Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)

Learning Portable Symbolic Representations

Steven James
School of Computer Science and Applied Mathematics
University of the Witwatersrand, South Africa
steven.james@wits.ac.za

Abstract

An open question in artificial intelligence is how to learn useful representations of the real world. One approach is to learn symbols, which represent the world and its contents, as well as models describing the effects on these symbols when interacting with the world. To date, however, research has investigated learning such representations for a single specific task. Our research focuses on approaches to learning these models in a domain-independent manner. We intend to use these symbolic models to build even higher levels of abstraction, creating a hierarchical representation which could be used to solve complex tasks. This would allow an agent to gather knowledge over the course of its lifetime, which could then be leveraged when faced with a new task, obviating the need to relearn a model every time a new unseen problem is encountered.

1 Introduction

A major goal of artificial intelligence is creating agents capable of acting effectively in a variety of complex environments. Robots, in particular, face the difficult task of generating behaviours while sensing and acting in a high-dimensional and continuous space. Planning at this low level is typically not feasible—the robot's innate action space involves directly actuating motors at a high frequency, but it would take thousands of such actuations to accomplish most useful goals. Abstraction, then, is clearly both desirable and necessary, but the question of how to link high-level reasoning with low-level sensing and control still remains open.

One approach is to represent the world using abstract symbols, with actions represented as operators that manipulate these symbols [Ghallab *et al.*, 2004]. Historically, constructing a symbolic representation of a real-world domain has proven difficult, often requiring substantial effort and expertise. However, recent work demonstrates how to learn a provably sound symbolic representation autonomously, given only the data obtained by executing the skills available to the agent [Konidaris *et al.*, 2018].

One major shortcoming of this framework is that these learned symbols are *directly tied to the task in which they were learned*. This results in a lack of generalisability—when an agent encounters a new task, it must relearn the appropriate symbolic representation from scratch. This is a data- and computation-intensive procedure involving clustering, probabilistic multi-class classification, and density estimation in high-dimensional spaces. We therefore propose a framework for learning *portable* symbolic representations, which consists of two phases. The first phase learns lifted symbolic rules which can be transferred between tasks, while the second combines these rules with problem-specific information to instantiate them for the current task.

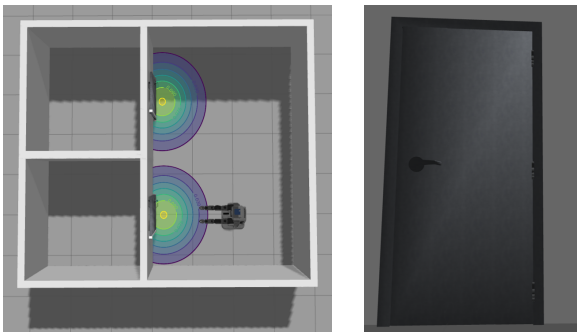
2 Agent-centric Symbols

Prior work defined symbols as names for distributions over low-level states [Konidaris *et al.*, 2018]. These symbols are thus directly tied to the task in which they were learned. Instead, we can adopt the approach of Konidaris *et al.* [2012], whereby tasks are related because they are faced by the same agent. Consider a robot equipped with various sensors that is required to perform a number of as yet unspecified tasks. The only aspect that remains constant across all these tasks is the presence of the robot, and more importantly its sensors, which map the state space to a portable observation space known as *agent space*.

Imagine a navigating robot learning an abstract representation of a map, where the robot possesses a motor skill for opening a door. For any specific map, the agent's state can be described by its xy-coordinates. If there are N doors in the environment, then the robot will learn N symbols, each of which is a distribution over coordinates specific to each door; when placed in a new map, it would have to learn everything from scratch. However, if we were to consider the state from the robot's perspective, we may find that the view when standing in front of each door looks identical (Figure 1). There is thus an opportunity to learn a single, generalisable symbol. Additionally, if a new unseen door is encountered, we need not learn a new distribution for it.

