Journal of Machine Learning Research 21 (2020) 1-53 Submitted 1/19; Revised 9/20; Published 10/20

Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning

Lucas Lehnert LUCAS_LEHNERT@BROWN.EDU
Computer Science Department
Carney Institute for Brain Science
Brown University
Providence, RI 02912, USA

Michael L. Littman MICHAEL_LITTMAN@BROWN.EDU
Computer Science Department
Brown University
Providence, RI 02912, USA

Editor: Shie Mannor

Abstract

A key question in reinforcement learning is how an intelligent agent can generalize knowledge across different inputs. By generalizing across different inputs, information learned for one input can be immediately reused for improving predictions for another input. Reusing information allows an agent to compute an optimal decision-making strategy using less data. State representation is a key element of the generalization process, compressing a high-dimensional input space into a low-dimensional latent state space. This article analyzes properties of different latent state spaces, leading to new connections between model-based and model-free reinforcement learning. Successor features, which predict frequencies of future observations, form a link between model-based and model-free learning: Learning to predict future expected reward outcomes, a key characteristic of model-based agents, is equivalent to learning successor features. Learning successor features is a form of temporal difference learning and is equivalent to learning to predict a single policy's utility, which is a characteristic of model-free agents. Drawing on the connection between model-based reinforcement learning and successor features, we demonstrate that representations that are predictive of future reward outcomes generalize across variations in both transitions and rewards. This result extends previous work on successor features, which is constrained to fixed transitions and assumes re-learning of the transferred state representation.

Keywords: Successor Features, Model-Based Reinforcement Learning, State Representations, State Abstractions

1. Introduction

A central question in Reinforcement Learning (RL) (Sutton and Barto, 2018) is how to process high dimensional inputs and compute decision-making strategies that maximize rewards. For example, a self-driving car has to process all its sensor data to decide when to accelerate or brake to drive the car safely. If the sensor data is high dimensional, obtaining an optimal decision-making strategy may become computationally very difficult because an optimal decision has to be computed for every possible sensor input. This "curse of

dimensionality" (Bellman, 1961) can be overcome by compressing high dimensional sensor data into a lower dimensional latent state space. In the self-driving car example, if the car is following another vehicle that is stopping, detecting brake lights is sufficient to make the decision to slow the self-driving car down. Other information such as the colour of the car in front can be ignored. In RL, these decisions are grounded through a reward function, which would give high reward if the self-driving car reaches its destination and low reward if an accident is caused. An intelligent agent can simplify a task by mapping high-dimensional inputs into a lower dimensional latent state space, leading to faster learning because information can be reused across different inputs. This article addresses the question how different principles of compressing inputs are related to one another and demonstrates how an agent can learn to simplify one task to accelerate learning on a different previously unseen task.

Previous work presents algorithms that reuse Successor Features (SFs) (Barreto et al., 2017a) to initialize learning across tasks with different reward specifications, leading to improvements in learning speed in challenging control tasks (Barreto et al., 2018; Zhang et al., 2017; Kulkarni et al., 2016). This article follows a different methodology: By analyzing which properties different latent state spaces are predictive of, different models of generalizations are compared leading to new connections between latent state spaces that support either *model-free RL* or *model-based RL*. Model-free RL memorizes and makes predictions for one particular decision-making strategy, while model-based RL aims at predicting future reward outcomes for any arbitrary decision-making strategy. Our analysis demonstrates how previous formulations of SFs learn latent state spaces that are predictive of the optimal decision-making strategy and are thus akin to model-free RL. This article introduces a new model, called the *Linear Successor Feature Model (LSFM)*, and presents results demonstrating that LSFMs learn latent state spaces that support model-based RL.

Latent state spaces model equivalences between different inputs and are suitable for transfer across different tasks only if these equivalences are preserved. Because LSFMs construct latent state spaces that extract equivalences of a task's transition dynamics and rewards, they afford transfer to tasks that preserve these equivalences but have otherwise different transitions, rewards, or optimal decision-making strategies. Ultimately, by compressing a high-dimensional state space, the representation learned by an LSFM can enable a model-based agent to learn a model of an MDP and an optimal policy significantly faster than agents that do not use a state abstraction or use a state abstraction of a different type. In contrast to SFs, which afford transfer across tasks with different rewards but assume fixed transition dynamics (Barreto et al., 2017a; Stachenfeld et al., 2017), LSFMs remove the assumption of a fixed transition function. While SFs are re-learned and adjusted to find the optimal policy of a previously unseen task (Lehnert et al., 2017), the presented experiments outline how LSFMs can preserve the latent state space across different tasks and construct an optimal policy by using less data than tabular RL algorithms that do not generalize across inputs.

To unpack the different connection points and contributions, we start by first presenting the two state representation types in Section 2 and provide a formal introduction of successor features in Section 3. Subsequently, Section 4 introduces LSFMs and presents the main theoretical contributions showing how LSFMs can be used for model-based RL. Then, Section 5 presents the link to model-free learning and presents a sequence of examples and

simulation illustrating to what extent the representations learned with LSFMs generalize across tasks with different transitions, rewards, and optimal policies.

2. Predictive State Representations

An RL task is formalized as a Markov Decision Processes (MDP) $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where the state space $\mathcal{S}$ describes all possible sensor inputs, and the finite action space $\mathcal{A}$ describes the space of all possible decisions. The state space $\mathcal{S}$ is assumed to be an arbitrary set and can either be finite or uncountably infinite. All results presented in this article are stated for arbitrary state spaces $\mathcal{S}$ unless specified otherwise. How the state changes over time is determined by the transition function p, which specifies a probability or density function of reaching a next state s' if an action a is selected at state s. Transitions are Markovian and the probability or density of transitioning to a state s' is specified by $p(s'|s, a)$. The reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [-R, R]$ specifies which reward is given for each transition and rewards are bounded in magnitude by a constant $R \in \mathbb{R}$. Besides assuming a bounded reward function, both reward and transition functions are assumed to be arbitrary.

A *policy* π describes an agent's decision-making strategy and specifies a probability $\pi(s, a)$ of selecting an action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$. A policy's performance is described by the value function

$$V^\pi(s) = \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_1 = s\right], \tag{1}$$

which predicts the expected discounted return generated by selecting actions according to π when trajectories are started at the state s. The expectation[1] in Equation (1) is computed over all infinite length trajectories that select actions according to π and start at state s. Similarly, the Q-value function is defined as

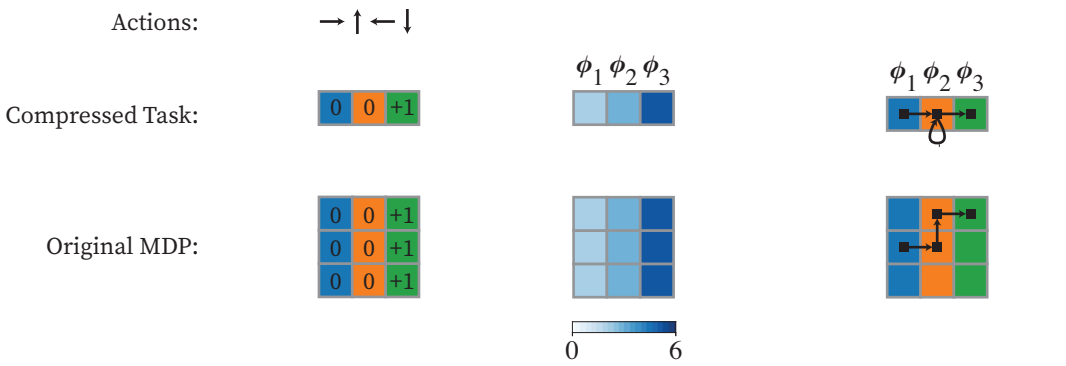
$$Q^\pi(s, a) = \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_1 = s, a_1 = a\right]. \tag{2}$$

The expectation of the Q-value function in Equation (2) is computed over all infinite length trajectories that start at state s with action a and then follow the policy π.

A latent state space $\mathcal{S}_\phi$ is constructed using a *state representation* function $\phi : \mathcal{S} \to \mathcal{S}_\phi$. A state representation can be understood as a compression of the state space, because two different states s and $\tilde{s}$ can be assigned to the same latent state $\phi(s) = \phi(\tilde{s})$. In this case, the state representation ϕ aliases s and $\tilde{s}$.

Figure 1(a) presents a grid world example where nine states are compressed into three different latent states and $\mathcal{S}_\phi = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3\}$. In this example, the state representation partitions the state space along three different columns and constructs a smaller latent grid world of size 3×1. If an intelligent agent uses this state representation for learning, the agent would only maintain information about which column it is in but not which row and effectively operate on this smaller latent 3×1 grid world.

1. The subscript of the expectation operator $\mathbb{E}$ denotes the probability distributions or densities over which the expectation is computed.



(a) Column World Example (b) State Values V^π (c) Example Trajectory

Figure 1: State Representations Construct Lower Dimensional Latent State Spaces. 1(a): The column world example is a 3×3 grid world where an agent can move up (action $\uparrow$), down (action $\downarrow$), left (action $\leftarrow$), or right (action $\rightarrow$) to adjacent grid cells and entering the right (green) column is rewarded. The number in each grid cell indicate the reward obtained by entering the respective cell. The state representation merges each column (colour) into a different latent state. 1(b): The lower panel presents a matrix plot of the state values V^π for a policy that selects actions uniformly at random. Grid cells of the same column have equal distance to the rewarding column and thus equal state values. Because this state representation only generalizes across states of the same column, the constructed latent state space can be used to predict the value function V^π as well. The top panel, which presents a matrix plot of state values for the three latent states, illustrates how the latent state space can be used for value predictions as well. 1(c): Using the latent state space, a latent 3×1 MDP can be constructed. Both latent MDP and original MDP produce equal reward sequences for trajectories that follow the same action sequence and that start at corresponding states and latent states. The figure illustrates such an example trajectory that starts at (1,1) or latent state $\boldsymbol{\phi}_i$ and then follows the action sequence $\rightarrow, \downarrow, \rightarrow$. Selecting the action $\downarrow$ is modelled as a self loop in the latent MDP. Both trajectories generate the same reward sequence of $0, 0, 1$. Intuitively, the compressed 3×1 grid world simulates reward sequences in exactly the same way as the 3×3 column world task.

