Learning Abstractions by Transferring Abstract Policies to Grounded State Spaces

Lawson L. S. Wong
Department of Computer Science, Brown University
Providence, RI 02912, USA
lsw@brown.edu

Abstract

Learning from demonstration is an effective paradigm to teach specific tasks to robots. However, such demonstrations often have to be performed on the robot, which is both time-consuming and often still requires expert knowledge (e.g., kinesthetically controlling the joints). It is often easier to specify tasks at a high level of abstraction, and let the robot figure out the grounding to the robot/agent space. We consider how to learn such a mapping. In particular, we consider the task of learning to navigate on a mobile robot given only an abstraction of the path and potential landmarks. We cast this as a learning problem between abstract and robot (grounded) state spaces and illustrate how this works in several cases. Through these cases, we see that the "abstract navigation" task touches on many interesting issues related to abstraction, and suggest avenues for further investigation.

Introduction

To tackle the high-dimensional complexity of the world and long-horizon nature of complex tasks, agents need *abstraction*, the act of compressing both state and time in service of certain goals. Much of artificial intelligence has been devoted to manually endowing agents with abstractions, such as via symbols (state abstraction) (Dietterich 2000; Konidaris, Kaelbling, and Lozano-Pérez 2018) and subtasks/options (temporal abstraction) (Sutton, Precup, and Singh 1999). However, agents that operate in a continual and lifelong setting will eventually encounter conditions unforeseen to the designer, and must come up with its own abstractions. Existing work in learning abstractions, most notably in reinforcement learning, typically require much experience within the domain, and arguably have not achieved widespread success. Indeed, one of the challenging aspects of abstraction is that in the time it takes to induce an abstraction and learn how to use it effectively, the specific ground / non-abstract task could already have been solved.

In contrast, humans use abstractions very effectively. For example, when provided a 2-D map of a new location (e.g., Figure 1), people can typically follow the map to reach a desired destination on the first try, without requiring the numerous episodes of trial and error that reinforcement learners require. This feat is even more remarkable when considering that the real-world looks nothing like the 2-D map: it is 3-D, is perceived from a first-person perspective (instead of bird's-eye for maps), and contains many more objects and other distractors compared to the map itself. Even so, when encountering these completely new percepts and 'states', people can follow where they are on the map and navigate as desired. Humans have mastered the abstraction of 2-D maps: from the current *ground* state in the real world, they are able to find the corresponding *abstract* state as a 2-D point on the map, determine the appropriate next *abstract* action within the abstract world, and then *ground* this action into physical motion. Furthermore, humans have mastered the entire *class* of such 2-D map abstractions; given a *new* instance of the abstraction (e.g, a map of a new place), humans can immediately perform the necessary grounding.

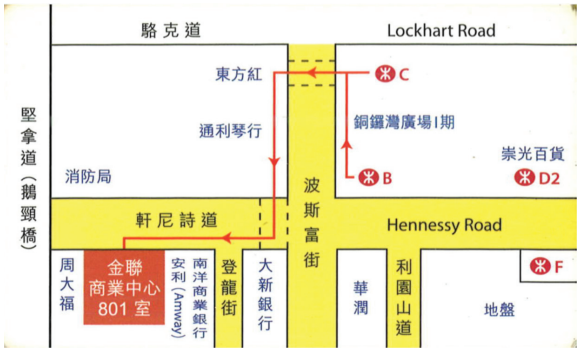

Figure 1: Humans can navigate in new places by using abstract 2-D maps, such as by following the walking directions depicted by the red arrows in the map above, which direct a person to exit a certain subway exit and cross two roads to reach an office (red square in the bottom left). They are able to take *abstract* policy-related knowledge encoded in the map, and *ground* the relevant actions in the real world. If robotic agents can learn to use existing abstractions, not only will they be easier to instruct by humans, they may even be able to produce abstract and interpretable knowledge.

