Reward Predictive Representations Generalize Across Tasks in Reinforcement Learning

Lucas Lehnert
Computer Science Department
Brown University
Providence, RI 02912, USA
lucas_lehnert@brown.edu

Michael J. Frank
Department of Cognitive,
Linguistic & Psychological Sciences
Carney Institute for Brain Science
Brown University
Providence, RI 02912, USA

Michael L. Littman
Computer Science Department
Brown University
Providence, RI 02912, USA

Abstract

One question central to reinforcement learning is which representations can be generalized or re-used across different tasks. Existing algorithms that facilitate transfer typically are limited to cases in which the transition function or the optimal policy is portable to new contexts, but achieving "deep transfer" characteristic of human behavior has been elusive. This article demonstrates that model reductions that minimize error in predictions of reward outcomes generalize across tasks with different transition and reward functions. Such state representations compress the state space of a task into a lower dimensional representation by combining states that are equivalent in terms of both the transition and reward functions. Because only state equivalences are considered, the resulting state representation is not tied to the transition and reward functions themselves and thus generalizes across tasks with different reward and transition functions. The presented results motivate further experiments to investigate if humans or animals learn such a representation, and whether neural systems involved in state representation reflect the modeled equivalence relations.

1 Introduction

A central question in reinforcement learning [26] is how to generalize across different states of a given task. Humans are adept at such flexible transfer but existing RL algorithms are much more limited. As a lay example, consider shifting gears in a manual transmission car. In a right-hand-drive country, the steering wheel is on the left side of the car and the right arm is used for shifting, whereas the opposite is the case in a left-hand-drive country. A person who has learned in one scenario can quickly generalize to the other, despite the fact that both tasks require different coordination of motor skills. Both tasks are the same in an abstract sense: In each case, there is a progression from 1st to 2nd gear and so on, which should be coordinated with the clutch pedal and steering, and this structure can be generalized from a left-hand-drive car to a right-hand-drive car [12, 6] and a driver does not have to learn how to drive from scratch.

By treating two different sensory inputs or states as equivalent, an agent can generalize what it has learned for one state to another and speed up learning [6, 1]. Such equivalences can be modeled using *state abstractions* [19, 12], which map states to a compressed latent representation. The usual RL

Preprint. Under review.

framework considers Markov Decision Processes (MDPs) [21] where an agent's sole objective is to maximize reward. In transfer or lifelong learning, an agent observes a sequence of MDPs and attempts to learn a state abstraction that can be re-used to speed up learning in a previously unseen task. State abstractions can be constructed in different ways, for example by merging states with the same optimal action or Q-values into the same latent or *abstract state*. This article considers two types of state abstractions:

1. *reward-maximizing state abstractions*, which allow an agent to maximize total reward, and
2. *reward-predictive state abstractions*, which allow an agent to predict future reward outcomes.

While many different RL transfer algorithms have been proposed (see Taylor and Stone [27] for a survey), this article demonstrates that reward-maximizing state abstractions fail to generalize across tasks, whereas reward-predictive state abstractions can be leveraged to improve generalization even when both transition and reward functions change across tasks. The presented analysis and simulations motivate the design of new RL algorithms that can discover such state abstractions as well as further experiments to investigate whether neural mechanisms in biological agents facilitate learning of such representations.

Previous work [17] shows that reward-predictive state abstractions can be extracted from the *successor representation* (SR) [8], which predicts the discounted expected frequency of visiting future states given the current state. While re-using a previously learned SR has been shown to speed up learning [20, 3] when reward functions change, these methods are limited to tasks with common transition functions. Further, if the optimal decision-making strategy differs between two tasks, the SR has to be re-learned [18]. In contrast, this article shows that reward-predictive state abstractions are not directly tied to the transition and reward functions themselves and thus afford transfer across tasks with different transition functions, reward functions, and optimal policies. Such "deep transfer" across environments, even in the absence of prior experience with specific transition or reward functions, is predicted by behavioral and neural signatures of human structure learning [6, 2, 13] but not afforded by alternative algorithms that compress the transition function directly [24, 23].

2 Generalization Across States

An MDP is a quintuple $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function p, and a reward function r. The discount factor γ controls how strongly short-term rewards are favored over long-term rewards. In *model-free learning*, for example Q-learning [29], the optimal decision-making strategy, called *policy*, is learned through trial and error interactions in an MDP. Throughout these interactions, a sub-optimal policy is improved incrementally to increase its Q-values. During learning, only the policy π and some form of cached Q-values of the policy π are stored at any point in time. In *model-based learning* [25, 5] an agent attempts to build a model of an MDP by learning to predict a sequence of future reward outcomes $(r_1, r_2, ...)$ given a particular sequence of actions $(a_1, a_2, ...)$. This approach is orthogonal to memorization and model-free learning, because using this model an RL agent can predict the value of any arbitrary policy. In this case, the agent's "knowledge" is sufficient to generalize across the space of all possible policies [25].