2.1 Value-Predictive State Representations

A value-predictive state representation constructs a latent state space that retains enough information to support accurate value or Q-value predictions. Figure 1(b) shows that the value of states of the same column are equal. Suppose each latent state is represented as a one-hot bit vector[2] in three dimensions and $\mathcal{S}_\phi = \{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\}$. Because each grid cell is mapped to a one-hot bit vector, $\phi(s) = \boldsymbol{e}_i$ for some index i. Furthermore there exists a real valued vector $\boldsymbol{v} \in \mathbb{R}^3$ such that

$$V^\pi(s) = (\phi(s))^\top \boldsymbol{v} = \boldsymbol{e}_i^\top \boldsymbol{v}. \tag{3}$$

Because the state representation ϕ outputs one-hot bit vectors, each entry of the vector $\boldsymbol{v}$ contains the state value associated with one of the three columns. Similarly, the state representation ϕ can be used for Q-value predictions and for each action a, $Q^\pi(s,a) = \boldsymbol{e}_i^\top \boldsymbol{q}_a$, where $\boldsymbol{q}_a \in \mathbb{R}^3$. This article refers to such state representations that serve as basis functions (Sutton, 1996; Konidaris et al., 2011) for accurate value or Q-value predictions as *value-predictive*. Because value-predictive state representations are only required to be predictive of a particular policy π, they implicitly depend on the policy π.

2.2 Reward-Predictive State Representations

A reward-predictive state representation constructs a latent state space that retains enough information about the original state space to support accurate predictions of future expected reward outcomes. Figure 1(c) shows a trajectory that starts at grid cell (1,1) and follows an action sequence to produce a reward sequence of $0, 0, 1$. Because the state representation constructs a latent MDP that is a 3×1 grid world, a trajectory starting at the latent grid cell $\boldsymbol{\phi}_1$ and following the same action sequence results in a latent state sequence of $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3$. If a positive reward is associated with entering the state $\boldsymbol{\phi}_3$, the latent grid world would also produce a reward sequence of $0, 0, 1$. For any start state and any arbitrary action sequence, both the latent grid world and the original grid world produce equal reward sequences. This property can be formalized using a family of functions $\{f_t\}_{t \in \mathbb{N}}$ that predicts the expected reward outcome after executing an action sequence $a_1, ..., a_t$ starting at state s:

$$f_t : (s, a_1, ..., a_t) \mapsto \mathbb{E}_p \left[r_t | s, a_1, ..., a_t \right]. \tag{4}$$

The expectation in Equation (4) is computed over all trajectories that start in state s and follow the action sequence $a_1, ..., a_t$. A state representation is *reward-predictive* if the function f_t can be re-parameterized in terms of the constructed latent state space and if there exists a family of functions $\{g_t\}_{t \in \mathbb{N}}$ such that

$$\forall t \geq 1, s, a_1, ..., a_t, \;\; f_t(s, a_1, ..., a_t) = g_t(\phi(s), a_1, ..., a_t). \tag{5}$$

Because reward-predictive state representations are designed to produce accurate predictions of future expected reward outcomes, they need to encode information about both the transition and reward functions. Unlike value-predictive state representations, reward-predictive state representations are independent of a particular policy.

2. A one-hot bit vector $\boldsymbol{e}_i$ is a column vector of zeros but with the ith entry set to one. Vectors are denoted with bold lower-case letters. Matrices are denoted with bold capitalized letters.

2.3 Learning State Representations

Figure 1 presents an example where the same state representation is both value- and reward-predictive. In this article, we will present the distinctions and connection points between value-predictive and reward-predictive state representations. Further, we will discuss learning algorithms that search the space of all possible state representations to identify approximations of value- or reward-predictive state representations. The following results and examples demonstrate that value- and reward-predictive state representations can generalize across states very differently. Specifically, re-using a reward-predictive state representation ϕ across tasks can accelerate learning because a model-based agent only needs to construct the function g_t (which is defined on a small latent state space) for the new task instead of re-learning the function f_t from scratch. To ensure a fair comparison between different approximations, this article assumes that the dimensionality of the latent state space or the number of latent states is a fixed hyper-parameter.

3. Successor Features

Successor features (Barreto et al., 2017a) are a generalization of the Successor Representation (SR) (Dayan, 1993). The SR can be defined as follows: For finite state and action spaces, the transition probabilities while selecting actions according to a policy π can be written as a stochastic transition matrix $\boldsymbol{P}^\pi$. If the start state with index s is represented as a one-hot bit vector $\boldsymbol{e}_s$, the probability distribution of reaching a state after one time step of executing policy π can be written as a row vector $\boldsymbol{e}_s^\top \boldsymbol{P}^\pi$. After t time steps, the probability distribution over states can be written as a vector $\boldsymbol{e}_s^\top (\boldsymbol{P}^\pi)^t$. Suppose the path across the state space has a random length that follows the Geometric distribution with parameter γ: At each time step, a biased coin is flipped and the path continues with probability γ. In this model, the probability vector of reaching different states in t time steps is $(1-\gamma)\gamma^{t-1}\boldsymbol{e}_s^\top (\boldsymbol{P}^\pi)^t$. Omitting the factor $(1-\gamma)$, the SR recursively computes the marginal probabilities over all time steps:

$$\boldsymbol{\Psi}^\pi = \sum_{t=1}^{\infty} \gamma^{t-1} (\boldsymbol{P}^\pi)^{t-1} = \boldsymbol{I} + \gamma \boldsymbol{P}^\pi \boldsymbol{\Psi}^\pi. \tag{6}$$

Each entry (i, j) of the matrix $(1-\gamma)\boldsymbol{\Psi}^\pi$ contains the marginal probability across all possible durations of transitioning from state i to state j. Intuitively, the entry (i, j) of the matrix $\boldsymbol{\Psi}^\pi$ can be understood as the frequency of encountering state j when starting a path at state i and following the policy π. An *action conditioned SR* describes the marginal probability across all possible durations of transitioning from state i to state j, but first a particular action a is selected, and then a policy π is followed:

$$\boldsymbol{\Psi}_a^\pi \overset{\text{def.}}{=} \sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{P}_a (\boldsymbol{P}^\pi)^{t-2} = \boldsymbol{I} + \gamma \boldsymbol{P}_a \boldsymbol{\Psi}^\pi, \tag{7}$$

where $\boldsymbol{P}_a$ is the stochastic transition matrix describing all transition probabilities when action a is selected. Because $\boldsymbol{P}_a$ is a stochastic matrix, it can be shown that $\boldsymbol{\Psi}^\pi$ is invert-

ible and that there exists a one-to-one correspondence between each transition matrix and action-conditional SR matrix.[3]

SFs combine this idea with arbitrary state representations. Given a state representation ϕ, the SF is a column vector defined for each state and action pair (Kulkarni et al., 2016; Zhang et al., 2017) and

$$\boldsymbol{\psi}^{\pi}(s,a) = \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{\phi}_{s_t}\middle| s_1 = s, a_1 = a\right], \tag{8}$$

where the expectation in Equation (8) is computed over all infinite length trajectories that start at state s with action a and then follow the policy π. The output at state s of the vector-valued state-representation function ϕ is denoted by $\boldsymbol{\phi}_s$. We will refer to this vector $\boldsymbol{\phi}_s$ as either the latent vector or latent state. If following the policy π leads to encountering a particular latent vector $\boldsymbol{\phi}_{s'}$ many times, then this latent vector will occur in the summation in Equation (8) many times.[4] Depending on the state representation ϕ, the vector $\boldsymbol{\psi}^{\pi}(s,a)$ will be more similar to the latent vector $\boldsymbol{\phi}_{s'}$ and $\boldsymbol{\psi}^{\pi}(s,a)$ will be more dis-similar to a latent vector $\boldsymbol{\phi}_{\tilde{s}}$ if $\tilde{s}$ is a state that cannot be reached from the state s. A SF vector $\boldsymbol{\psi}^{\pi}$ can be understood as a statistic measuring how frequently different latent states vectors are encountered.

The following two sections draw connections between SFs, reward-predictive, and value-predictive state representations and outline under what assumptions learning SFs is equivalent to learning reward- or value-predictive state representations.

4. Connections to Reward-Predictive Representations

A reward-predictive state representation constructs a latent state space that is rich enough to produce predictions of future expected reward outcomes for any arbitrary actions sequence. For accurate predictions, the empirical transition probabilities between latent states have to mimic transitions in the original task. Figure 2 presents a reward-prediction example where only one action is available to the agent. In this task, the goal is to predict that a positive reward is obtained in three time steps if the agent starts at state s_1. This example compares two different state representations, the representation ϕ, which does not compress the state space, and $\tilde{\phi}$, which merges the first two states into one latent state. These two state representations lead to different empirical latent transition probabilities. While the first representation preserves the deterministic transitions of the task, the second representation does not. If states s_1 and s_2 are mapped to the same latent state $\tilde{\boldsymbol{\phi}}_1$, then a transition from state s_1 to s_2 appears as a self-loop from latent state $\tilde{\boldsymbol{\phi}}_1$ to itself and a transition from s_2 to s_3 appears as a transition from $\tilde{\boldsymbol{\phi}}_1$ to $\tilde{\boldsymbol{\phi}}_2$. Because the state representation ϕ constructs a latent state space with empirical latent transition probabilities that match the transition probabilities of the original task, this state representation is reward predictive.