We first formalize the notion of abstraction, then frame the problem of learning how to use existing abstractions as a fully supervised, learning-from-demonstration problem. For the "abstract navigation" task described above, we consider several classes of possible abstractions, some of which are

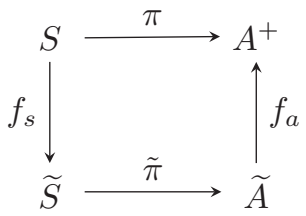

Figure 2: Abstraction diagram.

easy to learn, whereas others remain unsolved. Finally, we discuss directions of ongoing and future investigation.

Related Work

The general problem setup has ties to transfer learning (Taylor and Stone 2009) and learning from demonstration (Argall et al. 2009). Cobo et al. (Cobo et al. 2014) also explored learning abstractions from demonstrations, using an approach based on feature selection and task decomposition.

The formulation of abstractions in this work is inspired by the pioneering work of Ravindran (Ravindran 2004) and subsequent work by Abel et al. (Abel, Hershkowitz, and Littman 2016). Both lines of work analyze theoretical properties of abstraction in reinforcement learning.

Recently, there has been work on navigation using abstract 2-D maps such as hand-sketched maps (Boniardi et al. 2015; 2016), floor plans (Gao et al. 2017), and mazes (Brunner et al. 2018). However, most of these approaches are specific to 2-D robot/agent navigation.

Model and Problem Formulation

The agent operates in the grounded state space S and action space A. The objective is to determine a plan or policy $\pi : S \rightarrow A$ that achieves some given task in the world. The premise of this work is that we are given an abstract solution for the task, such as a route to follow on an abstract 2-D map that reaches a desired goal location. Formally, we are given an abstraction in abstract state and action spaces $\widetilde{S}, \widetilde{A}$, as well as an abstract policy $\tilde{\pi} : \widetilde{S} \rightarrow \widetilde{A}$.

The abstract policy is a solution for the (grounded) task if there exist *abstraction functions* $f_s : S \rightarrow \widetilde{S}$ and $f_a : \widetilde{A} \rightarrow A^+$ that can produce a grounded policy according to the diagram in Figure 2. In particular, the requirement is:

$$\pi = f_a \circ \tilde{\pi} \circ f_s \tag{1}$$

To find the next ground action(s), we first lift the ground state to the abstract state using f_s, apply the given abstract policy $\tilde{\pi}$, then ground the resulting abstract action using f_a to an executable primitive action (or action sequence, if there is temporal abstraction). If $\tilde{\pi}$ is an abstract solution for the task, then repeatedly applying this procedure should result in the agent reaching the goal in its grounded space.

Our goal is to *learn* the abstraction functions f_s and f_a, such that when presented with a new instance of the abstraction class (e.g., a 2-D map of a new location), the agent can follow the given abstract solution via Equation 1, i.e., transfer an abstract policy to the agent's grounded state space.

To learn the abstraction functions, we need training data. We consider the simplest setting, where paired trajectories in both ground and abstract spaces are provided. This is a fully-supervised, learning-from-demonstration setting, where the agent is shown grounded solutions to various task instances (e.g., by guiding it through the real world), together with annotated abstract solutions to the same problems (e.g., by drawing the route on the 2-D map).

Abstract Navigation

We consider several instances of a problem where the task is to follow a specified path, given in an abstract space. The grounded state space in all these cases is the state of the robotic agent, which includes highly relevant state dimensions such as odometry (noisy estimate of location relative to its starting position), moderately relevant features such as detected landmarks, and irrelevant features such as its arms' joint angles (if it has arms) or its battery level.

Isometric path

In the simplest case, the abstract path is given as a 2-D trajectory that accurately preserves relative lengths and angles, except possibly in a different global coordinate frame and scale. (This would be the case if the path was specified in most popular web mapping services such as Google Maps.) If the abstract path is also annotated at each point with the appropriate ground action, which could also be easily inferred from an isometric 2-D solution trajectory, then f_a can be assumed to be the identity function. The paired trajectories during training give corresponding pairs $(s, \tilde{s})$ of high-dimensional ground states and 2-D abstract states respectively. Learning f_s then becomes a multi-label linear regression problem (mapping s to $\tilde{s}$), since the ground and abstract states are related via an affine transformation (in the case of an isometric abstract path). In simulation, this method alone is highly effective at ignoring irrelevant features in the ground state s and handling zero-mean additive noise.

