ac.il

Abstract

Understanding the emerging behaviors of reinforcement learning agents may be difficult because such agents are often trained using highly complex and expressive models. In recent years, most approaches developed for explaining agent behaviors rely on domain knowledge or on an analysis of the agent's learned policy. For some domains, relevant knowledge may not be available or may be insufficient for producing meaningful explanations. We suggest using formal model abstractions and transforms, previously used mainly for expediting the search for optimal policies, to automatically explain discrepancies that may arise between the behavior of an agent and the behavior that is anticipated by an observer. We formally define this problem of *Reinforcement Learning Policy Explanation* (RLPE), suggest a class of transforms which can be used for explaining emergent behaviors, and suggest methods for searching efficiently for an explanation. We demonstrate the approach on standard benchmarks.

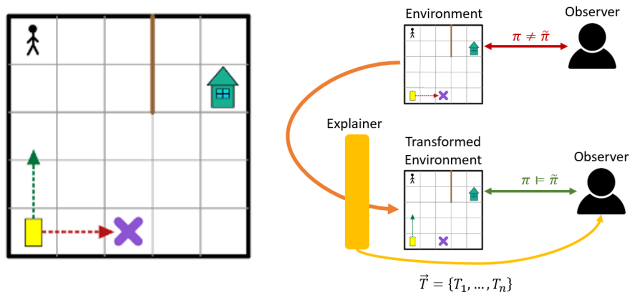

Figure 1: Reinforcement Learning Policy Explanation model (Right), and Taxi domain example (Left) where the agent's anticipated policy depicted by the green arrow, and the actual one by the red arrow.

Introduction

A significant limitation of AI models is the performance-transparency trade-off. As the inner workings of a model increase in complexity, it becomes more powerful, but the process through which it makes decisions becomes harder to understand. The *Explainable Artificial Intelligence* trend is particularly prevalent in *reinforcement learning* (RL) and *deep reinforcement learning* (DRL), where an agent autonomously learns how to operate in its environment. Therefore, in order to allow humans to collaborate with them effectively, it is important to develop methods for reasoning and explaining the agents' behaviors.

While there has been a variety of recent works on explainability of DRL, most of these methods do not exploit the full formal model of the environment, typically modeled as a Markov Decision Process (MDP), but instead focus on one element of the model or reasoning about the structure of the underlying network.

In this work, we suggest a novel approach to explainability that makes use of formal transforms of the underlying environment to automatically generate explanations of emerging behavior. As is common in most RL work, we assume the stochastic environment is modeled as a factored Markov Decision Process (MDP). This makes it possible to generate explanations without relying on domain knowledge, and instead, using transforms (abstractions) used so far in the literature to expedite RL and planning solutions. While for planning, the benefit of such transforms is in highlighting the features of the environment that influence the solution quality and time, we use them to isolate features of the environment that cause an agent to deviate from a behavior that is anticipated by an observer.

We consider an explainability setting, which we refer to as *Reinforcement Learning Policy Explanation* (RLPE), as comprised of three entities: an *actor*, who is an agent operating in the environment, an *observer*, who has some anticipation about the behavior of the actor that may not be met by the actual behavior, and an *explainer*, who wishes to clarify the discrepancy between the anticipated and actual behavior.

The contributions of this work are threefold. First, we present the first use of formal model transforms and abstractions to produce explanations of RL agent behaviors. Second, we present a formal definition of the Reinforcement Learning Policy Explanation (RLPE) problem and specify classes of state and action abstractions that can be used to produce meaningful explanations. Finally, we implement our approach and provide an empirical evaluation of this approach on a set of standard RL benchmarks.

MDP Transforms as Explanations

Definition 1 *A Reinforcement Learning Policy Explanation (RLPE) model is defined by the tuple $R = \langle M, A, \tilde{\pi}, T\rangle$, where:*

- *M is an MDP representing the environment,*
- *$A : \mathcal{M} \rightarrow \Pi$ the actor, which is associated with an RL algorithm that it uses to compute a policy $\pi \in \Pi$,*
- *$\tilde{\pi}$ is a set of possible* anticipated partial policies *defined over M an observer expects the actor to follow and,*
- *$T : \mathcal{M} \rightarrow \mathcal{M}$ is a finite set of transforms.*

The explanation is generated by searching for a set of MDP transforms such that the actor's behavior in the modified model aligns with the observer's expectations. If the transition from the original to the transformed environment is meaningful, the difference between the models can help the observer reason about the actor's behavior, thus representing an explanation. To account for different modifications that can be applied, we define a *transform* as any mapping $T : \mathcal{M} \mapsto \mathcal{M}$ that can be applied to an MDP to produce another MDP. The anticipated behavior of the observer is expressed as a partial policy describing what the observer expects the actor to do in a subset of its reachable states.