3. By Equation (6), $\boldsymbol{\Psi}^{\pi} = \boldsymbol{I} + \gamma \boldsymbol{P}^{\pi}\boldsymbol{\Psi}^{\pi} \iff \boldsymbol{I} = (\boldsymbol{I} - \gamma \boldsymbol{P}^{\pi})\boldsymbol{\Psi}^{\pi} \iff (\boldsymbol{\Psi}^{\pi})^{-1} = \boldsymbol{I} - \gamma \boldsymbol{P}^{\pi}$. Equation (7) outlines how to construct $\boldsymbol{\Psi}^{\pi}_{a}$ from $\boldsymbol{P}_a$ for all actions. The reverse direction follows from $\boldsymbol{\Psi}^{\pi}_{a} = \boldsymbol{I} + \gamma \boldsymbol{P}_a \boldsymbol{\Psi}^{\pi} \iff (\boldsymbol{\Psi}^{\pi}_{a} - \boldsymbol{I})(\boldsymbol{\Psi}^{\pi})^{-1}/\gamma = \boldsymbol{P}_a$.

4. The remainder of the article will list function arguments in the subscript of a symbol and $\boldsymbol{\phi}_s$ always denotes the output of ϕ at state s.

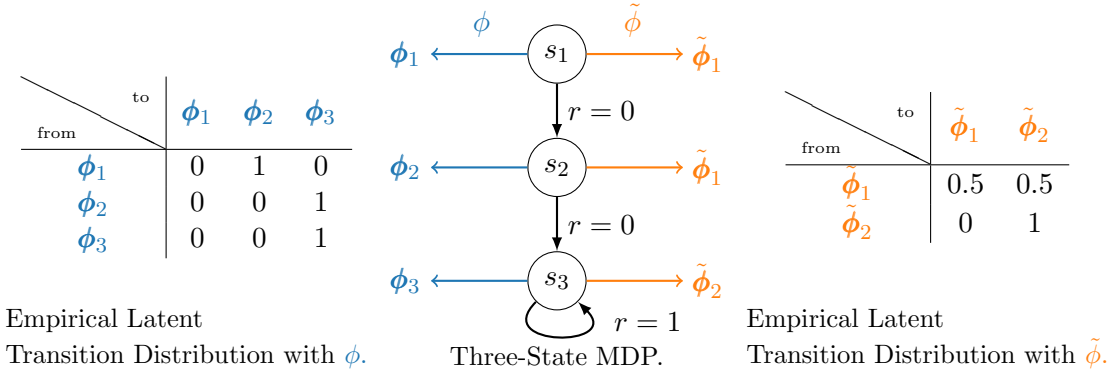
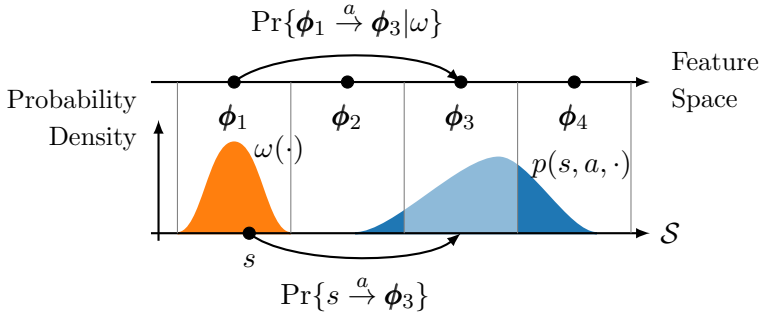


Figure 2: Three-State MDP Example. The centre schematic shows a single action three-state MDP with deterministic transitions (black arrows). Only the self-looping transition at state s_3 is rewarded. The two state representations ϕ and $\tilde{\phi}$ map the three states to different feature vectors, resulting in different empirical feature-transition probabilities. These probabilities are computed from observed trajectories that start at state s_1.



$\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$: The probability of transitioning from s to any state mapped to $\boldsymbol{\phi}_3$ by selecting action a.

$\Pr\{\boldsymbol{\phi}_1 \xrightarrow{a} \boldsymbol{\phi}_3 | \omega\}$: Empirical probability of transitioning from latent state $\boldsymbol{\phi}_1$ to latent state $\boldsymbol{\phi}_3$ by selecting action a.

Figure 3: Empirical Latent Transition Probabilities Depend on State-Visitation Frequencies. This example illustrates how empirical latent transition probabilities depend on state-visitation frequencies. In this example, the state space $\mathcal{S}$ is a bounded interval in $\mathbb{R}$ that is clustered into one of four latent states: $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \boldsymbol{\phi}_3$, or $\boldsymbol{\phi}_4$. State-visitation frequencies are modelled for each partition independently using the density function ω. The schematic plots the density function p over states of selecting action a at state s (blue area) and the density function ω over the state partition $\boldsymbol{\phi}_1$ (orange area). The probability $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$ of transitioning into the partition $\boldsymbol{\phi}_3$ is the blue shaded area. The probability $\Pr\{\boldsymbol{\phi}_1 \xrightarrow{a} \boldsymbol{\phi}_3 | \omega\}$ of a transition from $\boldsymbol{\phi}_1$ to $\boldsymbol{\phi}_3$ occurring is the marginal of $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$ over all states s mapping to $\boldsymbol{\phi}_1$, weighted by ω.

A reward-predictive state representation can be used in conjunction with a Linear Action Model (Sutton et al., 2008; Yao and Szepesvári, 2012) to compute expected future reward outcomes.

Definition 1 (Linear Action Model (LAM)). *Given an MDP and a state representation $\phi : \mathcal{S} \to \mathbb{R}^n$, a LAM consists of a set of matrices and vectors $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, where $\boldsymbol{M}_a$ is of size $n \times n$ and the column vector $\boldsymbol{w}_a$ is of dimension n.*

Given a fixed state representation, the transition matrices of a LAM $\{\boldsymbol{M}_a\}_{a\in\mathcal{A}}$ model the empirical latent transition probabilities and the vectors $\{\boldsymbol{w}_a\}_{a\in\mathcal{A}}$ model a linear map from latent states to expected one-step reward outcomes. The expected reward outcome after following the action sequence $a_1, ..., a_t$ starting at state s can then be approximated with

$$\mathbb{E}_p\left[r_t | s, a_1, ..., a_t\right] \approx \boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} \cdots \boldsymbol{M}_{a_{t-1}} \boldsymbol{w}_{a_t}. \tag{9}$$

The following sections will address how a state representation ϕ and a LAM can be found to predict expected future reward outcomes as accurately as possible. Because this article's goal is to establish different connections between learning successor features and model-based RL and to demonstrate that the learned reward-predictive state representations are suitable for transfer across variations in transitions and rewards, an extension of these model to non-linear latent transition and reward functions is left to future work.

To tie SFs to reward-predictive state representations, we first introduce a set of square real-valued matrices $\{\boldsymbol{F}_a\}_{a\in\mathcal{A}}$ such that, for every state s and action a,

$$\boldsymbol{\phi}_s^\top \boldsymbol{F}_a \approx \boldsymbol{\psi}^\pi(s, a) \tag{10}$$

$$= \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} \boldsymbol{\phi}_{s_t} \middle| s_1 = s, a_1 = a\right] \tag{11}$$

where the policy π is defined on the latent state space. A Linear Successor Feature Model (LSFM) is then defined using the matrices $\{\boldsymbol{F}_a\}_{a\in\mathcal{A}}$:

Definition 2 (Linear Successor Feature Model (LSFM)). *Given an MDP, a policy π, and a state representation $\phi : \mathcal{S} \to \mathbb{R}^n$, an LSFM consists of a set of matrices and vectors $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, where $\boldsymbol{F}_a$ is of size $n \times n$ and the column vector $\boldsymbol{w}_a$ is of dimension n. The matrices $\{\boldsymbol{F}_a\}_{a\in\mathcal{A}}$ are used to model a linear map from latent state features to SFs as described in Equation* (10).

LSFMs require the state representation ϕ to (linearly) approximate the SF $\boldsymbol{\psi}^\pi(s, a)$ using the matrices $\{\boldsymbol{F}_a\}_{a\in\mathcal{A}}$ (Equation (10)). Previously presented SF frameworks (Barreto et al., 2017a, 2018) do not use the state representation ϕ to approximate SF vectors $\boldsymbol{\psi}^\pi$ as described in Equation (10) and construct the state-representation function ϕ differently, for example, by setting the output of ϕ to be one-step rewards (Barreto et al., 2017b). In contrast, LSFMs are used to learn the state-representation function ϕ that satisfies Equation (10). Because LSFMs distinctly incorporate this approximative property, SFs can be connected to model-based RL.

An intelligent agent that uses a state representation ϕ operates directly on the constructed latent state space and is constrained to only search the space of policies that are defined on its latent state space. These policies are called *abstract policies*.

Definition 3 (Abstract Policies)**.** *An abstract policy π_ϕ is a function mapping latent state and action pairs to probability values:*

$$\forall s \in \mathcal{S}, a \in \mathcal{A}, \pi_\phi(\boldsymbol{\phi}_s, a) \in [0,1] \text{ and } \sum_a \pi_\phi(\boldsymbol{\phi}_s, a) = 1.$$

For a fixed state representation ϕ, the set of all abstract policies is denoted with Π_ϕ.

The following sections first tie learning LAMs to reward-predictive state representations. We then show that learning LSFMs is equivalent to learning LAMs tying LSFMs to reward-predictive state representations.

