Learning AI-System Capabilities under Stochasticity

Pulkit Verma*, **Rushang Karia***, **Gaurav Vipat, Anmol Gupta,** and **Siddharth Srivastava**
Autonomous Agents and Intelligent Robots Lab,
School of Computing and Augmented Intelligence,
Arizona State University, AZ, USA
{verma.pulkit, rushang.karia, gvipat, anmolgupta, siddharths}@asu.edu

Abstract

Learning interpretable generalizable models of sequential decision-making agents is essential for user-driven assessment as well as for continual agent-design processes in several AI applications. Discovering an agent's broad capabilities in terms of concepts a user understands, and summarizing them for a user is a comparatively new solution approach for agent assessment. Prior work on this topic focuses on deterministic settings, or settings where the names of the agent's capabilities are already known, or situations where the learning system has access to only passively collected data regarding the agent's behavior. These settings result in a limited scope and/or accuracy of the learned models. This paper presents an approach for discovering a black-box sequential decision-making agent's capabilities and interactively learning an interpretable model of the agent in stochastic settings. Our approach uses an initial set of observations to discover the agent's capabilities and a hierarchical querying process to learn a probability distribution of the discovered stochastic capabilities. Our evaluation demonstrates that our method learns lifted SDM models with complex capabilities.

1 Introduction

AI systems that can do sequential decision making like the household robots have seen increased deployments in the last few years. This invariably leads to situations where people who are not experts in AI end up interacting with these systems, and they are unsure of what these systems can (or cannot) do. In fact, the limits and capabilities of many AI systems are not always immediately clear even to the experts, as they may use black box policies, e.g., ATARI game-playing agents [Greydanus et al., 2018], text summarization tools [Paulus et al., 2018], mobile manipulators [Popov et al., 2017], etc. Additionally, it is common for non-experts to feel hesitant about asking questions related to new AI tools [Mou and Xu, 2017]. This often leads to a lack of knowledge about assessing the safe limits and capabilities of an AI system. The lack of understanding about the limits of an imperfect AI system can lead to unproductive usage or, in the worst-case scenario, serious accidents [Randazzo, 2018]. Such incidents can significantly limit the adoption and productivity of AI systems.

The related problem of learning generalizable relational models that describe an agent's interactions with its environment has been well-established as a critical problem in sequential decision-making (SDM). Learning such models can help in the continual design and improvement of SDM systems [Pasula et al., 2007, Walsh et al., 2009, Bryce et al., 2016, Juba and Stern, 2022] as well as in creating explanations of the behavior of SDM systems [Hayes and Shah, 2017, Sreedharan et al., 2018, Lage et al., 2019] and in the user-driven assessment of AI systems [Verma et al., 2021, Nayyar et al., 2022, Verma et al., 2022, 2023]. Several researchers have investigated methods for learning such models based on the available observations of agent behavior. However, they do not address the problem

*These authors contributed equally to this work

NeurIPS 2023 Workshop on Generalization in Planning (GenPlan 2023).

of *discovering* high-level user-interpretable capabilities that an agent can perform using arbitrary, internal behavior synthesis algorithms in a stochastic environment.

This work presents a novel approach for *discovering* capabilities of black-box sequential decision-making agents (SDMAs) acting in stochastic environments. The approach interactively learns a probabilistic model for black-box SDM agents' "capabilites". Such capabilities can represent high-level skills, such as driving an autonomous vehicle to reach a target destination. We use a popular, relational language PPDDL [Younes and Littman, 2004] to express the learned models in terms of preconditions denoting the conditions under which the agent can execute a particular capability and the resulting sets of effects, with probabilities that can occur as a result of such an execution. The resulting models can be used in model-based planning and approaches for explaining agent behavior or analyzing their limits and capabilities.

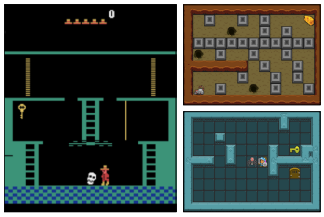

Figure 1: The environments used for the experiments: (a) shows the ATARI game Montezuma's Revenge; (b) shows the GVGAI domains Escape (top) and Zelda (bottom).

There has been a lot of progress in learning generalizable symbolic models from passively collected observations of agent behavior [Pasula et al., 2007, Martínez et al., 2016, Juba and Stern, 2022]; and by exploring the state space using simulators [Ng and Petrick, 2019, Chitnis et al., 2021]. However, in addition to needing the capabilities as inputs, passive learning approaches can also learn incorrect models as they do not have the ability to generate interventional or counterfactual data; exploration techniques can be sample inefficient because they currently don't take into account uncertainty and incompleteness in the model being learned to guide their exploration.

