LEARNING PORTABLE SYMBOLIC REPRESENTATIONS

Steven James[1], Benjamin Rosman[1,2] and George Konidaris[3]

[1]University of the Witwatersrand, Johannesburg, South Africa
[2]Council for Scientific and Industrial Research, Pretoria, South Africa
[3]Brown University, Providence RI 02912, USA

WITS BROWN RAIL LAB

Contribution

A framework for learning high-level symbolic operators from raw data that can be transferred between tasks

- **Operators are domain-independent**
- **Action schemata are provably sufficient for planning**

Introduction

Major AI challenges:
- Sense and act in real world.
- Plan at high/abstract level.
- Agents that can **generalise** across variety of tasks.

Previous work [1,2] can accomplish this given a set of high-level skills (options).

Results in propositional symbolic representation sufficient for planning, but which is directly tied to the low-level representation of the environment.
- **No opportunity for transfer**

We propose
1. First learning **domain-independent** symbolic **rules** from agent-centric observations.
2. Then learning **instantiations** of these rules for the current task faced.

This **lifts the representation** from propositional to predicate logic.

References

[1] G.D. Konidaris, L.P. Kaelbling and T. Lozano-Perez. Constructing Symbolic Representations for High-Level Planning. In *AAAI 2014*.
[2] G.D. Konidaris, L.P. Kaelbling, and T. Lozano-Perez. Symbol Acquisition for Probabilistic High-Level Planning. In *IJCAI 2015*.
[3] G.D. Konidaris and A.G. Barto. Building Portable Options: Skill Transfer in Reinforcement Learning. In *IJCAI 2007*.

Background

Must be able to symbolically represent the following:
1. **Precondition:** when can the skill be executed?
   - Probabilistic classification
2. **Image:** what is the effect of executing a skill?
   - Density estimation

We can do this if all skills are **subgoal**: the effect of each skill o does not depend on where it was initiated from, i.e. $\Pr(s'|s,o) = \Pr(s'|o)$.
We may need to **partition** a skill to ensure this property holds.

Agent is equipped with **sensors** that are always present, regardless of the task. This induces an observation space $\mathcal{D}$, known as ***agent space***. We refer to our original state space as ***problem space***. In other words, we have an observation function $\phi : \mathcal{S} \rightarrow \mathcal{D}$.

Because $\mathcal{D}$ remains the same for all environments the agent faces, we can use it to transfer [3]. We assume **agent-space options:** the initiation set, policy and termination condition of options are defined over $\mathcal{D}$.

Learning Portable Symbols

To learn portable symbols, ϕ is necessarily *not* injective, since we rely on distinct states in problem space **looking the same** in agent space.

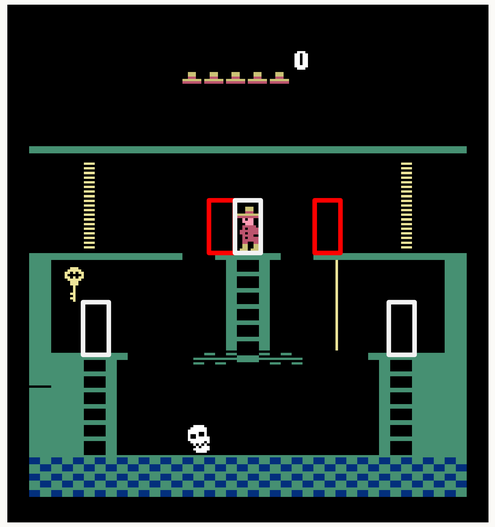
This is an issue if the **goal** is in **problem space**. We need some extra information to overcome this. We can get this by **partitioning** the agent-space skills in **problem space.**

Definition: The partitioning function $\psi_{\mathcal{S}} : \mathcal{O} \rightarrow \mathcal{P}(\mathbb{N})$ partitions each agent-space option so that the subgoal property holds in problem space. Every partition number is unique.

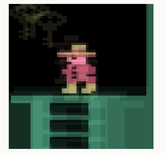
Theorem: The ability to represent the preconditions and image operator of each option in agent space, together with the partition function, is sufficient for determining the probability of being able to execute any probabilistic plan.

Procedure:
1. Learn symbols using observations → portable rules.
2. For a given task, partition options using $\psi_{\mathcal{S}}$.
3. Learn a mapping between partition numbers from data.
4. Partition numbers become parameters to portable rules.

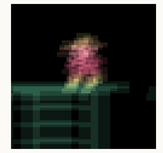
Experiments

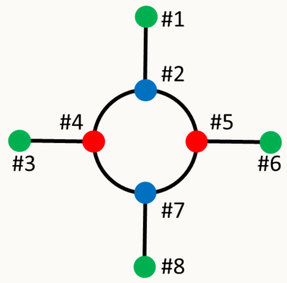


Descend-Ladder

Precondition	Image
	

Jump-Left-Gap

Precondition	Image
	



Portable Rules

Skill	Precondition	Image
Clockwise1	Blue	Red
Clockwise2	Red	Blue
Outward	Blue or Red	Green
Inward	Green	Red **w.p. 0.5** Blue **w.p. 0.5**

Instantiated Rules

Clockwise1(X):
Precondition: Blue(X)
Effect: $\neg$ Blue(X) $\wedge$ Red($f(X)$)

Inward(X):
Precondition: Green(X)
Effect: $\neg$ Green(X) $\wedge \begin{cases} \text{Blue}(g(X)) \text{ if } X = 8 \vee X = 1 \\ \text{Red}(h(X)) \text{ if } X = 3 \vee X = 6 \end{cases}$

Future Work

1. Apply to complex domain.
2. Demonstrate advantage of transfer over a number of tasks.