If the number of states in an MDP is large, it may be inefficient to learn the optimal policy using the state space directly. In this case, state abstractions [19, 22] provide a framework for modeling generalization across states in RL. A state abstraction, also called state representation, is a function ϕ mapping the state space $\mathcal{S}$ to some other latent space. Because state representations are many-to-one relations, they can map different states to the same outputs and create a partitioning of the state space. A state partition is a subset of the state space that maps to the same latent or abstract state. An agent using a state abstraction ϕ operates on the space of state partitions and generalizes knowledge learned in one state across the entire state partition. For example, in Q-learning, a value update is applied to the entire state partition even if the update is computed only from one specific state transition, resulting in faster learning [1, 6].

This article investigates which state abstraction ϕ should be learned from a hypothesis space of all possible state abstractions $\mathcal{H}$. A state abstraction is a function mapping states to some latent abstract state space $\mathcal{S}_\phi$. The state abstraction hypothesis space is then $\mathcal{H} = \{\phi : \mathcal{S} \rightarrow \mathcal{S}_\phi\}$. An agent that uses a state abstraction ϕ operates directly on the latent space $\mathcal{S}_\phi$ rather than the underlying state space $\mathcal{S}$. Depending on how ϕ constructs the latent state space $\mathcal{S}_\phi$, the agent may not be able to

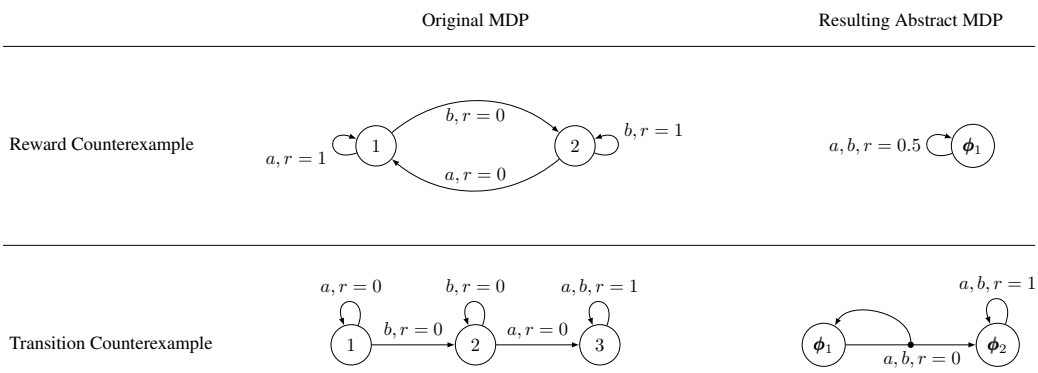


Figure 1: Sub-Optimal State Abstraction Examples. The reward counterexample illustrates what happens if a state abstraction generalizes across states with different one-step rewards. This example consists of two states and two actions. Depending on which state the agent is in, either action a or action b is optimal. Selecting a sub-optimal action results in receiving a reward of zero, instead of one, and transitioning to the other state. Here, if an agent collapses across these states, it cannot select actions optimally and will receive on average a sub-optimal reward of 0.5. The transition counterexample illustrates how a state abstraction can mask the transition dynamics of the underlying MDP. A trajectory $(1, 2, 3, 3, ...)$ through this MDP will be mapped to a trajectory $(\boldsymbol{\phi}_1, \boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_2, ...)$. In the original problem, all transitions are deterministic, but the resulting empirical transition probabilities in the latent state space are stochastic. In this case, the agent cannot use just the latent state $\boldsymbol{\phi}_1$ to accurately predict when the $+1$ reward is collected, because the transition into the latent state $\boldsymbol{\phi}_2$ is non-deterministic. To ensure accurate predictions of future reward outcomes and that Eq. (1) holds, the empirical transition probabilities in the latent space have to match the transition probabilities in the original state space.

distinguish between a rewarding and a non-rewarding state, for example. Figure 1 provides examples how different state abstractions can construct different (sub-optimal) abstract MDPs.

A reward-maximizing state abstraction $\phi \in \mathcal{H}$ is one that maximizes the total reward an agent can obtain in a given MDP, compared to any other abstraction. In contrast, a reward-predictive state abstraction $\phi \in \mathcal{H}$ allows an agent to best predict which reward will be observed after executing a decision sequence starting at a specific state. Specifically, such a state abstraction needs to satisfy for any start state s and action sequence $a_1, ..., a_n$ that

$$\mathbb{E}[r_t | s, a_1, ..., a_t] = \mathbb{E}[r_t | \phi(s), a_1, ..., a_t]. \tag{1}$$

The expectation on the left-hand side in Eq. (1) is conditioned on the state s and is computed over all possible trajectories in an MDP that follow the action sequence $a_1, ..., a_t$. The expectation on the right-hand side in Eq. (1) is conditioned only on the latent state $\phi(s)$, and consequently is computed over all possible trajectories in that latent space. The probability of observing a trajectory in the latent space is governed by the empirical transition probabilities between different latent states. For Eq. 1 to hold for all possible start states and decision sequences, the empirical transition probabilities in the latent space have to match the corresponding transition probabilities in the state space [15]. Figure 1 provides a counterexample explaining this idea further.