Our approach for learning such models is to autonomously generate queries in the form of instruction sequences, policies, and/or reachability objectives for the agent; the agent responds by executing these sequences in a simulator. Since the set of possible queries of this form is exponential in the state space, naïve approaches for enumerating and selecting useful queries based on information gain metrics are infeasible. We present novel algorithms for autonomously *synthesizing* queries that result in informative agent behavior. To summarize, our contributions are following:

- We introduce a novel approach that discovers the high-level capabilities of an SDMA in stochastic settings.
- We introduce a new approach to identify which transitions generated by the SDMA correspond to the same high-level capability.
- We introduce a framework that enables the assessment of different kind of AI systems like symbolic agents and game-playing agents.

We perform experiments on one ATARI-based simulator for the game Montezuma's Revenge, two General Video Game AI (GVGAI) [Perez-Liebana et al., 2016] domains shown in Fig. 1, and one benchmark symbolic domain.

Running Example In the rest of the paper, we use an example of the Zelda game shown in Fig. 1 (b). This SDMA setup uses a GVGAI game-playing agent featuring a protagonist player *Link* who must defeat the antagonist monster *Ganon*, and escape through the door using a key. Its primitive actions are keystrokes but they do not help convey the agent's capabilities because (i) they are too fine-grained, and (ii) they show the set of actions available to the AI system, although its true capabilities depend on its AI planning and learning algorithms. This work aims to discover these capabilities and learn their descriptions.

2 Related Work

The problem of learning probabilistic relational agent models from a given set of observations has been well studied. Jiménez et al. [2012] and Arora et al. [2018] present a comprehensive review of such approaches. We discuss the closest related research directions here.

Passive learning Several methods learn a probabilistic model of the agent and environment from a given set of agent executions. Many approaches [Zettlemoyer et al., 2005, Pasula et al., 2007] learn the models in the form of noisy deictic rules (NDRs) where an action can correspond to multiple

NDRs and also model noise. Mourão et al. [2012] learn such operators using action classifiers to predict the effects of an action. Martínez et al. [2015] leverage ILP to learn candidate models and use optimization to filter the best operators from each model. Rodrigues et al. [2011] learn non-deterministic models as a collection of rule sets and learn these rule sets incrementally. They take a bound on the number of rules as input. Juba and Stern [2022] provide a theoretical framework to learn safe probabilistic models with a range of probabilities for each probabilistic effect while assuming that each effect is atomic and independent of others. A common issue with such approaches is that they are susceptible to incorrect and sometimes inefficient model learning as they cannot control the input data used for learning or perform interventions on it.

Sampling of transitions Approaches like Sarathy et al. [2021], Jin et al. [2022], etc. learn the operator descriptions from exploring the state space but focus on deterministic models. A few reinforcement learning approaches have been explored for learning the relational probabilistic action model by exploring the state space using pre-determined criteria to generate better samples [Ng and Petrick, 2019]. Konidaris et al. [2018] explore learning PPDDL models for planning, but they aim to learn the high-level symbols needed to describe a set of input low-level options and use a huge number of samples to learn those symbols. Thon et al. [2011] learn a probability distribution over sequences of relational state descriptions using efficient sampling. Walsh et al. [2009] introduced a class of sample efficient exploration approaches for relational RL. GLIB [Chitnis et al., 2021] also learns probabilistic relational models using goal sampling as a heuristic for generating relevant data, whereas we use planning for this.

Learning symbolic models using physics simulators Zhang et al. [2018] learn symbolic transition models and use them effectively for planning using a set of input interpretable attributes. These attributes are interpretable and the set of actions are learned in the form of transitions using random walks in the environment. This needs extensive hand-coding as each of the states must manually be assigned all attributes that are present or true in that state. Also, the set of attributes is given as input which does not take into account user preferences. Kansky et al. [2017] learn symbolic models of simulators based on ATARI games by learning action effects by using conjunctions of binary input features. Some other approaches learn models using symbolic physics engine parameters and graph neural networks [Battaglia et al., 2016, Cranmer et al., 2020]. Through their description language, they can generalize over multiple tasks using the learned entities and operators. Agrawal et al. [2016] and Fragkiadaki et al. [2016] use convolutional neural networks to learn intuitive physical models of object interaction. Some methods create interpretable descriptions of reinforcement learning policies using trees [Liu et al., 2018] or specialized programming languages [Verma et al., 2018]. These approaches solve the orthogonal problem of learning the functionality of an agent that could help a user understand how an agent would solve a problem, whereas we focus on learning capabilities of the agent that could help a user understand and answer what type of problems it could solve.