4.1 Encoding Bisimulation Relations

To ensure accurate predictions of future reward outcomes, the previous discussion suggests that the empirical latent transition probabilities have to match the transition probabilities in the original task. Figure 3 presents a schematic explaining these dependencies further. In this example, the state space is a bounded interval in $\mathbb{R}$ that is mapped to four different latent states, $\boldsymbol{\phi}_1$, $\boldsymbol{\phi}_2$, $\boldsymbol{\phi}_3$, or $\boldsymbol{\phi}_4$. The probability of transitioning from the state s to any state that is mapped to $\boldsymbol{\phi}_3$ is denoted with $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$. This probability $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$ is the marginal over all states s' that are mapped to the latent state $\boldsymbol{\phi}_3$. Assume that ω is a density function over all states that are mapped to the latent state $\boldsymbol{\phi}_1$. This density function could model the visitation frequencies of different states as an intelligent agent interacts with the MDP. The empirical probability of transitioning from latent state $\boldsymbol{\phi}_1$ to $\boldsymbol{\phi}_3$ is then the marginal over all states mapping to $\boldsymbol{\phi}_1$ and

$$\Pr\left\{\boldsymbol{\phi}_1 \xrightarrow{a} \boldsymbol{\phi}_3 \middle| \omega\right\} = \int_{s:\phi(s)=\boldsymbol{\phi}_1} \omega(s) \Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\} ds = \mathbb{E}_\omega \left[\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\} \middle| \phi(s) = \boldsymbol{\phi}_1\right]. \quad (12)$$

The expectation in Equation (12) is computed with respect to ω over all states s that map to the latent state $\boldsymbol{\phi}_1$. As Equation (12) outlines, the empirical transition probability $\Pr\{\boldsymbol{\phi}_1 \xrightarrow{a} \boldsymbol{\phi}_3 | \omega\}$ depends on the visitation frequencies ω. The probability $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$ of transitioning from a state s into a partition only depends on the transition function p itself.

Consider two different states s and $\tilde{s}$ that map to the same latent state and $\phi(s) = \phi(\tilde{s})$. If the state representation is constructed such that

$$\forall a, \forall \boldsymbol{\phi}_i,\ \Pr\{s \xrightarrow{a} \boldsymbol{\phi}_i\} = \Pr\{\tilde{s} \xrightarrow{a} \boldsymbol{\phi}_i\} \text{ and } \mathbb{E}_p\left[r(s,a,s') \middle| s, a\right] = \mathbb{E}_p\left[r(\tilde{s},a,s') \middle| \tilde{s}, a\right], \quad (13)$$

then the empirical latent state transition probabilities would become independent of ω because the integrand in Equation (12) is constant and

$$\Pr\left\{\boldsymbol{\phi}_1 \xrightarrow{a} \boldsymbol{\phi}_3 \middle| \omega\right\} = \int_{s:\phi(s)=\boldsymbol{\phi}_1} \omega(s) \underbrace{\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}}_{\text{constant}} ds = \Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}. \quad (14)$$

Equation (14) follows directly from the transition condition in line (13), because the probability $\Pr\{s \xrightarrow{a} \boldsymbol{\phi}_3\}$ is constant for all states s that are mapped to the latent state vector $\boldsymbol{\phi}_1$. If the two identities in line (13) hold, then the resulting latent state space constructs latent transition probabilities that correspond to the transition probabilities in the original task.

Equation (13) describes an informal definition of bisimulation (Givan et al., 2003). Definition 4 listed in Appendix A.1 presents a formal measure theoretic definition of bisimulation on arbitrary (measurable) state spaces. This definition is used to prove the theorems stated in this section. To prove that LAMs encode state representations that generalize only across bisimilar states, two assumptions are made.

Assumption 1. *The state space $\mathcal{S}$ of an MDP can be partitioned into at most n different partitions of bisimilar states, where n is a natural number.*

Assumption 2. *A state representation $\phi : \mathcal{S} \to \{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$ is assumed to have a range that consists of all n one-hot bit vectors. For each i, there exists a state s such that $\phi(s) = \boldsymbol{e}_i$.*

Assumption 1 is not particularly restrictive in a learning context: If an agent has observed n distinct states during training, then a state representation assigning each state to one of n different one-hot bit vectors can always be constructed. While doing so may not be useful to generalize across different states, this argument suggests that Assumption 1 is not restrictive in practice. Assumption 2 is relaxed in the following sections.

If action a is selected at state s, the expected next feature vector is

$$\mathbb{E}_p\left[\phi(s') \middle| s, a\right] = \sum_{j=1}^{n} \Pr\{s \xrightarrow{a} \boldsymbol{e}_j\} \boldsymbol{e}_j = \left[\cdots, \Pr\{s \xrightarrow{a} \boldsymbol{e}_j\}, \cdots\right]^\top. \tag{15}$$

The expected value in Equation (15) is computed over all possible next states s' that can be reached from state s by selecting action a. In Equation (15), the next state s' is a random variable whose probability distribution or density function is described by the MDP's transition function p. By Assumption 2, each state is mapped to some one-hot bit vector $\boldsymbol{e}_j$. Because there are only n different one-hot bit vectors of dimension n, the summation in Equation (15) is finite. Each entry of the resulting vector in Equation (15) stores the probability $\Pr\{s \xrightarrow{a} \boldsymbol{e}_j\}$ of observing the feature vector $\boldsymbol{e}_j$ after selecting action a at state s.

Because the expected next feature vector $\mathbb{E}_p\left[\phi(s')|s, a\right]$ is a probability vector, the transition matrices $\{\boldsymbol{M}_a\}_{a\in\mathcal{A}}$ of a LAM are stochastic: If $\phi(s) = \boldsymbol{e}_i$ and $\boldsymbol{e}_i^\top \boldsymbol{M}_a = \mathbb{E}_p\left[\boldsymbol{e}_j|s, a\right]$, then the ith row of the matrix $\boldsymbol{M}_a$ is equal to the probability vector shown in Equation (15). If $\boldsymbol{e}_i^\top \boldsymbol{w}_a = \mathbb{E}_p\left[r(s, a, s')|s, a\right]$, then the weight vectors of a LAM $\{\boldsymbol{w}_a\}_{a\in\mathcal{A}}$ encode a reward table. These observations lead to the first theorem.[5]

Theorem 1. *For an MDP $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, let $\phi : \mathcal{S} \to \{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$ be a state representation and $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ a LAM. Assume that $\mathcal{S}$ can be partitioned into at most n partitions of bisimilar states. If the state representation ϕ satisfies*

$$\forall s \in \mathcal{S}, \forall a \in \mathcal{A},\ \boldsymbol{\phi}_s^\top \boldsymbol{w}_a = \mathbb{E}_p\left[r(s, a, s') \middle| s, a\right] \ \text{ and } \ \boldsymbol{\phi}_s^\top \boldsymbol{M}_a = \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}|s, a\right], \tag{16}$$

then ϕ generalizes across bisimilar states and any two states s and $\tilde{s}$ are bisimilar if $\boldsymbol{\phi}_s = \boldsymbol{\phi}_{\tilde{s}}$.

The proof of Theorem 1 uses the fact that the expected value of one-hot bit vectors encode exact probability values. A similar observation can be made about the SFs for a

5. Appendix A.1 presents formal proofs for all presented theorems.

one-hot bit-vector state representation. In this case, the $(1-\gamma)$ rescaled SF contains the marginal of reaching a state partition across time steps:

$$(1-\gamma)\mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{e}_t\middle|s,a_1\right]=\left[...,\sum_{t=1}^{\infty}(1-\gamma)\gamma^{t-1}\mathbb{E}_{p,\pi}\left[\Pr\left\{s\overset{a_1\cdots a_t}{\longrightarrow}\boldsymbol{e}_i\right\}\middle|s,a_1\right],...\right]^{\top}, \quad (17)$$

where the expectation in Equation (17) is computed over infinite length trajectories starting at state s with action a. This observation leads to the following theorem stating that LSFMs can be used to identify a one-hot state representation that generalizes across bisimilar states.

Theorem 2. *For an MDP $\langle\mathcal{S},\mathcal{A},p,r,\gamma\rangle$, let $\phi:\mathcal{S}\to\{\boldsymbol{e}_1,...,\boldsymbol{e}_n\}$ be a state representation and $\{\boldsymbol{F}_a,\boldsymbol{w}_a\}_{a\in\mathcal{A}}$ an LSFM. If, for one policy $\pi\in\Pi_\phi$, the representation ϕ satisfies*

$$\forall s\in\mathcal{S},\forall a\in\mathcal{A},\ \boldsymbol{\phi}_s^{\top}\boldsymbol{w}_a=\mathbb{E}_p\left[r(s,a,s')\middle|s,a\right]\ \text{ and }\boldsymbol{\phi}_s^{\top}\boldsymbol{F}_a=\boldsymbol{\phi}_s^{\top}+\gamma\mathbb{E}_{p,\pi}\left[\boldsymbol{\phi}_{s'}\boldsymbol{F}_{a'}|s,a\right], \quad (18)$$

then ϕ generalizes across bisimilar states and any two states s and $\tilde{s}$ are bisimilar if $\boldsymbol{\phi}_s=\boldsymbol{\phi}_{\tilde{s}}$. If Equation (18) holds for one policy $\pi\in\Pi_\phi$, then Equation (18) also holds every other policy $\tilde{\pi}\in\Pi_\phi$ as well.