For abstract paths that are not perfect isometries, we need to learn non-linear regression functions. This is still strictly within the realm of supervised machine learning, for which many approaches exist to learn non-linear f_s functions.

The issue of orientation

The previous case provided a way to accurately find the abstract (x, y) location on the provided abstract path. However, the first problem one encounters when implementing the strategy on a point robot is orientation: if the robot is not facing in the same direction as the path intended, then following the abstract policy $\tilde{\pi}$ causes the robot to deviate from the path. The main issue is that the abstraction is insufficient to distinguish between the canonical path-following orientation from other states sharing the same abstract (x, y).

There are several potential ways to fix this. The simplest is to expand the abstract space to incorporate orientation θ as well; however, this requires a more complicated abstract policy to be specified. Alternatively, the burden may be placed

on the agent, by formulating each step of the path-following as a subtask (instead of a primitive action), where the subgoal is to return to a canonical orientation. The canonical orientation can be learned during training, or may be required to be the initial heading of the robot.

More generally, incomplete abstractions are likely to be encountered, and it would be useful to detect them and make local corrections, such as by inserting subgoals. This points to one argument for learning both ground and abstract transition models, T and $\widetilde{T}$ respectively: an incomplete abstraction will not generally be able to enforce one-step consistency between $f_s \circ T$ and $\widetilde{T} \circ f_s$. Thus transition models enable error detection in abstraction.

Landmarks

In typical maps, even in the case that the map is an isometry, there are additional features such as street names, room numbers, and other iconic elements such as architecturally distinct buildings. For example, in Figure 1, various street names (black font) and store names (blue font) are given near their respective locations. As humans, our sense of odometry is likely worse than mobile robots, so we must rely on these highly distinguishable landmark cues for robustness. If the robot is provided with detectors that allow it to detect landmark features, then these detections can simply be incorporated as additional ground state dimensions, and we can proceed to learn f_s from demonstrations via nonlinear regression. In simulation, we considered landmarks in the form of 'color patches' encountered in local regions of the world; if the color is confined to a unique region in the abstract space, these landmarks are highly informative and can correct for otherwise inaccurate geometric mappings.

Topological path

In the previous case, landmarks provide information that is redundant with the geometric abstract map. Hence it is possible to remove the geometric aspects of the abstraction and simply retain the topological information provided by landmarks. A path in the space of landmarks can now be represented as a deterministic finite automaton; for example, in the case of street names as landmarks, nodes may correspond to streets, and edges with street intersections (with an appropriate output ground action to perform the correct turn, if any). Note that this abstraction only allows specifying a single action to be repeatedly performed between two landmarks; for example, when on a certain street, the agent can only move in one direction on the street, until an intersection is encountered. In this case, uniqueness of landmarks is essential, since they are the only source of information, unless transition models are also provided to enable tracking.

Richer abstractions

The initial motivation for the abstract mapping task was to follow an abstract 2-D map, such as the one in Figure 1. Ultimately, these maps are typically perceived via vision, and it would be much easier for a robot to use existing maps if it can process them in image form, rather than requiring a manual encoding of the abstract policy $\tilde{\pi}$. Compared to previous cases, using the 2-D map in image form is interesting because it is both featurally richer compared to previous abstractions, while at the same time still much lower-dimensional with respect to the robot. One possibility for using this image-based abstraction is to extract features from it, such as using convolutional neural networks, and to then learn to map ground states to abstract visual features.

The automaton-based abstraction in the previous case is also closely related to using natural language instructions for navigation. For example, "go straight on street A for two blocks until the intersection with street B, then turn left" can be represented as an automaton. We can therefore consider using natural language itself as an abstraction, either by mapping the sequence of instructions to an automaton, or by directly mapping ground states to abstract linguistic features, as in the case for images.