Agent Assessment Verma et al. [2021] proposed the concept of assessing AI systems using hierarchical querying. This work assumed the agent to be working in a stationary setting, and learned a deterministic model of its low-level actions given the action names. Nayyar et al. [2022] use the same setup but does not assume the agent model and environment to be stationary and deterministic. Verma et al. [2023] learns a probabilistic model of the agent's capabilities but assumes access to the capability names. Verma et al. [2022] was the first method to discover the high-level capabilities of an agent and learning a description for them, but only works when environment dynamics are deterministic. We show that discovering capabilities for non-deterministic setting is not trivial and our approach discovers capabilities and learns their description accurately.

Automata learning Inspired from Angluin [1988], there are several active learning approaches [Aarts et al., 2012, Vaandrager, 2017] that learn automata to represent the system's model. These approaches assume access to a teacher (or an oracle) that can determine whether the automata are correct and provide a counterexample if it is incorrect.

Concept acquisition There is also a lot of ongoing and existing work on the orthogonal problem of learning predicates or concepts from user-provided examples or feedback [Amershi et al., 2009, Koh and Liang, 2017, Kim et al., 2018, Sreedharan et al., 2022] – this complements our research and can be used with our methods. Additionally, resolving the problem of obtaining user-interpretable predicates does not resolve the research problem that we focus on: deriving high-level descriptions of agent capabilities using those predicates.

3 Preliminaries

Abstraction Using an object-centric predicate representation induces an abstraction of environment states $\mathcal{X}$ to high-level logical states $\mathcal{S}$ expressible in predicate vocabulary $\mathcal{P}$. This abstraction can be formalized using a surjective function $f : \mathcal{X} \to \mathcal{S}$. E.g., in the case of the cafe server robot, the concrete state x may refer to $\langle x, y, z, r, p, \gamma \rangle$ tuples for all objects representing their positions in xyz coordinate with roll, pitch, and yaw values, respectively. On the other hand, the abstract state s corresponding to x will consist of truth values of all the predicates.

Propositional Concepts We are taking as input the set of propositional concepts that a user may associate with the task. We learn a classifier corresponding to each concept. One can use the concepts that are generated by the domain experts, or we could let the user interact with or observe the SDMA and then provide a possible set of concepts relevant for that task. E.g., the set of concepts used for the Zelda game in Fig. 1(b) can be `(at ?object ?location)`, `(alive ?monster)`, `(adjacent ?object1 ?object2)`, etc. Here, `?` precedes an argument that can be replaced by an object in the environment. E.g., `(adjacent link ganon)` means "link and ganon are adjacent to each other." As mentioned earlier, learning such concepts [Mao et al., 2022, Sreedharan et al., 2022, Das et al., 2023] is an interesting but orthogonal research direction, and it is not the focus of this work.

Each concept corresponds to a binary classifier, which detects whether the proposition is present or absent in a given internal state.

Probabilistic transition model Following the framework proposed by Mao et al. [2022], we assume that there exists an arbitrary latent space $\mathcal{S}$ expressible in $\mathcal{P}$. This induces an abstract transition model $\mathcal{T}' : \mathcal{S} \times \mathcal{C} \times \mathcal{S} \to [0, 1]$. This is done by converting each transition $\langle x, c, x' \rangle \in \mathcal{T}$ to $\langle s, c, s' \rangle \in \mathcal{T}'$ using predicate evaluators such that $f(x) = s$ and $f(x') = s'$. Now, $\mathcal{T}'$ can be expressed as model M that is a set of parameterized action (capability in our case) schema, where each $c \in \mathcal{C}$ is described as $c = \langle \textit{name(c)}, \textit{pre(c)}, \textit{eff(c)} \rangle$, where $\textit{name}(c) \in \mathcal{C}_N$ refers to name and arguments (parameters) of c; *pre(c)* refers to the preconditions of the capability c represented as a logical formula defined over $\mathcal{P}$ that must be true in a state to execute c; and *eff(c)* refers to the set of logical formulas over $\mathcal{P}$, each of which becomes true on executing c with an associated probability. The result of executing c for a model M is a state $c(s) = s'$ such that $P_M(s'|s, c) > 0$ and one (and only one) of the effects of c becomes true in s'. We also use $\langle s, c, s' \rangle$ triplet to refer to $c(s) = s'$. This representation is similar to the probabilistic planning domain definition language (PPDDL) [Younes and Littman, 2004], which can compactly describe the SDMA's capabilities.

```
(:capability c4
  :parameters (?player1 ?cell1
    ?monster1 ?cell2)
  :precondition (and
    (alive ?monster1)
    (at ?player1 ?cell1)
    (at ?monster1 ?cell2)
    (next_to ?monster1))
  :effect (probabilistic
    0.7 (and (clear ?cell2)
      (not (alive ?monster1))
      (not (at ?monster1 ?cell2))
      (not (next_to ?monster1)))
    0.2 (and (game-over)
      (not (at ?player1 ?cell1))
      (not (alive ?player1)))
    0.1 (and )))       #No-change
```