Equation (18) describes a fixed-point equation similar to the Bellman fixed-point equation:

$$\begin{aligned}\boldsymbol{e}_s^{\top}\boldsymbol{F}_a&=\mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{e}_{s_t}\middle|s,a\right] &(19)\\ &=\boldsymbol{e}_s^{\top}+\gamma\mathbb{E}_p\left[\mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{e}_{s_t}\middle|s',a'\right]\middle|s,a\right] &(20)\\ &=\boldsymbol{e}_s^{\top}+\gamma\mathbb{E}_p\left[\boldsymbol{e}_{s'}^{\top}\boldsymbol{F}_{a'}\middle|s,a\right]. &(21)\end{aligned}$$

Finding a policy $\pi\in\Pi_\phi$ to test if Equation (18) holds for a state representation ϕ is trivial, because it is sufficient to test the state representation for any single policy. Theorems 1 and 2 show that LAMs and LSFMs can be used to identify one-hot reward-predictive state representations. To arrive at an algorithm that can learn reward-predictive state representations, the following sections convert the conditions outlined in Theorems 1 and 2 into learning objectives. The next section presents an analysis showing how violating these conditions by some margin results in increased reward-sequence prediction errors. We refer to state representations that can only approximately predict expected reward sequences as an *approximate reward-predictive state representation.*

4.2 Approximate Reward-Predictive State Representations

In this section, we analyze to what extent a state representation is reward predictive if it only approximately satisfies the conditions outlined in Theorems 1 and 2. In addition, we will also generalize beyond one-hot representations and relax Assumption 2 by considering state representations that map the state space into $\mathbb{R}^n$. The latent feature's dimension n is considered a fixed hyper-parameter.

Because LAMs only model one-step transitions but are used to predict entire reward sequences, the scale and expansion properties of the constructed latent state space influences how prediction errors scale and compound (Talvitie, 2018; Asadi et al., 2018). Define the following variables:[6]

$$W = \max_{a \in \mathcal{A}} ||\boldsymbol{w}_a||, \;\; M = \max_{a \in \mathcal{A}} ||\boldsymbol{M}_a||, \;\; N = \sup_{s \in \mathcal{S}} ||\boldsymbol{\phi}_s||. \tag{22}$$

To identify approximate reward-predictive state representations, a state representation ϕ is analyzed by its one-step reward-prediction error and one-step expected transition error. These quantities are computed using a LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ and are defined as

$$\varepsilon_r = \sup_{s,a} \left| r(s,a) - \boldsymbol{\phi}_s^\top \boldsymbol{w}_a \right| \text{ and} \tag{23}$$

$$\varepsilon_p = \sup_{s,a} \left| \left| \mathbb{E}_p \left[\boldsymbol{\phi}_{s'}^\top \middle| s, a \right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \right| \right|. \tag{24}$$

Equivalently, a state representation is also analyzed using an LSFM that predicts the SF for a policy that selects actions uniformly at random. For an LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$, define

$$\overline{\boldsymbol{F}} = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \boldsymbol{F}_a. \tag{25}$$

For such an LSFM, the linear SF prediction error is defined as

$$\varepsilon_\psi = \sup_{s,a} \left| \left| \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'}^\top \overline{\boldsymbol{F}} \middle| s, a \right] - \boldsymbol{\phi}_s^\top \boldsymbol{F}_a \right| \right|. \tag{26}$$

Because the matrix $\overline{\boldsymbol{F}}$ averages across all actions, the LSFM computes the SFs for a policy that selects actions uniformly at random. Here, we focus on uniform random action selection to simplify the analysis. While the matrix $\overline{\boldsymbol{F}}$ could be constructed differently, the proofs of the theoretical results presented in this section assume that $\overline{\boldsymbol{F}}$ can only depend on the matrices $\{\boldsymbol{F}_a\}_{a \in \mathcal{A}}$ and is not a function of the state s.

Similar to the previous discussion, LSFMs are closely related to LAMs and the one-step transition error ε_p can be upper bounded by the linear SF error ε_ψ. We define the quantities ε_r, ε_t, and ε_ψ to upper bound the magnitude with which the identities provided in Theorems 1 and 2 are violated. The following analysis generalizes the previously presented results by showing that, if a state representation approximately satisfies the requirements of Theorems 1 and 2, then this state representation is approximately reward predictive.

Lemma 1. *For an MDP, a state representation ϕ, an LSFM and a LAM,*

$$\varepsilon_p \leq \varepsilon_\psi \frac{1 + \gamma M}{\gamma} + C_{\gamma,M,N} \Delta, \tag{27}$$

where $C_{\gamma,M,N} = (1+\gamma)(1+\gamma M)N/(\gamma(1-\gamma M))$ and $\Delta = \max_a ||\boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a||$.

6. All norms are assumed to be L_2. The Euclidean norm is used for vectors. The norm of a matrix $\boldsymbol{M}$ is computed with $||\boldsymbol{M}|| = \sqrt{\sum_{i,j} \boldsymbol{M}(i,j)^2}$, where the summation ranges over all matrix entries $\boldsymbol{M}(i,j)$.

Lemma 1 presents a bound stating that if an LSFM has low linear SF prediction errors, then a corresponding LAM can be constructed with low one-step transition error ε_p, assuming that Δ is close to zero. If $\Delta = 0$, then the matrices $\{\boldsymbol{F}_a\}_{a\in\mathcal{A}}$ can be thought of as action-conditional SR matrices for the transition matrices $\{\boldsymbol{M}_a\}_{a\in\mathcal{A}}$. In Section 3, Equation (7), the action-conditional SR matrix is defined as $\boldsymbol{\Psi}_a^\pi = \boldsymbol{I} + \gamma \boldsymbol{P}_a \boldsymbol{\Psi}^\pi$, where $\boldsymbol{P}_a$ is a stochastic transition matrix for a finite MDP. Furthermore, Section 3 also shows that there exists a bijection between the transition matrices $\{\boldsymbol{P}_a\}_{a\in\mathcal{A}}$ and the action-conditional SR matrices $\{\boldsymbol{\Psi}_a^\pi\}_{a\in\mathcal{A}}$. Similarly, if $\Delta = 0$, then

$$\boldsymbol{F}_a = \boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}}. \tag{28}$$

In fact, the proof of Lemma 1 first proceeds by assuming $\Delta = 0$ and showing a one-to-one correspondence between the LSFM matrices $\{\boldsymbol{F}_a\}_{a\in\mathcal{A}}$ and the LAM's transition matrices $\{\boldsymbol{M}_a\}_{a\in\mathcal{A}}$. For arbitrary state representations and LSFMs, Equation (28) may not hold and $\Delta > 0$.

The following theorem presents a bound stating that low one-step reward and one-step transition errors lead to state representations that support accurate predictions of future expected reward outcomes. By Lemma 1, the following results also apply to LSFMs because low linear SF prediction errors lead to low one-step expected transition errors.

Theorem 3. *For an MDP, state representation $\phi \to \mathbb{R}^n$, and for all $T \geq 1, s, a_1, ..., a_T$,*

$$\left| \boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} \cdots \boldsymbol{M}_{a_{T-1}} \boldsymbol{w}_{a_T} - \mathbb{E}_p \left[r_T | s, a_1, ..., a_T \right] \right| \leq \varepsilon_p \sum_{t=1}^{T-1} M^t W + \varepsilon_r. \tag{29}$$

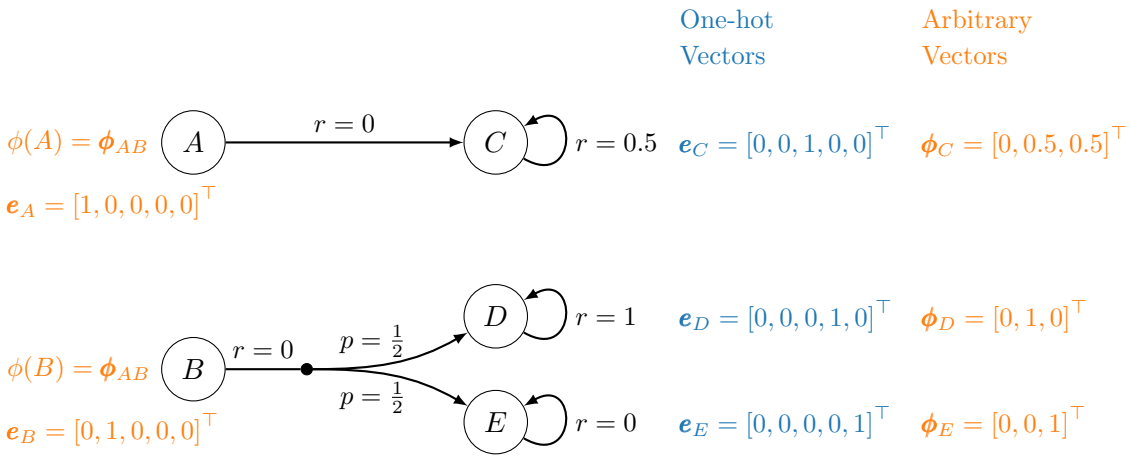
Theorem 3 shows that prediction errors of expected rollouts are bounded linearly in ε_r and ε_p and prediction errors tend to zero as ε_r and ε_p tend to zero. Because LAMs are one-step models, prediction errors increase linearly with T if $M \leq 1$ as the model is used to generalize over multiple time steps. Prediction errors may increase exponentially if the transition matrices are expansions and $M > 1$, similar to previously presented bounds (Asadi et al., 2018).

The following theorem bounds the prediction error of finding a linear approximation of the Q-function $Q^\pi(s,a) \approx \boldsymbol{\phi}_s^\top \boldsymbol{q}_a$ using a state representation ϕ and a real valued vector $\boldsymbol{q}_a$.