Clearly, for any two policies, state and action mappings can be applied to cause any policy to satisfy another policy. In order to produce valuable explanations, we need meaningful transforms that are applied to the underlying MDP, which change it in a way that highlights the elements of the model that cause unexpected behaviors in the actor's policy. Such transforms are suggested in our evaluation section.

Empirical Evaluation

We used three domains: one deterministic Taxi domain (TAXI) described in Fig. 1 and two stochastic domains: Frozen Lake (FROZEN) and Apple Picking (APPLE) [1].

Observer: We consider an observer which has partial knowledge about the environment. For all environments we assume that the observer anticipates that the actor follows an optimal policy in the observer's partial model. We compute the observers anticipated policy using a breadth-first search, using a determinization for stochastic domains.

Actor: We use the DQN, SARSA and CEM algorithms, from the keras-rl library[2] to represent the actor. Note that we could use any other algorithm since our framework is agnostic to the method used by the actor. We trained the agents for 600,000 episodes for all domains, each with 60 maximum steps per episode. Our experiments were run on a cluster using a mix of CPUs, with 4 cores using 16GB RAM.

Explainer: For the explainer, we use the following transforms: *most likely outcome* which is a special case of single-outcome determinization (Yoon, Fern, and Givan 2007), *precondition relaxation* as variant of the usage by Sreedharan et al. (2020) and *all outcome determinization* suggested by Keller and Eyerich (2011). We implemented all three transforms as atomic transforms that each modify a single action. We used three methods for searching for explanations, which are detailed in supplementary materials. The *BASE* Dijkstra search, *PRE-TRAIN* Dijkstra search in which we bootstrap the learning in the modified environment by using the policy on the original environment. We also prune transforms that do not directly effect the anticipated policy. The third method, *CLUSTER*, computes values of transform clusters.

[1]Further details described in the supplementary material

[2]*https://github.com/keras-rl/keras-rll*

	BASE		**PRE-TRAIN**		**CLUSTER**	
	sol	time	sol	time	sol	time
DQN						
TAXI	99%	5.5h	95%	3h	98%	1h
FROZEN	100%	2h	0%	2h	79%	0.5h
APPLE	98%	6h	94%	4h	93%	1.5h
SARSA						
TAXI	100%	5h	97%	2.5h	98%	1h
FROZEN	85%	3h	0%	3h	81%	0.5h
APPLE	99%	6.5h	96%	4h	93%	1.5h
CEM						
TAXI	95%	3h	93%	2.5h	91%	0.75h
FROZEN	50%	0.5h	50%	0.5h	74%	0.25h
APPLE	92%	4h	88%	3h	78%	1h

Table 1: Comparing method performance.

Results

Table 1 compares the performance of the different search methods, showing the ratio of instances that found the sequence of transforms that yields the anticipated policy (denoted as "sol") and the average computation time in half hour intervals (denoted as "time") for the instances that were solved by all approaches. The results show that, as expected, the exhaustive BASE method outperforms the two other methods in terms of ratio of instances solved. However, it consumes up to 5 times more computation time. In addition, although the PRE-TRAIN and CLUSTER methods prune transform sequences heuristically, the compromise in terms of solved instances is negligible. We assume that highlighting the different parts of the factored MDP model can be a meaningful explanation. Nevertheless, one of our next steps is to evaluate this assumption on human users.

There are various ways to extend our approach. First, while this work uses a restrictive satisfaction relation that required a complete match between the anticipated policy and the actor's behavior, it may be useful to use more flexible qualitative evaluation metrics for satisfaction. Secondly, we plan to extend this approach to explain behaviors of agents in multi-agent domains and to add transforms that are relevant to both collaborative and adversarial multi-agent settings.

References

Keller, T.; and Eyerich, P. 2011. A Polynomial All Outcome Determinization for Probabilistic Planning. In *ICAPS*.

Sreedharan, S.; Soni, U.; Verma, M.; Srivastava, S.; and Kambhampati, S. 2020. Bridging the Gap: Providing Post-Hoc Symbolic Explanations for Sequential Decision-Making Problems with Black Box Simulators. *CoRR*.

Yoon, S. W.; Fern, A.; and Givan, R. 2007. FF-Replan: A Baseline for Probabilistic Planning. In *ICAPS*.