Figure 2: Sample PPDDL description for a capability of the Zelda agent.

4 Problem Setting

Sequential decision making agent (SDMA) This work focuses on sequential decision-making agents that operate in stochastic and fully observable environments. An SDMA can be represented as a 3-tuple $\langle \mathcal{X}, \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{X}$ is the environment state space that the SDMA operates in, $\mathcal{A}$ is the set of SDMA's actions (action names, e.g., "place object x at location y" or "move table x") that the SDMA can execute, and $\mathcal{T} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to [0, 1]$ is the stochastic black-box transition model determining the effects of SDMA's capabilities on the environment and the probabilities associated with them. Note that the semantics of $\mathcal{A}$ are not known to the user(s) and $\mathcal{X}$ may not be user-interpretable. The only input from the SDMA is the instruction set in the form of action names, represented as $\mathcal{A}_N$. This isn't a restricting assumption because the AI agents must reveal their instruction sets for usability.

Problem Given an SDMA $\langle \mathcal{X}, \mathcal{A}, \mathcal{T} \rangle$, a set of propositional concepts $\mathcal{P}$, along with their associated classifiers, discover the capabilities $\mathcal{C}$ of the SDMA, and learn a relational description of the abstract transition mode $\mathcal{T}'$ of the SDMA expressible using $\mathcal{P}$ and $\mathcal{C}$.

Assumptions We assume that (i) the data used to train the concept classifiers is available to us. In practice, this can be acquired using user studies like Kim et al. [2018], Sreedharan et al. [2022], etc.; (ii) the SDMA has a simulator available to answer the queries; (iii) the concept classifiers used to identify the state are noiseless.

5 Stochastic Capability Discovery and Assessment (SCaDA)

The stochastic capability discovery and assessment approach has three main components. The first component is to discover the candidate capabilities, the second component is to learn the description of the discovered capabilities, and the final component is to combine the candidate capabilities by identifying which capabilities are different probabilistic effects of each other and hence can be combined. We see each of these in detail.

5.1 Discovering Capabilities

Generating execution traces First, we collect a set of low-level execution traces E. An *execution trace* e is a sequence of states of the form $\langle s_0, s_1, \ldots, s_{n-1}, s_n \rangle$, such that $\forall i \in [1, n] \quad \exists a_i \in A$ $a_i(s_{i-1}) = s_i$. Note that the corresponding action a_i is not part of the input. These traces are obtained by giving the SDMA a set of tasks of the form $\langle s_I, s_G \rangle$, where $s_I, s_G \in \mathcal{S}$, and asking it to reach s_G from s_I using its internal policy. The actions it takes to complete the task and the intermediate states form the set of execution traces E. Partial capability descriptions A in the user's vocabulary are generated from E.

Generating candidate capabilities Given an execution trace $e = \langle s_0, s_1, \ldots, s_n \rangle \in E$, whenever $s_i \neq s_{i+1}$, we store the transition as a possible new capability $a_{s_i - s_{i+1}}$. For each new possible capability $a_{s_i - s_{i+1}}$, the states before and after executing these capabilities are stored as possible sets of preconditions and effects. We then combine sets of possible capabilities that cause similar state transitions. Here, any two transitions are similar if there is a one-to-one mapping of the predicates in both the transitions. In a manner similar to prior work by Stern and Juba [2017] and Verma et al. [2022], for each of these sets, we create a possible set of preconditions and effects by taking the intersection of the predicates that were true in the states before and after these capabilites were executed, respectively. This gives us partial capability descriptions of these high-level capabilities. The next step after this is to complete the partial descriptions.