Theorem 4. *For an MDP, state representation $\phi : \mathcal{S} \to \mathbb{R}^n$, any arbitrary abstract policy $\pi \in \Pi_\phi$, and LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, there exists vectors $\boldsymbol{v}^\pi$ and $\{\boldsymbol{q}_a = \boldsymbol{w}_a + \gamma \boldsymbol{M}_a \boldsymbol{v}^\pi\}_{a\in\mathcal{A}}$ such that, for all states s and actions a,*

$$\left| V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi \right| \leq \frac{\varepsilon_r + \gamma \varepsilon_p \left|\left| \boldsymbol{v}^\pi \right|\right|}{1-\gamma} \text{ and } \left| Q^\pi(s,a) - \boldsymbol{\phi}_s^\top \boldsymbol{q}_a \right| \leq \frac{\varepsilon_r + \gamma \varepsilon_p \left|\left| \boldsymbol{v}^\pi \right|\right|}{1-\gamma}. \tag{30}$$

By Theorem 4, an (approximate) reward-predictive state representation (approximately) generalizes across all abstract policies, because the same state representation can be used to predict the value of every possible abstract policy $\pi \in \Pi_\phi$. Prediction errors tend to zero as ε_r and ε_p tend to zero. The value prediction error bounds stated in Theorem 4 are similar to bounds presented by Bertsekas (2011) on approximate (linear) policy iteration, because the presented results also approximate value functions using a function that is linear in



(a) Reward-predictive representations can be encoded using different state features.

Model	Prediction Target	with one-hot	with arbitrary
LAM	$\mathbb{E}_p[\boldsymbol{\phi}\|A]$	$= [0,0,1,0,0]^\top$	$= [0, 0.5, 0.5]^\top$
	$\mathbb{E}_p[\boldsymbol{\phi}\|B]$	$= [0,0,0,0.5,0.5]^\top$	$= [0, 0.5, 0.5]^\top$
LSFM	$\mathbb{E}_p[\sum_{t=1}^{\infty} \gamma^{t-1}\boldsymbol{\phi}_t\|A]$	$= [1,0,9,0,0]^\top$	$= \boldsymbol{\phi}_{AB} + [0, 3.5, 3.5]^\top$
	$\mathbb{E}_p[\sum_{t=1}^{\infty} \gamma^{t-1}\boldsymbol{\phi}_t\|B]$	$= [0,1,0,3.5,3.5]^\top$	$= \boldsymbol{\phi}_{AB} + [0, 3.5, 3.5]^\top$

(b) Prediction targets of a LAM and LSFM for both state representations.

Figure 4: Real-valued reward-predictive state representations may not encode bisimulations, but support predictions of future expected reward outcomes. 4(a): In this five-state, example no states are bisimilar. Each edge is labelled with the reward given to the agent for a particular transition. The transition departing state B is probabilistic and leads to state D or E with equal probability. All other transitions are deterministic. Two different state representations are considered. One representation maps states to one-hot bit vectors and the other representation maps states to real-valued vectors. 4(b): Prediction targets for both LAM and LSFM depend on what state representation is used. For a one-hot state representation, the LAM and LSFM have different prediction targets for states A and B, because a one-hot bit-vector state representation can be used to detect that transition probabilities are different between A and B. In contrast, real valued state representations may lead to equal prediction targets for both LAM and LSFM, because the state features $\boldsymbol{\phi}_C$, $\boldsymbol{\phi}_D$, and $\boldsymbol{\phi}_E$ can hide different transition probabilities. The state representation ϕ is reward predictive and $\varepsilon_r = \varepsilon_p = \varepsilon_\psi = 0$.

some basis function ϕ. Conforming to these previously presented results on linear value function approximation, prediction errors scale linearly in one-step prediction errors ε_ψ and ε_p. Theorems 4 and 3 show that, by learning a state representation that predicts SFs for policies that select actions uniformly at random, an approximate reward-predictive state representation is obtained. This state representation generalizes across the entire space of abstract policies, because accurate predictions of each policy's value function are possible if prediction errors are low enough. Appendix A.2 presents formal proofs of Theorems 3 and 4.

Figure 4 presents an example highlighting that reward-predictive state representations do not necessarily encode bisimulation relations. In this example, states A and B are not bisimilar, because the probabilities with which they transition to C, D, or E are different. The state representation ϕ generalizes across these two states and $\varepsilon_r = \varepsilon_p = \varepsilon_\psi = 0$. The expected reward sequence for transitioning out of A or B is always $0, 0.5, 0.5, ...$, so both states have equal expected future reward outcomes and the state representation ϕ is reward predictive. However, the state representation is not predictive of the probability with which a particular reward sequence is observed. For example, the latent state space constructed by ϕ would have to make a distinction between state A and B to support predictions stating that a reward sequence of $0, 1, 1, ...$ can be obtained from state B with probability 0.5. The example in Figure 4 highlights the difference between the analysis presented in this section and the previous section: By relaxing Assumption 2 and considering state-representation functions that map states to real-valued vectors instead of one-hot bit vectors, one may still obtain a reward-predictive state representation, but this representation may not necessarily encode a bisimulation relation.

Note that an (approximate) reward-predictive state representation $\phi : \mathcal{S} \to \mathbb{R}^n$ could in principle map each state to a distinct latent state vector. As demonstrated by the following simulations, the idea behind generalizing across states is that the dimension n of the constructed latent space is sufficiently small to constrain the LSFM learning algorithm to assign approximately the same latent state vector to different states. The following section illustrates how (approximate) reward-predictive state representations model generalization across different states.

4.3 Learning Reward-Predictive Representations

Using the previously presented theoretical results, this section designs a loss objective to learn approximate reward-predictive state representations. By optimizing a loss function, a randomly chosen state-representation function $\phi : \mathcal{S} \to \mathbb{R}^n$ is iteratively improved until the function ϕ can be used to accurately predict future reward sequences. The cluster plots in Figure 5 illustrate this process: Starting with a random assignment of feature vectors to different grid states, a state representation is iteratively improved until all states of the same column are clustered approximately into the same latent state. These latent state vectors were only clustered because the loss function assesses whether a state representation is reward predictive. The fact that states of the same column are assigned approximately to the same latent state is an artifact of this optimization process. The hyper-parameter n can be understood as the size of the constructed latent space and a bound on the degree of compression of the state space. For example, if the state space consists of nine different

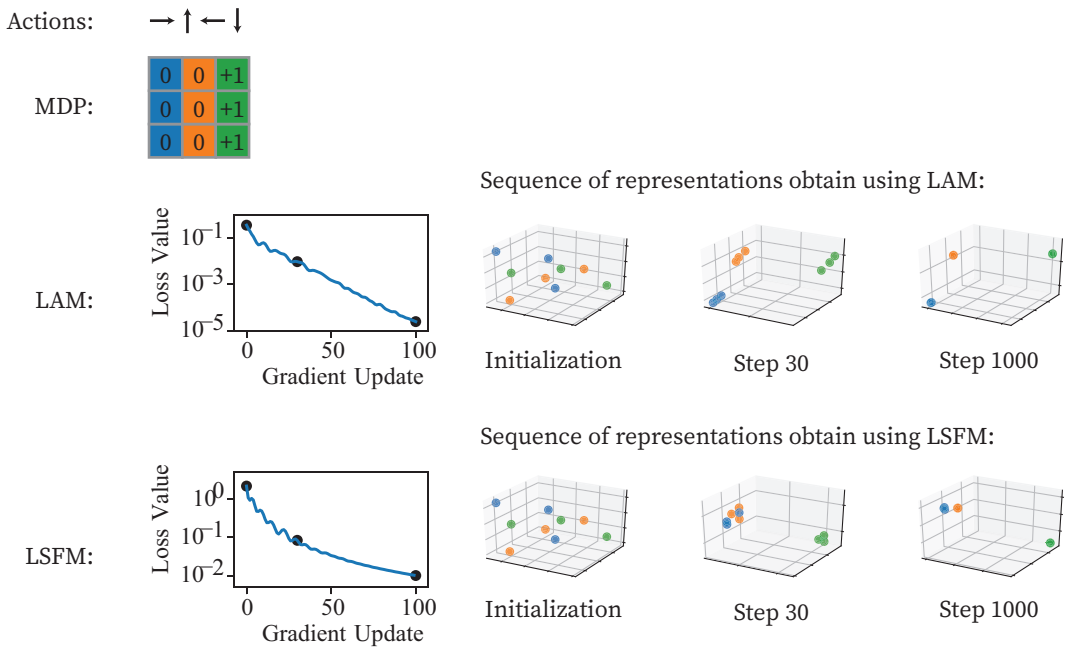


Figure 5: In the column world task, learning reward-predictive state representations leads to clustering grid cells by each column. The top row illustrates a map of the column world task and a colouring of each column. The middle row presents an experiment that optimizes across different state representations to find a LAM that can be used for accurate one-step reward and one-step expected transition predictions. Each latent state is plotted as a dot in 3D-space and dots are coloured by the column they correspond to. At the end of the optimization process, three clusters of the same colour are formed showing that approximately the same latent state is assigned to states of the same column. The third row repeats the same experiment using an LSFM, which assesses whether the constructed latent state space can be used for accurate one-step reward predictions and SF predictions. Appendix C.1 describes this experiment in detail.

states, setting $n = 9$ could result in not compressing the state space and mapping nine states onto nine distinct one-hot bit vectors. The following experiments explore how choosing a low enough feature dimension leads to compression and generalization across states.