Successor features (SFs) [3] are a generalization of the SR [8], which predict the expected visitation frequencies in some latent feature space:

$$\boldsymbol{\psi}^{\pi}_{s,a} = \mathbb{E}\left[\phi(s_1) + \gamma\phi(s_2) + \gamma^2\phi(s_3) + \cdots \middle| s = s_1, a = a_1, \pi\right], \tag{2}$$

where the expectation is computed over all infinite length trajectories $(s_1, s_2, s_3, ...)$. Intuitively, SFs predict the frequencies with which states are encountered along a trajectory, because a frequently encountered state will occur in the summation in Eq. (2) more often than a rarely encountered state. If the empirical transition probabilities in the latent state space match the transition probabilities on the state space $\mathcal{S}$, then the frequency with which a latent state is encountered matches the frequency with which a state is encountered. In this case, the SFs and the expectation in Eq. (2) can be conditioned directly on the latent state $\phi(s)$, similar to Eq.(1). Reward-predictive state abstractions predict SFs and learning SFs is equivalent to learning such abstractions [17].

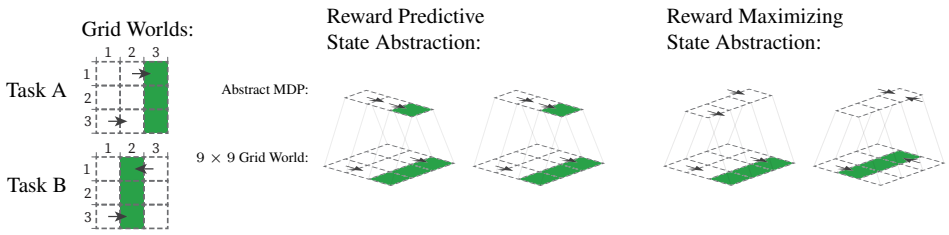


Figure 2: Transferring State Abstractions between MDPs. In both grid world tasks, the agent can move up, down, left, or right, and is rewarded if a green cell is entered. The two centre schematics show a reward-predictive state abstraction for each task, which in this case generalizes across rows and merges each column into one latent state. In both tasks, reward sequences can be predicted using only the compressed representation. The schematics on the right show one possible reward-maximizing state abstraction. While it is possible to learn or compute the optimal policy using this state abstraction in Task A, this state abstraction cannot be used to learn the optimal policy in Task B. In Task B, the abstract representation cannot be used to represent a policy of moving to the centre. This reward-maximizing state abstraction does not generalize to Task B.

3 Generalization Across Tasks

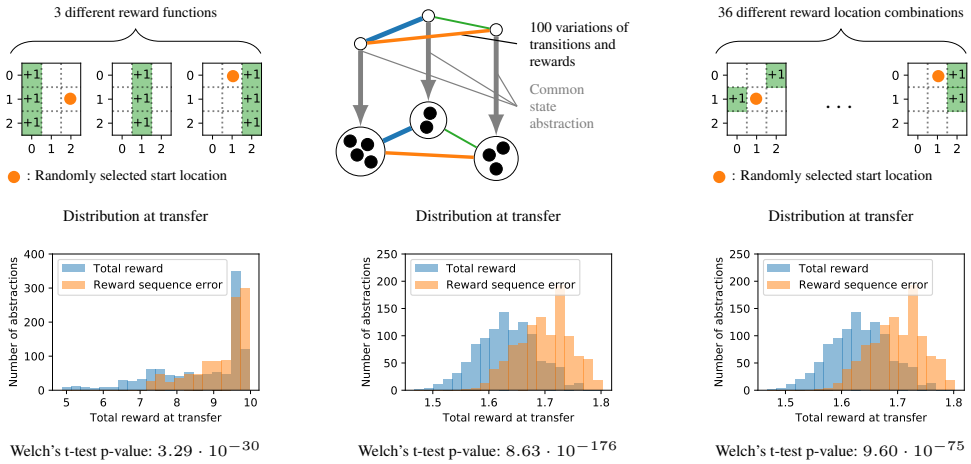
To generalize knowledge across different tasks, an agent can re-use a learned state abstraction when learning how to solve a previously unseen task. Consider the transfer example in Figure 2, where an agent is first presented Task A, and then transfers a state abstraction to Task B. In this example the agent is required to generalize and compress the nine grid states into three latent states. The difference between both tasks is only in which column of the grid world a reward is given to the agent. In this toy example, the reward-predictive state abstraction can be re-used in Task B, while the reward-maximizing state abstraction cannot be re-used in Task B. Because state abstractions only model equivalences between states, they are not directly tied to the reward function, transition function, or optimal policy. The following experiments will also transfer state abstractions across tasks with different transition functions. This approach to transfer stands in contrast to the usual approach of transferring SFs. While re-using SFs learned for one task on a task with a different reward function can speed up learning [20, 3], if the reward function changes the optimal policy, then the SFs have to be relearned [18]. These SF transfer algorithms also assume a fixed transition function. Whittington et al. [30] present a model of predictive representations in the hippocampus that is designed to predict future outcomes. While this model is not designed in the usual RL framework, predicting future outcomes or stimuli can also be understood as a form of reward prediction. However, this model is trained directly on entire interaction sequences to predict future outcomes and the learned representations are thus tied to the transition function. The example in Figure 2 shows that reward-predictive state abstractions are not restricted by these limitations and can be directly re-used, assuming certain state equivalences are preserved.