5.2 Learning Capability Descriptions

Given the high level candidate capabilities, we use hierarchical querying mechanism to learn the description of each capability. SCaDA first initializes a model M^* with capabilities $\mathcal{C}$ learned in the first phase, and predicates $\mathcal{P}$. SCaDA generates an exhaustive set of hypotheses for each predicate $p \in \mathcal{P}$ at every location (precondition or effect in a capability). Given a location, the hypothesis space corresponding to a predicate will correspond to 3 transition models: one each corresponding to the three ways we can add the predicate in that location. We call these three hypotheses h_T, h_F, h_I, corresponding to adding p (true), *not(p)* (false), and not adding p (ignored), respectively at that location. SCaDA iterates over all combinations of locations and $\mathcal{P}$. For each pair, SCaDA creates 3 hypotheses h_T, h_F, and h_I as mentioned earlier. It then randomly takes two of these and generates a query q such that the response of the SDMA on that query can help prune out one of the hypotheses. Once the inconsistent hypothesis is pruned, it takes the remaining two hypotheses and repeats the same process.

Generating distinguishing queries SCaDA automates the generation of queries using search. As part of the algorithm, a model M is used to generate the three hypotheses corresponding to a specific predicate p and location l combination. So other than the predicate p at location l, the model representing the three hypotheses is exactly the same. A forward search is used to generate the policy simulation queries with two hypotheses h_i, h_j chosen randomly from h_T, h_F, and h_I. The forward search is initiated with an initial state $\langle s_0^i, s_0^j \rangle$ as the root of the search tree, where s_0^i and s_0^j are copies of the same state s_0 from which we are starting the search. The edges of the tree correspond to the capabilities with arguments replaced with objects in the environment. The nodes correspond to the two states resulting from applying the capability in the parent state according to the two hypotheses models. Now there will be an edge in the forward search tree with label c such that parent node is

$\langle s_0^i, s_0^j \rangle$ and child node is $\langle s_1^i, s_1^j \rangle$. The search process terminates when a node $\langle s^i, s^j \rangle$ is reached such that either the states s^i and s^j don't match, or the preconditions of the same capability were met in the state according to one of the hypotheses but not according to the other. Forward search can be slow depending on the number of capabilities and objects in the environment. So we use state-of-the-art planner PRP [Muise et al., 2012] used for search-based planning in non-deterministic environments. The output of this search is a policy π to reach a state where the two hypotheses, h_i and h_j differs. The query $\langle s_I, \pi, G \rangle$ resulting from this search is such that s_I is set to the initial state s_0, π is the output policy, G is the goal state where the hypotheses disagree. We next see how to use these queries to prune out the incorrect hypothesis.

Pruning the inconsistent hypothesis At this point, SCaDA already has a query such that the response to the query by the two hypotheses does not match. We next see how to prune out the hypothesis inconsistent with the SDMA. SCaDA poses the query generated earlier to the SDMA and gets its response. If the SDMA can successfully execute the policy, SCaDA matches the response of the two hypotheses with that of the SDMA and prunes out the hypothesis whose response does not match with that of the SDMA. If the SDMA cannot execute the policy, i.e., SDMA fails to execute some capability in the policy, then the hypotheses cannot be pruned directly. In such a case, a new initial state s_0 must be chosen to generate a new query starting from that initial state. This process to generate new queries for the same pair of hypotheses can take a long time, hence we preempt this issue by creating a pool of states $\mathbb{S}$ that can execute the capabilities using a directed exploration of the state space using partially learned models.

5.3 Combining Capabilities

Most of the existing approaches that identify which transitions correspond to the same capability use only the precondition [Silver et al., 2021, 2022], i.e., if the precondition of two candidate capabilities is same, then their effects are merged to create a new combined capability. This approach does not work well for settings when more than one action has same precondition. E.g., to throw a cup and to throw a cup on the wall and to keep a cup gently on table, the corresponding capabilities will have the same precondition that the cup needs to be in the hand. But these are clearly different capabilities. So to solve this problem, we ask the SDMA to repeat a capability execution multiple times, and only if while executing one capability the effect of another capability is seen and vice versa, we combine the capabilities. To ask the agent to execute the same capability, the agent must understand what transition the capability corresponds to. Hence, we maintain a list of transitions corresponding to each capability and ask the agent to repeat the transition instead of it to execute the capability.