The previous sections present bounds on prediction errors that are parameterized in the worst-case one-step reward-prediction error ε_r and worst-case linear SF prediction error ε_ψ. Given only a finite data set of transitions $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{D}$, it may not be possible to compute estimates for ε_r and ε_ψ without making further assumptions on the MDP at hand, such as a finite state space or a bounded Rademacher complexity of the transition and reward functions to obtain robust performance on a test data set (Mohri et al., 2018). Because the goal of this project is to study the connections between SFs and different models of generalization across different states, an analysis of how to find provably correct predictions of ε_r and ε_ψ given a finite data set $\mathcal{D}$ is beyond the scope of this article. Instead, the conducted experiments collect a data set $\mathcal{D}$ by sampling trajectories using a policy that selects actions uniformly at random. The data set $\mathcal{D}$ is generated to be large enough to cover all state and action pairs, ensuring that all possible transitions and rewards are implicitly represented in this data set. If the data set does not cover all states and actions, then the resulting reward-predictive state representation may only produce accurate predictions for some reward sequences because the data set does not communicate all aspects of the MDP at hand. A study of this interaction between a possibly limited training data set and the resulting model's ability to make accurate predictions is left for future work.

We design a loss objective to learn LSFMs $\mathcal{L}_{\text{LSFM}}$ that is the sum of three different terms: The first term $\mathcal{L}_r$ computes the one-step reward prediction error and is designed to minimize the reward error ε_r. The second term $\mathcal{L}_\psi$ computes the SF prediction error and is designed to minimize the SF error ε_ψ. The last term $\mathcal{L}_N$ is a regularizer constraining the gradient optimizer to find a state representation that outputs unit norm vectors. Empirically, we found that this regularizer encourages the optimizer to find a model with $M \approx 1$ and $W \approx 1$. (Reward-sequence prediction errors are lower for these values of M and W, as stated in Theorem 29.) Given a finite data set of transitions $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{D}$, the formal loss objective is

$$\mathcal{L}_{\text{LSFM}} = \underbrace{\sum_{i=1}^{D} \left(\boldsymbol{\phi}_{s_i}^\top \boldsymbol{w}_{a_i} - r_i\right)^2}_{=\mathcal{L}_r} + \alpha_\psi \underbrace{\sum_{i=1}^{D} \left|\left|\boldsymbol{\phi}_{s_i}^\top \boldsymbol{F} a - \vec{\boldsymbol{y}}_{s_i,a_i,r_i,s'_i}\right|\right|_2^2}_{=\mathcal{L}_\psi} + \alpha_N \underbrace{\sum_{i=1}^{D} \left(\left|\left|\boldsymbol{\phi}_{s_i}\right|\right|_2^2 - 1\right)^2}_{=\mathcal{L}_N}, \quad (31)$$

for a finite data set of transitions $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{D}$. In Equation (31), the prediction target

$$\vec{\boldsymbol{y}}_{s,a,r,s'} = \boldsymbol{\phi}_s^\top + \gamma \boldsymbol{\phi}_{s'}^\top \overline{\boldsymbol{F}}$$

and $\alpha_\psi, \alpha_{\text{N}} > 0$ are hyper-parameters. These hyper-parameters weigh the contribution of each error term to the overall loss objective. If α_ψ is set to too small a value, then the resulting state representation may only produce accurate one-step reward predictions, but not accurate predictions of longer reward sequences. This article presents simulations on finite state spaces and represents a state representation as a function $s \mapsto \boldsymbol{e}_s^\top \boldsymbol{\Phi}$ where $\boldsymbol{\Phi}$ is a weight matrix of size $|\mathcal{S}| \times n$. An approximation of a reward-predictive state representation is obtained by performing gradient descent on the loss objective $\mathcal{L}_{\text{LSFM}}$ with respect to

the free parameters $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ and $\boldsymbol{\Phi}$. For each gradient update, the target $\vec{\boldsymbol{y}}_{s,a,r,s'}$ is considered a constant. The previously presented bounds show that prediction errors also increase with $\Delta = \max_a ||\boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a||$. Minimizing $\mathcal{L}_\psi$ for a fixed state representation ϕ minimizes Δ, because

$$\Delta \leq c_\phi \mathcal{L}_\psi, \tag{32}$$

where c_ϕ is a non-negative constant. Appendix A.3 presents a formal proof for Equation (32).

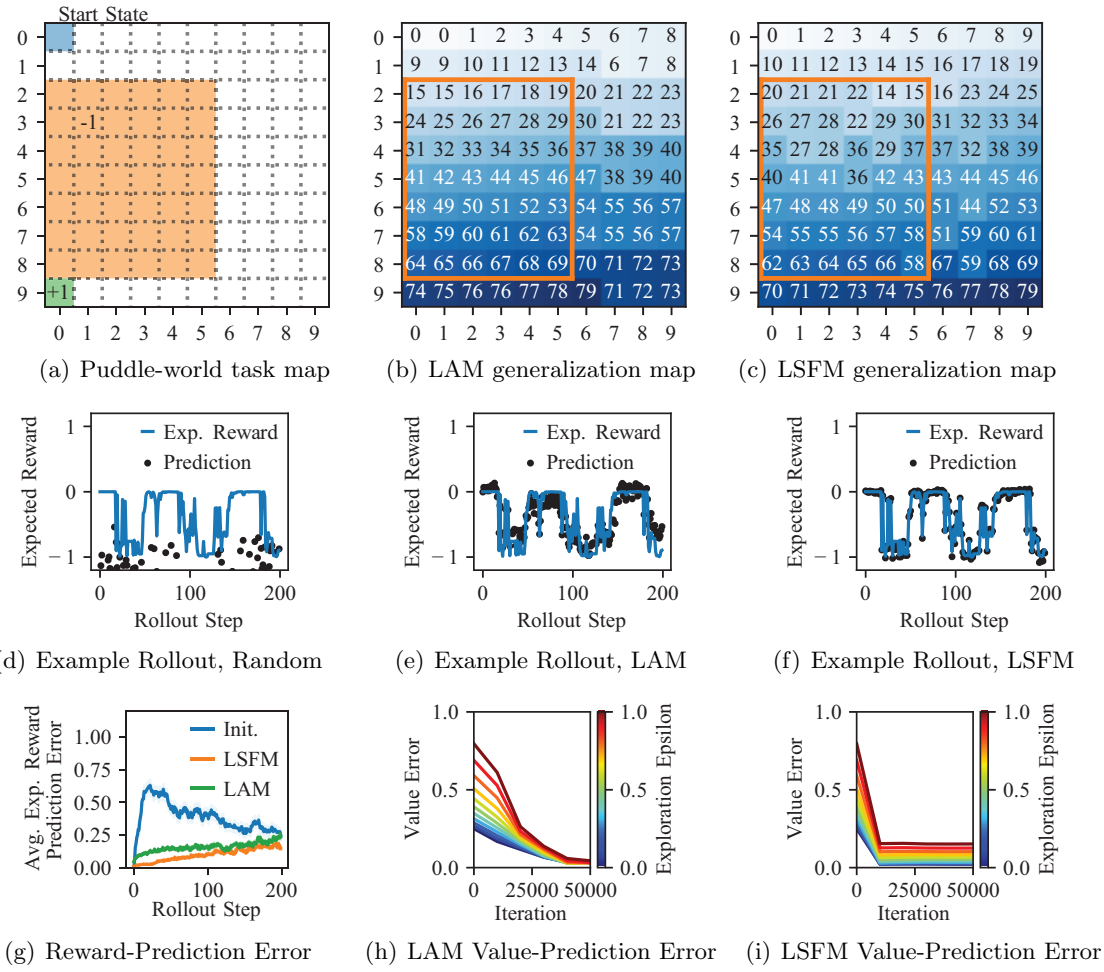
To assess whether minimizing the loss $\mathcal{L}_{\text{LSFM}}$ leads to approximating reward-predictive state representations, a transition data set was collected from the puddle-world task (Boyan and Moore, 1995). Conforming to the previous analysis, the LSFM is compared to a LAM that is trained using a similar loss function, described in Appendix C.2.

Figure 6 presents the puddle-world experiments and the results. In puddle-world (Figure 6(a)), the agent has to navigate from a start state to a goal to obtain a reward of one while avoiding a puddle. Entering the puddle is penalized with a reward of minus one. To predict future reward outcomes accurately, a state representation has to preserve the grid position as accurately as possible to predict the location of the different reward cells.

By constraining the latent state space to 80 dimensions, the optimization process is forced to find a compression of all 100 grid cells. To analyze across which states the learned reward-predictive state representation generalizes, all feature vectors were clustered using agglomerative clustering. Two different states that are associated with feature vectors $\boldsymbol{\phi}_s$ and $\boldsymbol{\phi}_{\tilde{s}}$ are merged into the same cluster if their Euclidean distance $||\boldsymbol{\phi}_s - \boldsymbol{\phi}_{\tilde{s}}||_2$ is low. For example, a randomly chosen representation would randomly assign states to different latent states and the partition map could assign the grid cell at $(0, 0)$ and $(9, 9)$ to the same latent state. Figures 6(b) and 6(c) plot the obtained clustering as a partition map. Grid cells are labelled with the same partition index if they belong to the same cluster and colours correspond to the partition index. To predict reward outcomes accurately, the position in the grid needs to be roughly retained in the constructed latent state space. The partition maps in Figures 6(b) and 6(c) suggest that the learned state representation extracts this property from a transition data set, by generalizing only across neighboring grid cells and tiling the grid space. Only a transition data set $\mathcal{D}$ was given as input to the optimization algorithm and the algorithm was not informed about the grid-world topology of the task in any other way.