Under the assumption that there is a non-injective[1] observation function that maps states to observations, we proceed to learn symbols over agent-centric observations. For each skill, we learn two symbols: the *precondition*, which is a

[1]The ability to transfer relies on multiple states looking identical in agent space. If the observation function is injective, then transfer is no better than relearning each task anew.



(a) Illustration of symbols as distributions over xy-coordinates, illustrated with Gaussian distributions.

(b) A single symbol viewed from an agent-centric perspective.

Figure 1: Representations for the symbol `InFrontOfDoor` in different state spaces in a navigation task with two doors. When operating in xy-space (a), we require two symbols, each of which is a distribution over particular coordinates. If the doors were located at different positions, then the distributions would need to be relearned. However, an agent-centric definition (b), requires only one symbol that is independent of the door's location. We must still, however, determine *which* particular door it is.

classifier that determines when a skill can be executed, and the *effect*, which is a distribution over successor states. Planning, however, necessitates that these symbols be *grounded* in the current task, which requires a mapping from observations to states. The goal of learning is therefore no longer to estimate the effects of interacting with the world, but rather to construct this mapping—a much easier problem. We have already shown that this is achievable with certain problem-specific information, and have implemented the approach in a video game environment. Results demonstrate that as we encounter more tasks, we require fewer samples to build a sufficiently accurate model.

3 Future Work

3.1 Object-centric Symbols

In order to ground the agent-centric symbols, we use a single global parameter that refers to spatial regions of the state space, which may not scale with the number of skills. It may be possible to achieve a more natural representation by taking an object-centric view, similar to the OO-MDP framework [Diuk *et al.*, 2008], where each object belongs to a *class*, with objects in the same class sharing the same attributes.

By constructing symbolic representations in an object-centric manner, we can autonomously learn the class types in an environment based on the actions that can be performed upon them. For instance, given a skill `pickup`, we can learn that wooden blocks belong to the class `pickupable`. This will allow us to learn portable symbolic operators parameterised by object classes.

In order to ground these symbols, we require the properties of the individual objects in the current task—the effect of pushing a block, for instance, depends on the weight of the block. As a result, we no longer have a single spatial parameter (as in the agent-centric case), but rather multiple parameters that are tied to objects and their attributes. Such representations will be applicable to real-world robotic manipulation tasks, where we intend to empirically validate our approach.

3.2 Portable Hierarchies

Having developed a method for reducing the low-level state space to a symbolic representation, we can repeat the procedure to form even higher levels of abstraction. Konidaris [2016] proposes the *skill-symbol loop* for constructing propositional symbol hierarchies. Consider that a symbolic representation of a state can be represented by a binary vector whose elements specify whether each proposition is set to true or false. This, then, simply represents a new state space in a more abstract environment. If we have access to skills in this new abstraction, we can then learn symbols once more, creating higher and higher levels of abstraction.

In the agent-centric paradigm, we expect to learn high-level symbolic rules, which must then be grounded using problem-specific information from all layers of the abstraction hierarchy. In an object-centric view, we intend to learn class hierarchies. For example, we may learn initially that a block is both `pickupable` and `put-downable`. At a higher level, we can conclude that the block is also `stackable`, which involves picking up and putting down the block.

4 Conclusion

We propose a framework for autonomously learning portable symbols, and have already shown that the addition of problem-specific information to agent-centric symbols is sufficient for learning a sound representation for planning. We intend to extend this work to object-centric symbols, and then construct portable symbolic hierarchies. This will allow us to leverage experience in solving new unseen tasks—an important step towards creating adaptable, long-lived agents.

References

[Diuk *et al.*, 2008] Carlos Diuk, Andre Cohen, and Michael Littman. An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine learning*, 2008.

[Ghallab *et al.*, 2004] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[Konidaris *et al.*, 2012] George Konidaris, Ilya Scheidwasser, and Andrew Barto. Transfer in reinforcement learning via shared features. *Journal of Machine Learning Research*, 13(May):1333–1371, 2012.

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

[Konidaris, 2016] George Konidaris. Constructing abstraction hierarchies using a skill-symbol loop. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, volume 2016, pages 1648–1654, 2016.