Learning the capability parameters Once the preconditions and effects of a capability are known, learning their parameters is a trivial process. The sets of predicates in the precondition and effect are sorted in a particular order (we sorted them alphanumerically for this work), and then the parameters are sorted according to the order in which they appear in the sorted predicates without repetition.

5.4 Learning the probabilities

After SCaDA learns the non-deterministic model, to learn the probabilities of the learned effects it uses the transitions collected as part of responses to queries. This is done using Maximum Likelihood Estimation (MLE). For each triplet $\langle s, c, s' \rangle$ seen in the collected data, let *count_c* be the number of times a capability c is observed. Now, for each effect set, the probability of that effect set becoming true on executing that capability c is given as the number of times that effect is observed on executing c divided by *count_c*.

6 Empirical Evaluation

We implemented SCaDA in Python and performed an empirical evaluation on three benchmark domains using 5.0 GHz Intel i9 CPUs with 64 GB of RAM. We found that our approach learns a set of capabilities for the three game environments that we show in Fig. 1. We briefly describe the environments below:

Escape The Escape domain, as shown in Fig. 1(b)(top), consists of movable blocks, fixed holes, and cheese. The blocks can be pushed into the holes to clear out a path. When moving in the blocks

adjacent to the holes, the agent can end up in hole even when not-moving directly towards it. The game is finished when the player reaches the location with cheese.

Zelda The Zelda-like domain, as shown in Fig. 1(b)(bottom), consists of a key, a door that opens using that key, the antagonist player *Link*, and the protagonist monster *Ganon*. To win the game, Link must defeat Ganon, and then should use the key to open the door to escape. Link can move one cell at a time in the direction it is facing. If Link moves into the cell adjacent to the key, Link picks up the key by executing the keystroke `E` (special keystroke). The same keystroke is used to Defeat Ganon (stochastic action) when Link is facing Ganon and is in a cell adjacent to Ganon, and to escape when Link is in a cell adjacent to the door and facing it.

Montezuma's revenge The Montezuma's revenge game is shown in Fig. 1(a). The low level state is a RAM-based state, which is the internal game state represented by the game controller's 256-byte array. We used ten concepts similar to Sreedharan et al. [2022], which we believed would be relevant to the game, for each screen and collected positive and negative examples using automated scripts.

Since, we do not have a ground truth model to compare the correctness of the learned model, we also ran experiments on a symbolic SDMA called driver agent [Verma et al., 2023]. We explain it below:

Driver agent This SDM setup is implemented using SOTA stochastic planning system used in planning literature. This is motivated from *Tireworld* setup introduced in the probabilistic track of IPC 2004 [Younes et al., 2005]. It consists of a robot moving around multiple locations. The move action can cause it to get a flat-tire with some probability. Not all locations have the option to change tire, but if available, a change-tire action will fix the flat-tire with a 100% probability.

Results We measure the number of queries required to learn the capabilities for each set of environments.

As mentioned before, we used the driver agent to compare the correctness of the learned descriptions, and we note that the correct two capabilities were learned in all the case where sufficient initial execution traces were captured before starting SCaDA. This shows that SCaDA is able to learn the correct model of transitions of an SDMA if they are captured.

Environment	Number of queries
Escape	592
Zelda	528
Montezuma	849
Driver Agent	34

Table 1: The number of queries used for the 4 environments in the experiments.

7 Conclusion

In this work, we presented an approach for discovering the capabilities of an agent and to learning a probabilistic model of an agent in terms of discovered capabilities using interactive querying.

Limitations and Future Work The work even though learning an accurate model of the SDMA's capabilities cannot be directly tested for correctness for domains where a ground truth model is not available. We can alleviate this similar to Silver et al. [2022] by using the learned model for planning. Though still not fully accurate, this would point if the learned representation can be used for reasoning about solving the tasks by a user or not. We will also try to make the code bases compatible so as a direct comparison with Silver et al. [2022] can be made possible to evaluate the planning efficiency of the learned model as compared to the state of the art.

We also plan to remove the assumption that the classifiers have to be non-noisy to accommodate for the real world setups. The current assumption limits the implementation to scenarios where environment properties are so distinct that they can easily be detected using a classifier, and needed expert knowledge. In addition, the classifiers currently work for the RAM and image based representations. We want to extend this setup to have classifiers (or some other form of concept evaluators) work for embodied AI settings too.