3.1 Transfer with Single State Abstractions

To further investigate if reward-maximizing or reward-predictive state abstractions generalize across different tasks, the first set of experiments considers an agent that learns one state abstraction on one task and then re-uses the learned abstraction on a set of different tasks. The goal of this experiment is to be algorithm agnostic: Rather than focusing on how a particular algorithm performs at transfer with a single learned state abstraction, the entire hypothesis space $\mathcal{H}$ is enumerated and evaluated on all transfer tasks. For each task and state abstraction, the compressed abstract MDP is computed [22] and this abstract MDP is solved using value iteration [26, Chapter 4.4] (see also Appendix A). A reward-maximizing state abstraction is then identified by testing the computed policy in each task for N trials over T time steps and computing the total reward

$$R_{\text{total}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} r_{n,t}, \tag{3}$$

where $r_{n,t}$ is the reward incurred in trial n at time step t. A reward-predictive state abstraction is identified by sampling N random state and action sequence $a_1, ..., a_T$ and predicting the reward



(a) Column World Transfer. (b) Random MDP Transfer. (c) Grid World Navigation Transfer.

Figure 3: Minimizing reward-sequence prediction errors identifies state abstractions amenable for "deep transfer". For each experiment, all possible state abstractions in $\mathcal{H}$ were enumerated using Algorithm U [16] to obtain a ground truth distribution over the hypothesis space $\mathcal{H}$. State abstractions were scored by compressing an MDP using the state abstraction of interest [19]. The total reward score was computed by running the computed policy 20 times for 10 time steps in the MDP from a randomly selected start state. The reward-sequence error was computed by selecting 20 random start states and then performing a random walk for 10 time steps. The histograms report averages over all repeats and transfer MDPs. The Welch's t-test was performed to compute the p-values of the difference in mean total reward being insignificant for each histogram.

sequence $\hat{r}_{n,1}, ..., \hat{r}_{n,T}$ using the abstract MDP. The reward-sequence prediction error is

$$RS_{\text{error}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} |r_{n,t} - \hat{r}_{n,t}|. \tag{4}$$

On three transfer task sets, Figure 3 shows that a state abstraction's ability to predict future reward outcomes is indicative that this representation can be re-used in a previously unseen MDP. On a previously unseen MDP, a reward-predictive state abstraction performs better than reward-maximizing state abstractions. As explained by the examples in Figure 1, a policy optimal in the compressed MDP is not necessarily optimal in the original MDP and cannot generate a high total reward. For each transfer experiment, the top 5% scoring state abstractions were then re-evaluated on the remaining transfer MDPs and the total rewards generated by these state abstractions are plotted as histograms in Figure 3. In all cases, state abstractions with low reward-sequence prediction errors RS_{error} generate a higher total reward at transfer than state abstractions that were selected based on their ability to construct a well performing policy and produce a high R_{total} score on the original MDP. This result indicates that reward-predictive state abstractions encode information about an MDP that can be generalized across different MDPs that share the same abstract structure (see Section 3.2 for extension to environments in which multiple structures are possible and have to be inferred).

Figure 3(a) shows the results for the transfer experiment discussed in Figure 2. The histogram in Figure 3(a) indicates that state abstractions with low reward-sequence prediction errors outperform on average state representations that only maximize total reward in one of the tasks. The experiment in Figure 3(b) is similar to the previous experiment in that each of the 100 randomly generated MDPs can be compressed with the same state representation. These MDPs are constructed by generating random three-state MDPs and then "inflating" the state space size using one common but randomly generated state abstraction. Besides this common "hidden" state representation, these 100 MDPs differ in both transition and reward functions. The histogram confirms the claim that low reward-sequence prediction errors are indicative of a state abstraction's ability to generalize across different MDPs. State representations that result in high total reward in only one of the 100 MDPs generate on average less reward on any of the remaining MDPs. Figure 3(c) presents a transfer experiment where two

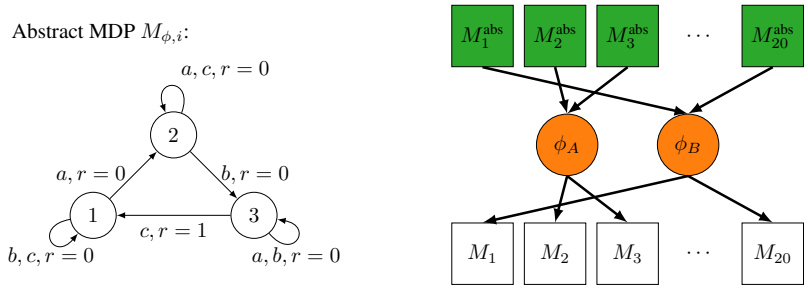


Figure 4: Transfer with Multiple State Abstractions Curriculum. A curriculum of transfer tasks is generated by first constructing the three-state MDP shown on the left. At each state, only one action causes a transition between different states and only one transition between different states is rewarded. To generate a sequence of abstract MDPs $M_1^{\text{abs}}, ..., M_{20}^{\text{abs}}$, the action labels are randomly permuted as well as the transitions generating positive reward (similar to the Diabolical Rooms Problem [12]). In generated MDPs, the optimal policy is to cycle through all node states. Two hidden state abstractions ϕ_A and ϕ_B were randomly selected to "inflate" each abstract MDP to a nine-state problem. One state abstraction was used with a frequency of 75% and the other with a frequency of 25%. The resulting MDP sequence $M_1, ..., M_{20}$ is presented to the agent, without any information about which state abstraction was used to construct the task sequence.