Discussion

We considered the problem of learning to use existing abstractions in novel environments, in the context of the problem of navigation using abstract 2-D maps. The problem was formulated as a fully-supervised, learning from demonstration problem, and several cases of potential abstraction classes were considered. In the process of analyzing these cases, various aspects and issues of abstraction were encountered, and many problems and solutions still lie ahead.

There remains the issue of learning the action abstraction function f_a. This is the problem of temporal abstraction, which has arguably received greater attention in the field thus far. One way to consider an abstract action $\tilde{a}$ is to view it as a subgoal, which instantiates a local planning problem. This was a potential strategy used to overcome the lack of orientation information in the abstract 2-D map.

So far, the problem has only been considered in the fully-supervised setting. Although this provides the strongest signal for learning, it also requires significant effort from the user. One possibility is provide weak supervision through reinforcement learning, in the extreme case only providing reward if the correct path is followed. An intermediate regime would be to still provide demonstrations, but no longer with ground-abstract state correspondences.

Two cases in the previous section touched upon the utility of learning transition models. The benefit so far appears to be increased robustness in determining the correct abstract state. Transition models are also needed for planning; if the solution path is not provided, and only an abstract map is given (which is the case when using a standard map), then planning in the abstract space will be necessary. This is a useful extension to the problem considered so far: find an abstract policy and follow it via the same grounding mechanism, with the assumption that the abstraction is a "faithful" representation of the world with respect to the task.

The proposed approach may also provide useful theoretical analysis of abstractions. Since the problem of learning abstractions has been transformed into one of supervised learning, we may be able to adapt theoretical tools from computational learning theory in this more familiar setting, and characterize the utility of various abstractions. In particular, to use the given abstraction effectively, we had to learn

the abstraction function f_s (and eventually f_a); the complexity of this learning problem tells us how practical the abstraction is. If it is difficult to learn f_s, then it may not be worth the extra learning effort for the potential reduction in representational complexity. An abstraction may only be useful if it is an accurate representation of the world with respect to the task, provides some degree of information compression, *and the abstraction functions are easy to learn*.

References

Abel, D.; Hershkowitz, D.; and Littman, M. 2016. Near optimal behavior via approximate state abstraction. In *International Conference on Machine Learning*.

Argall, B.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469–483.

Boniardi, F.; Behzadian, B.; Burgard, W.; and Tipaldi, G. 2015. Robot navigation in hand-drawn sketched maps. In *European Conference on Mobile Robots*.

Boniardi, F.; Valada, A.; Burgard, W.; and Tipaldi, G. 2016. Autonomous indoor robot navigation using a sketch interface for drawing maps and routes. In *IEEE International Conference on Robotics and Automation*.

Brunner, G.; Richter, O.; Wang, Y.; and Wattenhofer, R. 2018. Teaching a Machine to Read Maps with Deep Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*.

Cobo, L.; Subramanian, K.; Isbell, C.; Lanterman, A.; and Thomaz, A. 2014. Abstraction from demonstration for efficient reinforcement learning in high-dimensional domains. *Artificial Intelligence* 216:103–128.

Dietterich, T. 2000. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* 13:227–303.

Gao, W.; Hsu, D.; Lee, W.; Shen, S.; and Subramanian, K. 2017. Intention-net: Integrating planning and deep learning for goal-directed autonomous navigation. In *Conference on Robot Learning*.

Konidaris, G.; Kaelbling, L.; and Lozano-Pérez, T. 2018. From skills to symbols: Learning symbolic representations for abstract high-level planning. *Journal of Artificial Intelligence Research* 61:215–289.

Ravindran, B. 2004. *An Algebraic Approach to Abstraction in Reinforcement Learning*. Ph.D. Dissertation, University of Massachusetts Amherst.

Sutton, R.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1):181–211.

Taylor, M., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(1):1633–1685.