Acknowledgments and Disclosure of Funding

We thank anonymous reviewers for their valuable feedback and suggestions. This work was supported by the ONR under grant N00014-21-1-2045.

References

Fides Aarts, Faranak Heidarian, Harco Kuppens, Petur Olsen, and Frits Vaandrager. Automata learning through counterexample guided abstraction refinement. In *International Symposium on Formal Methods*, 2012.

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to Poke by Poking: Experiential Learning of Intuitive Physics. In *Proc. NIPS*, 2016.

Saleema Amershi, James Fogarty, Ashish Kapoor, and Desney Tan. Overview Based Example Selection in End User Interactive Concept Learning. In *Proc. UIST*, 2009.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, apr 1988.

Ankuj Arora, Humbert Fiorino, Damien Pellier, Marc Métivier, and Sylvie Pesty. A review of learning planning action models. *Knowledge Engineering Review*, 33:E20, 2018.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction Networks for Learning about Objects, Relations and Physics. In *Proc. NIPS*, 2016.

Dan Bryce, J Benton, and Michael W Boldt. Maintaining evolving domain models. In *International Joint Conference on Artificial Intelligence*, 2016.

Rohan Chitnis, Tom Silver, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. GLIB: Efficient exploration for relational model-based reinforcement learning via goal-literal babbling. In *AAAI Conference on Artificial Intelligence*, 2021.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering Symbolic Models from Deep Learning with Inductive Biases. In *Proc. NeurIPS*, 2020.

Devleena Das, Sonia Chernova, and Been Kim. State2Explanation: Concept-based explanations to benefit agent learning and user understanding. In *Advances in Neural Information Processing Systems*, 2023.

Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning Visual Predictive Models of Physics for Playing Billiards. In *Proc. ICLR*, 2016.

Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and Understanding Atari Agents. In *Proc. ICML*, 2018.

Bradley Hayes and Julie A. Shah. Improving robot controller transparency through autonomous policy explanation. In *HRI*, 2017.

Sergio Jiménez, Tomás De La Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. A review of machine learning for automated planning. *Knowledge Engineering Review*, 27(4): 433–467, 2012.

Mu Jin, Zhihao Ma, Kebing Jin, Hankz H. Zhuo, Chen Chen, and Chao Yu. Creativity of AI: Automatic symbolic option discovery for facilitating deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2022.

Brendan Juba and Roni Stern. Learning probably approximately complete and safe action models for stochastic worlds. In *AAAI Conference on Artificial Intelligence*, 2022.

Ken Kansky, Tom Silver, David A. Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics. In *Proc. ICML*, 2017.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *International Conference on Machine Learning*, 2018.

Pang Wei Koh and Percy Liang. Understanding Black-Box Predictions via Influence Functions. In *Proc. ICML*, 2017.

George Konidaris, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research*, 61(1):215–289, January 2018.

Isaac Lage, Daphna Lifschitz, Finale Doshi-Velez, and Ofra Amir. Exploring computational user models for agent policy summarization. In *International Joint Conference on Artificial Intelligence*, 2019.

Guiliang Liu, Oliver Schulte, Wang Zhu, and Qingcan Li. Toward Interpretable Deep Reinforcement Learning with Linear Model U-Trees. In *Proc. ECML PKDD*, 2018.

Jiayuan Mao, Tomás Lozano-Pérez, Joshua B. Tenenbaum, and Leslie Pack Kaelbing. PDSketch: Integrated domain programming, learning, and planning. In *Advances in Neural Information Processing Systems*, 2022.

David Martínez, Tony Ribeiro, Katsumi Inoue, Guillem Alenyà, and Carme Torras. Learning probabilistic action models from interpretation transitions. In *International Conference on Logic Programming*, 2015.

David Martínez, Guillem Alenyà, Carme Torras, Tony Ribeiro, and Katsumi Inoue. Learning relational dynamics of stochastic domains for planning. In *International Conference on Automated Planning and Scheduling*, 2016.

Yi Mou and Kun Xu. The Media Inequality: Comparing the Initial Human-Human and Human-AI Social Interactions. *Computers in Human Behavior*, 72:432–440, 2017. ISSN 0747-5632.

Kira Mourão, Luke Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. Learning STRIPS operators from noisy and incomplete observations. In *Conference on Uncertainity in Artificial Intelligence*, 2012.

Christian Muise, Sheila McIlraith, and Christopher Beck. Improved non-deterministic planning by exploiting state relevance. In *International Conference on Automated Planning and Scheduling*, 2012.