reward locations are permuted in a grid world. In this experiment, the MDPs cannot be compressed without incurring some loss, because the grid location is important for predicting where the goal locations are and what action is optimal at each location. However, the histogram in Figure 3(c) indicates that representations selected based on minimizing the reward-sequence prediction error criterion still perform better than selecting representations by their total reward. Because grid worlds have a specific topology of the state space, a state representation clustering only neighbouring states preserves approximately the grid location information and would be expected to perform relatively well across all MDPs.

3.2 Transfer with Multiple State Abstractions

The previous experiment assumes that all tasks share a common "hidden" state abstraction that can be learned by an agent. This section presents an experiment that violates this assumption and presents a non-parametric Bayesian model for an agent that maintains a belief space of possible state abstractions [6, 12]. Figure 4 illustrates how the curriculum of tasks is generated. After observing an MDP sequence $M_1, ..., M_t$, the agent updates its belief space $\mathcal{B}_t$ using a posterior over which state abstraction can be used to solve a given task M_t:

$$\Pr(\phi|M_t, \mathcal{B}_t, c_t) \propto \Pr(\phi|M_t)\Pr(\phi|\mathcal{B}_t, c_t), \tag{5}$$

where $c_t(\phi)$ is the count of how often an abstraction ϕ was used in the previous $t-1$ tasks. These counts are used to construct a Chinese Restaurant Process (CRP) [14, 28] prior for an intensity $\alpha \geq 0$:

$$\Pr(\phi|\mathcal{B}_t, c_t) = \begin{cases} \frac{c_t(\phi)}{t-1+\alpha} & \text{if } \phi \in \mathcal{B}_t \\ \frac{\alpha}{t-1+\alpha} & \text{otherwise.} \end{cases} \tag{6}$$

The posterior is also conditioned on the MDP M_t through the factor $\Pr(\phi|M_t)$. Using a loss function l, each state abstraction ϕ can be scored and for $\beta \geq 0$, the probability of this state abstraction being suitable to solve M_t is the soft-max probability

$$\Pr(\phi|M_t) \propto e^{-\beta l(\phi)}. \tag{7}$$

To determine which state abstraction should be added into the abstraction belief set $\mathcal{B}_t$, the non-parametric Bayesian agent has access to an oracle that returns the best scoring state abstraction $\phi_{\text{next-best}}$ not included into $\mathcal{B}_t$. The posterior $\Pr(\phi|M_t, \mathcal{B}_t, c_t)$ is computed over the set of state abstraction $\mathcal{B}_t \cup \{\phi_{\text{next-best}}\}$. Similar to the previous experiment, the goal is not to design an algorithm that can solve a sequence of tasks efficiently, but to analyze which state abstractions generalize across different tasks. Using an oracle that tabulates the hypothesis space gives insight into which state

abstractions generalize across different tasks, while being algorithm agnostic. If α increases, the resulting prior and posterior assign a higher probability to adding the next-best state abstraction $\phi_{\text{next-best}}$ into $\mathcal{B}_t$. If β increases, then more emphasis is given on using the loss function l to determine which state abstraction should be used from the set $\mathcal{B}_t \cup \{\phi_{\text{next-best}}\}$.

Rather than using the empirical scores R_{total} or RS_{error}, the agent is allowed to observe a tabulation of all possible transitions and rewards to obtain a ground truth score for each abstraction. In these experiments, reward-maximizing state abstractions are identified by accessing how much using a state abstraction impacts the value of the policy π_ϕ, which is optimal in the abstract MDP:

$$l_{\text{maximizing}}(\phi) = \max_{s \in \mathcal{S}} V^{\pi^*}(s) - V^{\pi_\phi}(s), \tag{8}$$

where V^π is the discounted value function [26] and π^* is the optimal policy. Reward-predictive state abstractions are scored by the loss function $l_{\text{predictive}}$ bounding the reward-sequence prediction error

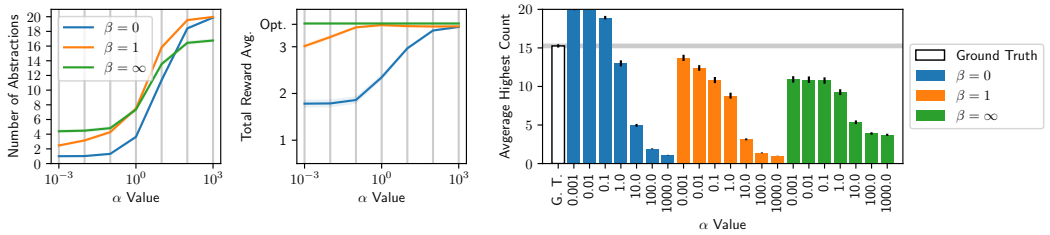
$$\forall i, \forall a_1, ..., a_t,\ l_{\text{predictive}}(\phi) C_{\gamma,t} \geq |\mathbb{E}[r_t | i, a_1, ..., a_t] - \hat{r}_t|\,, \tag{9}$$

where $C_{\gamma,t}$ is a constant that depends on the action sequence length t and discount factor γ. The loss function $l_{\text{predictive}}$ is computed using the SF model presented by [17]. Appendix B presents all details on how to compute $l_{\text{predictive}}$. If any of the two loss functions evaluates to zero for a state abstraction ϕ, then ϕ is either a globally optimal reward-maximizing or reward-predictive state abstraction.

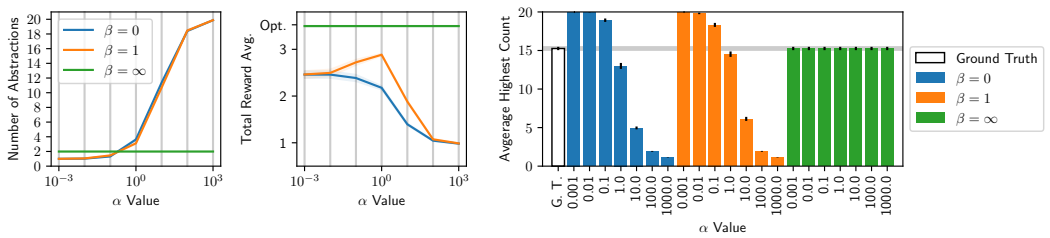
Figure 5 plots the results from testing the agent with each loss function for various α and β settings. Setting $\beta = \infty$ means that the probability $\Pr(\phi|M_t)$ is deterministic: The highest scoring state abstraction is assigned a probability of one. This case is equivalent to only using the loss function to select a state abstraction while ignoring the CRP prior $\Pr(\phi|\mathcal{B}_t, c_t)$. For each state abstraction ϕ, the policy optimal in the corresponding abstract MDP is computed. A control policy was constructed by using the posterior to mix the policies optimal in the respective abstract MDPs. For low β settings, the prior is used to determine which state abstraction is used. If α is high, then up to 20 state abstractions are added into the belief set $\mathcal{B}_t$. Because the prior influences the posterior heavily, the total reward of the resulting agent is comparably low, because the agent is not well informed about which state abstraction should be used on a given task. For $\beta = \infty$, the loss function influences the posterior strongly. In this case, an optimal total reward level is obtained using either loss function. The key difference between both loss functions becomes apparent when analyzing how the agent maintains the belief space $\mathcal{B}_t$. Using the loss function $l_{\text{predictive}}$, which identifies reward-predictive state abstractions, the agent identifies the correct ground truth state abstractions that were used to generate the task sequences. The count bar charts in Figure 5 also show that the agent correctly learns that one state abstraction occurs with a frequency of 75%. Because the agent only maintains two belief abstractions, the agent correctly estimates that the other abstraction occurs with a frequency of 25%. In contrast, when the loss function $l_{\text{maximizing}}$ is used, the agent does not isolate the correct number of state abstractions for any of the tested α settings. At best, using $l_{\text{maximizing}}$ and a small α value the agent is capable of isolating five state abstractions. For high α settings, the agent effectively memorizes a solution for almost every task by increasing the size of its belief set $\mathcal{B}_t$, because a previously used state abstraction does not generalize to the next task. When using the loss function $l_{\text{predictive}}$, the agent can correctly identify which state abstraction to use for which MDP, confirming the claim that reward-predictive state abstractions generalize across different tasks.

4 Discussion

In RL, the agent's goal is to find a reward-maximizing policy in an MDP. In the context of transferring knowledge across different tasks with large state spaces, this article demonstrates that an agent's objective changes when it comes to learning state abstractions. Rather then learning reward-maximizing state abstractions, the presented results show that an agent should focus on learning reward-predictive state abstractions. While reward-predictive state abstractions do not limit an agent's ability to obtain an optimal policy for an MDP [15], the solution space of possible reward-predictive state abstractions is far more constrained. François-Lavet et al. [11] construct latent state representations as part of a model-free and model-based hybrid model that constructs a latent state representation and extracts the underlying state transition dynamics. In contrast to their method, reward-predictive state abstractions compress the state space by generalizing across bisimilar states. Li et al. [19] present theoretical results showing that reward-predictive state abstractions, also called model reductions, construct a latent state space that is not as low dimensional as one that generalizes across states that share



(a) Reward-Maximizing State Abstractions using $l_{\text{maximizing}}$



(b) Reward-Predictive State Abstractions using $l_{\text{predictive}}$

Figure 5: Results for Transfer with Multiple State Abstractions Experiment. Each plot shows averages over 100 repeats and standard errors are reported as shaded areas (variations are very low if not visible). The left panels plot the average size of $\mathcal{B}_t$ after training. The middle panels plot the average total reward per MDP. If $\beta = \infty$ the agent obtains an optimal total reward level when using either loss function for ten time steps in each MDP. The right panels plot the average count for the most frequently used state abstraction. The white bar labelled "G. T." plots the ground-truth frequency.

the same common optimal action, for example. Abel et al. [1] consider approximate versions of state abstractions that generalize across states with equal optimal actions or Q-values. However, results show that such state abstractions need to be iteratively improved by observing a sequence of MDPs. This article presents results indicating that reward-predictive state abstractions can be directly re-used across tasks, without having to iteratively improve a previously learned state abstraction. The non-parametric Bayesian model presented in Section 3.2 demonstrates that by searching for reward-predictive state abstractions, an agent can correctly identify which state abstraction to use on a given MDP. While this article uses the SF model presented by Lehnert and Littman [17] to compute $l_{\text{predictive}}$, several other methods exist to evaluate reward-predictive state abstractions [7, 10, 9].