Rashmeet Kaur Nayyar, Pulkit Verma, and Siddharth Srivastava. Differential assessment of black-box AI agents. In *AAAI Conference on Artificial Intelligence*, 2022.

Jun H. A. Ng and Ronald P. A. Petrick. Incremental learning of planning actions in model-based reinforcement learning. In *International Joint Conference on Artificial Intelligence*, 2019.

Hanna M. Pasula, Luke S. Zettlemoyer, and Leslie Pack Kaelbling. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, 29:309–352, 2007.

Romain Paulus, Caiming Xiong, and Richard Socher. A Deep Reinforced Model for Abstractive Summarization. In *Proc. ICML*, 2018.

Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, and Simon Lucas. General Video Game AI: Competition, Challenges and Opportunities. In *AAAI Conference on Artificial Intelligence*, 2016.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient Deep Reinforcement Learning for Dexterous Manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

Ryan Randazzo. What went wrong with Uber's Volvo in fatal crash? Experts shocked by technology failure. *The Arizona Republic*, March 2018.

Christophe Rodrigues, Pierre Gérard, and Céline Rouveirol. Incremental learning of relational action models in noisy environments. In *International Conference on Inductive Logic Programming*, 2011.

Vasanth Sarathy, Daniel Kasenberg, Shivam Goel, Jivko Sinapov, and Matthias Scheutz. SPOTTER: Extending symbolic planning operators through targeted reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, 2021.

Tom Silver, Rohan Chitnis, Joshua Tenenbaum, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

Tom Silver, Ashay Athalye, Joshua B Tenenbaum, Tomas Lozano-Perez, and Leslie Pack Kaelbling. Learning neuro-symbolic skills for bilevel planning. In *Proceedings of Conference on Robot Learning*, 2022.

Sarath Sreedharan, Siddharth Srivastava, and Subbarao Kambhampati. Hierarchical expertise level modeling for user specific contrastive explanations. In *International Joint Conference on Artificial Intelligence*, 2018.

Sarath Sreedharan, Utkarsh Soni, Mudit Verma, Siddharth Srivastava, and Subbarao Kambhampati. Bridging the gap: Providing post-hoc symbolic explanations for sequential decision-making problems with inscrutable representations. In *International Conference on Learning Representations*, 2022.

Roni Stern and Brendan Juba. Efficient, Safe, and Probably Approximately Complete Learning of Action Models. In *Proc. IJCAI*, 2017.

Ingo Thon, Niels Landwehr, and Luc De Raedt. Stochastic relational processes: Efficient inference and applications. *Machine Learning*, 82:239–272, 2011.

Frits Vaandrager. Model learning. *Communications of the ACM*, 60(2):86–95, 1 2017.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically Interpretable Reinforcement Learning. In *Proc. ICML*, 2018.

Pulkit Verma, Shashank Rao Marpally, and Siddharth Srivastava. Asking the right questions: Learning interpretable action models through query answering. In *AAAI Conference on Artificial Intelligence*, 2021.

Pulkit Verma, Shashank Rao Marpally, and Siddharth Srivastava. Discovering user-interpretable capabilities of black-box planning agents. In *International Conference on Principles of Knowledge Representation and Reasoning*, 2022.

Pulkit Verma, Rushang Karia, and Siddharth Srivastava. Autonomous capability assessment of sequential decision-making systems in stochastic settings. In *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Thomas J. Walsh, István Szita, Carlos Diuk, and Michael L. Littman. Exploring compact reinforcement-learning representations with linear regression. In *Conference on Uncertainity in Artificial Intelligence*, 2009.

Håkan L. S. Younes and Michael L. Littman. PPDDL 1.0: An extension to PDDL for expressing domains with probabilistic effects. Technical Report CMU-CS-04-167, Carnegie Mellon University, 2004.

Håkan L. S. Younes, Michael L. Littman, David Weissman, and John Asmuth. The first probabilistic track of the International Planning Competition. *Journal of Artificial Intelligence Research*, 24: 851–887, 2005.

Luke S. Zettlemoyer, Hanna M. Pasula, and Leslie Pack Kaelbling. Learning planning rules in noisy stochastic worlds. In *AAAI Conference on Artificial Intelligence*, 2005.

Amy Zhang, Sainbayar Sukhbaatar, Adam Lerer, Arthur Szlam, and Rob Fergus. Composable Planning with Attributes. In *Proc. ICML*, 2018.