Previous work on transfer with SFs has shown that re-using SFs on an MDP with a different reward function can provide faster learning [3, 20, 23]. However, SFs are fragile to changes in the optimal policy [18] and transition function, whereas latent state abstractions are more abstract and thus are not tied to particular transitions [6, 12]. In related work, Stachenfeld et al. [24] compress the SR of an MDP using PCA and demonstrate this representations suitability for transfer and connections to place cells and grid cells in the hippocampus. However, this compressed SR constructs a representation of the transition function itself, and hence transfer is limited to environments that share the same transition function. In contrast to Stachenfeld et al., reward-predictive state abstractions separate the transition dynamics (and the SR) from the compression on the state space itself, and thus generate a latent state representation of a task exploiting task equivalences. Whittington et al. [30] present a model for generalization in the hippocampal-entorhinal system [4]. Similar to reward-predictive state abstractions, this model learns a latent representation that is predictive of future outcomes or stimuli but is also tied to a fixed transition function. Collins and Frank [6] and Badre and Frank [2] demonstrate that reward prediction errors and prediction errors on the structure of the state space identify latent state representations that accelerate learning in humans when transferring knowledge across tasks. While this work considers contextual multi-armed bandits, reward-predictive state abstractions may extend this work to sequential decision making [13].

5 Conclusion

The presented results suggest that reward-predictive state abstractions generalize across tasks with different transition and reward functions, motivating the design of future transfer algorithms. The discussed connections to predictive representations in the brain and generalization in human and animal learning motivate further experiments to investigate if biological systems learn reward-predictive representations.

A State Abstractions

For finite state and action MDPs, the transition function can be presented as a set of left-stochastic transition matrices $\{\boldsymbol{P}^a\}_a$ where

$$\forall i, j \in \mathcal{S},\ \boldsymbol{P}^a(i,j) = p(i,a,j), \tag{10}$$

and a set of expected reward vectors $\{\boldsymbol{r}^a\}_a$ where

$$\forall i \in \mathcal{S},\ \boldsymbol{r}^a(i) = \mathbb{E}_{s'}[r(s,a,s')]. \tag{11}$$

A state abstraction is an MDP homomorphism [22] mapping an MPD $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ into an abstract MDP $M_\phi = \langle \mathcal{S}_\phi, \mathcal{A}, p_\phi, r_\phi, \gamma \rangle$. Using a weighting function ω, the abstract MDP "aggregates" the transition and reward function across states $s \in \mathcal{S}$ that map to the same abstract state $s_\phi \in \mathcal{S}_\phi$. Specifically, the abstract reward function r_ϕ is constructed as

$$r_\phi(s_\phi, a) = \sum_{s:\phi(s)=s_\phi} \omega(s) r(s,a), \tag{12}$$

where $r(s,a)$ are one-step expected rewards. The transition function is constructed as

$$p_\phi(s_\phi, a, s'_\phi) = \sum_{s:\phi(s)=s_\phi} \sum_{s':\phi(s')=s'_\phi} \omega(s) p(s,a,s'). \tag{13}$$

The weighting function is assumed to be non-negative and the sum across a state partition evaluates to one: $\sum_{s:\phi(s)=s_\phi} \omega(s) = 1$. This article assumes a uniform weighting function which averages across state partitions. Similar to Eq. (10) and Eq. (11), the abstract transition matrices $\{\boldsymbol{P}^a_\phi\}_a$ and abstract reward vectors $\{\boldsymbol{r}^a_\phi\}$ can be computed. A policy that is optimal in the abstract MDP is then computed using the Value Iteration algorithm directly on $\{\boldsymbol{P}^a_\phi\}_a$ and $\{\boldsymbol{r}^a_\phi\}$. Such a policy is then the ground truth optimal policy an agent can use while generalizing across states according to the state abstraction ϕ.

B Successor Features Identify Reward-Predictive State Abstractions

Lehnert and Littman [17] show that SFs can be used to identify reward-predictive state abstractions. For an MDP, the SR [8] is

$$\boldsymbol{\Psi}^\pi = \sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{P}^\pi, \tag{14}$$

where $\boldsymbol{P}^\pi$ is the transition matrix under a policy π. The action-conditional SR can be defined as

$$\boldsymbol{\Psi}^{a,\pi} = \boldsymbol{I} + \gamma \boldsymbol{P}^a \sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{P}^\pi = \boldsymbol{I} + \gamma \boldsymbol{P}^a \boldsymbol{\Psi}^\pi. \tag{15}$$

Using the transition matrix for the abstract MDP, the action conditional SR can be defined in the abstract state space as

$$\boldsymbol{F}^{a,\pi} = \boldsymbol{I} + \gamma \boldsymbol{P}^a_\phi \sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{P}^\pi_\phi. \tag{16}$$

To measure how well a state abstraction ϕ can predicts one-step rewards and the SFs, two quantities ε_r and ε_ψ are defined:

$$\forall s \in \mathcal{S},\ \forall a \in \mathcal{A},\ \varepsilon_r = |\boldsymbol{e}_s^\top \boldsymbol{r}^a_\phi - r(s,a)|, \quad \varepsilon_\psi = ||\boldsymbol{e}_s^\top \boldsymbol{F}^a - \boldsymbol{\psi}^\pi_{s,a}||_2, \tag{17}$$

where $\boldsymbol{e}_s$ is a one-hot bit vector that corresponds to the abstract state a state s is mapped to. If entry i is set to one in the vector $\boldsymbol{e}_s$, then $\boldsymbol{e}_s$ corresponds to the abstract state $i \in \mathcal{S}_\phi$. These prediction errors for one-step rewards and SFs bound the expected future reward prediction errors

$$\left| \boldsymbol{e}_s^\top \boldsymbol{P}_\phi^{a_1} \cdots \boldsymbol{P}_\phi^{a_{t-1}} \boldsymbol{r}_\phi^{a_t} - \mathbb{E}[r_t | s, a_1, ..., a_t] \right| \leq \varepsilon_r + C_{\gamma,t} \varepsilon_\psi \tag{18}$$

where $C_{\gamma,t}$ is a constant depending on the discount factor γ and sequence length T. The product $\boldsymbol{e}_s^\top \boldsymbol{P}_\phi^{a_1} \cdots \boldsymbol{P}_\phi^{a_{t-1}}$ evaluates to a row probability vector specifying with which probability an abstract state is reached. The expected reward is then predicted by computing the dot-product between this vector and the reward vector $\boldsymbol{r}_\phi^{a_t}$. The loss function $l_{\text{predictive}}$ is then computed with

$$l_{\text{predictive}}(\phi) = \varepsilon_r + \varepsilon_\psi. \tag{19}$$

If $l_{\text{predictive}}(\phi) = 0$, the ϕ is a reward-predictive state abstraction.

References

[1] David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 10–19, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/abel18a.html`.

[2] David Badre and Michael J Frank. Mechanisms of hierarchical reinforcement learning in cortico–striatal circuits 2: Evidence from fmri. *Cerebral cortex*, 22(3):527–536, 2011.

[3] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

[4] Timothy EJ Behrens, Timothy H Muller, James CR Whittington, Shirley Mark, Alon B Baram, Kimberly L Stachenfeld, and Zeb Kurth-Nelson. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018.

[5] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[6] Anne Gabrielle Eva Collins and Michael Joshua Frank. Neural signature of hierarchically structured expectations predicts clustering and transfer of rule sets in reinforcement learning. *Cognition*, 152:160–169, 2016.

[7] Gheorghe Comanici, Doina Precup, and Prakash Panangaden. Basis refinement strategies for linear value function approximation in mdps. In *Advances in Neural Information Processing Systems*, pages 2899–2907, 2015.

[8] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[9] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous markov decision processes. *SIAM Journal on Computing*, 40(6):1662–1714, 2011.

[10] Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *UAI*, pages 210–219. Citeseer, 2014.

[11] Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. *arXiv preprint arXiv:1809.04506*, 2018.

[12] Nicholas T Franklin and Michael J Frank. Compositional clustering in task structure learning. *PLoS computational biology*, 14(4):e1006116, 2018.

[13] Nicholas T Franklin and Michael J Frank. Generalizing to generalize: when (and when not) to be compositional in task structure learning. *bioRxiv*, 2019. doi: 10.1101/547406.

[14] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

[15] Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003.

[16] Donald E Knuth. *The Art of Computer Programming, Volume 4, Fascicle 3: Generating All Combinations and Partitions*. Addison-Wesley, 2005.

[17] Lucas Lehnert and Michael L Littman. Successor features support model-based and model-free reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2019.

[18] Lucas Lehnert, Stefanie Tellex, and Michael L Littman. Advantages and limitations of using successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1708.00102*, 2017.

[19] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

[20] Ida Momennejad, Evan M Russek, Jin H Cheong, Matthew M Botvinick, ND Daw, and Samuel J Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680, 2017.

[21] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[22] Balaraman Ravindran and Andrew G Barto. Approximate homomorphisms: A framework for non-exact minimization in markov decision processes. 2004.

[23] Evan M Russek, Ida Momennejad, Matthew M Botvinick, Samuel J Gershman, and Nathaniel D Daw. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS computational biology*, 13(9):e1005768, 2017.

[24] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20:1643 EP –, 10 2017. URL `https://doi.org/10.1038/nn.4650`.

[25] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the seventh international conference on machine learning*, pages 216–224, 1990.

[26] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[27] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[28] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. doi: 10.1198/016214506000000302. URL `https://doi.org/10.1198/016214506000000302`.

[29] Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.

[30] James Whittington, Timothy Muller, Shirely Mark, Caswell Barry, and Tim Behrens. Generalisation of structural knowledge in the hippocampal-entorhinal system. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8484–8495. Curran Associates, Inc., 2018.