Figures 6(d), 6(e), 6(f) plot an expected reward rollout and the predictions presented by a random initialization (6(d)), the learned representation using a LAM (6(e)), and the learned representation using a LSFM (6(f)). The blue curve plots the expected reward outcome $\mathbb{E}_p[r_t|s, a_1, ..., a_t]$ as a function of t for a randomly selected action sequence. Because transitions are probabilistic, the (blue) expected reward curve is smoothed and does not assume exact values of -1 or $+1$. While a randomly initialized state representation produces poor predictions of future reward outcomes (Figures 6(d)), the learned representations produce relatively accurate predictions and follow the expected reward curve (Figures 6(e) and 6(f)). Because the optimization process was forced to compress 100 grid cells into a 80-dimensional latent state space, the latent state space cannot preserve the exact grid cell position and thus approximation errors occur.

Figure 6(g) averages the expected reward-prediction errors across 100 randomly selected action sequences. While a randomly chosen initialization produces high prediction errors,



(a) Puddle-world task map (b) LAM generalization map (c) LSFM generalization map

(d) Example Rollout, Random (e) Example Rollout, LAM (f) Example Rollout, LSFM

(g) Reward-Prediction Error (h) LAM Value-Prediction Error (i) LSFM Value-Prediction Error

Figure 6: Puddle-World Experiment. 6(a): Map of the puddle-world task in which the agent can move up, down, left, or right to transition to adjacent grid cells. The agent always starts at the blue start cell and once the green reward cell is reached a reward of +1 is given and the interaction sequence is terminated. For each transition that enters the orange puddle, a reward of −1 is received. 6(b), 6(c): Partitioning obtained by merging latent states into clusters by Euclidean distance. 6(d), 6(e), 6(f): Expected reward and predictions for a randomly chosen 200-step action sequence using a randomly chosen representation, a representation learned with a LAM, and a representation learned with an LSFM. 6(g): Average expected reward-prediction errors with standard error for each representation. 6(h), 6(i): Optimizing the loss objective results in a sequence of state representations suitable for finding linear approximations of the value functions for a range of different ε-greedy policies. Appendix C.2 presents more details.

the learned state representations produce relatively low prediction errors of future reward outcomes. If expected reward-prediction errors are random after 200 time steps, then the γ-discounted return can be off by at most $0.9^{200} \cdot 1/(1-0.9) \approx 1.4 \cdot 10^{-9}$ after 200 time steps for $\gamma = 0.9$ and a reward range of $[-1, 1]$. Hence, planning over a horizon of more than 200 time steps will impact a policy's value estimate insignificantly. Reward-prediction errors decrease for a randomly chosen state representation (blue curve in Figure 6(g)) because the stochasticity of the task's transitions smooths future expected reward outcomes as the number of steps increases.

While the plots in Figure 6 suggest that both LSFMs and LAMs can be used to learn approximate reward-predictive state representations, the LSFM produces lower prediction errors for expected reward outcomes than the LAM and the LAM produces lower value-prediction errors. Because both models optimize different non-linear and non-convex loss functions, the optimization process leads to different local optima, leading to different performance on the puddle-world task. While prediction errors are present, Figure 6 suggests that both LSFM and LAM learn an approximate reward-predictive state representation and empirically the differences between each model are not significant.

4.4 Connection to Model-based Reinforcement Learning

The key characteristic of a model-based RL agent is to build an internal model of a task that supports predictions of future reward outcomes for any arbitrary decision sequence. Because reward-predictive state representations construct a latent state space suitable for predicting reward sequences for any arbitrary decision sequence, learning reward-predictive state representations can be understood as form of model-based RL. LSFMs tie SFs to reward-predictive state representations, which support predictions of future reward outcomes for any arbitrary decision sequence. The presented analysis describes how learning SFs is akin to learning a transition and reward model in model-based RL.

5. Connections to Value-Predictive Representations

This section first outlines how SFs are related to value-predictive state representations and how learning SFs is akin to model-free learning. Subsequently, we illustrate how reward-predictive state representations can be re-used across tasks with different transitions and rewards to find an optimal policy while re-using value-predictive state representations may prohibit an agent from learning an optimal policy. Barreto et al. (2017a) present SFs as a factorization of the Q-value function for an arbitrary fixed policy π and demonstrate that re-using SFs across tasks with different reward functions improves the convergence rate of online learning algorithms. This factorization assumes a state and action representation function $\xi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^m$ that serves as a basis function for one-step reward predictions and

$$\forall s \in \mathcal{S},\ \forall a \in \mathcal{A},\ \boldsymbol{\xi}_{s,a}^{\top} \boldsymbol{w} = \mathbb{E}_p \left[r(s,a,s') \middle| s, a \right]. \tag{33}$$

Using a state and action representation function, the Q-value function can be factored:

$$Q^\pi(s,a) = \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r(s_t,a_t,s_{t+1})\middle| s_1=s, a_1=a\right] \tag{34}$$

$$= \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{\xi}_{s_t,a_t}^\top \boldsymbol{w}\middle| s_1=s, a_1=a\right] \qquad \text{(by (33))} \tag{35}$$

$$= \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{\xi}_{s_t,a_t}^\top \middle| s_1=s, a_1=a\right]\boldsymbol{w} \tag{36}$$

$$= \left(\boldsymbol{\psi}_{\mathrm{SA}}^\pi(s_1,a_1)\right)^\top \boldsymbol{w}. \qquad \text{(where } s_1=s,\ a_1=a\text{)} \tag{37}$$

Equation (37) assumes the following definition for a SF ψ_{SA}^π:

$$\boldsymbol{\psi}_{\mathrm{SA}}^\pi(s,a) \overset{\text{def.}}{=} \mathbb{E}_{p,\pi}\left[\sum_{t=1}^{\infty}\gamma^{t-1}\boldsymbol{\xi}_{s_t,a_t}\middle| s_1=s, a_1=a\right]. \tag{38}$$

The state and action SF $\boldsymbol{\psi}_{\mathrm{SA}}^\pi$ is a basis function that allows accurate predictions of the Q-value function Q^π. Consequently, the representation $\boldsymbol{\psi}_{\mathrm{SA}}^\pi$ is a value-predictive state representation because it is designed to construct a latent feature space that supports accurate predictions of the Q-value function Q^π.

Figure 7 illustrates that value-predictive state representations generalize across different states differently than reward-predictive state representations. The counter example presented in Figure 7 demonstrates that it is not always possible to construct an optimal policy using a value-predictive state representation. If a sub-optimal policy is used, value-predictive state representations may alias states that have different optimal actions prohibiting an intelligent agent from finding an optimal policy. In this case the value-predictive state representation has to be adjusted for the agent to be able to find an optimal policy, a phenomenon that has been previously described by Russek et al. (2017). In contrast, reward-predictive state representations generalize across the entire policy space (Theorem 4) and allow an agent to find either an optimal policy or close to optimal policy in the presence of approximation errors.

In the following section, we show that learning SFs is akin to learning a value function in model-free RL and the presented argument ties SFs to model-free RL. Subsequently, we demonstrate that reward-predictive state representations can be re-used across tasks with different transitions and rewards to find an optimal policy on two transfer examples in Sections 5.2 and 5.3. While value-predictive state representations may prohibit an agent from learning an optimal policy (Figure 7), we illustrate in which cases reward-predictive state representations overcome this limitation.

5.1 Connection to Linear Temporal Difference Learning

Algorithms that learn SFs can be derived similarly to linear TD-learning (Sutton and Barto, 1998, Chapter 8.4). In linear TD-learning algorithms such as linear Q-learning or SARSA, all Q-values are represented with

$$Q^\pi(s,a;\boldsymbol{\theta}) = \boldsymbol{\xi}_{s,a}^\top \boldsymbol{\theta}, \tag{39}$$

Value predictive:

$Q^{\overline{\pi}}(A,a) = \frac{\gamma}{1-\gamma}0.5$

$Q^{\overline{\pi}}(A,b) = \frac{\gamma}{1-\gamma}0.5$

$Q^{\overline{\pi}}(B,a) = \frac{\gamma}{1-\gamma}0.5$

$Q^{\overline{\pi}}(B,b) = \frac{\gamma}{1-\gamma}0.5$

Reward predictive:

$\boldsymbol{\psi}^{\overline{\pi}}(A,a) = \boldsymbol{\phi}_A + \frac{1}{1-\gamma}\boldsymbol{\phi}_D$

$\boldsymbol{\psi}^{\overline{\pi}}(A,b) = \boldsymbol{\phi}_A + \frac{1}{1-\gamma}\boldsymbol{\phi}_C$

$\boldsymbol{\psi}^{\overline{\pi}}(B,a) = \boldsymbol{\phi}_B + \frac{1}{1-\gamma}\boldsymbol{\phi}_C$

$\boldsymbol{\psi}^{\overline{\pi}}(B,b) = \boldsymbol{\phi}_B + \frac{1}{1-\gamma}\boldsymbol{\phi}_D$

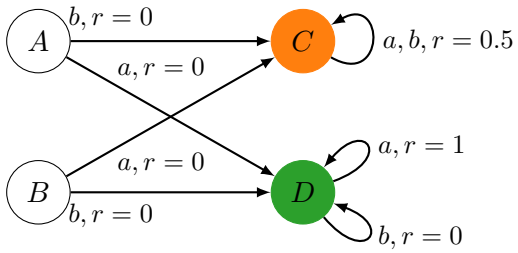


Figure 7: Value-predictive state representations may prohibit an agent from learning an optimal policy. In this MDP, the agent can choose between action a and action b. All transitions are deterministic and each edge is labelled with the reward given to the agent. If a uniform-random action-selection policy is used to construct a value-predictive state representation, then both states A and B will have equal Q-values. A reward-predictive state representation would always distinguish between A and B, because at state A the action sequence $b, a, a...$ leads to a reward sequence of $0, 0.5, 0.5, ...$ while at state B the action sequence $b, a, a, ...$ leads to a reward sequence of $0, 1, 1, ...$ An LSFM detects that states A and B should not be merged into the same latent state, because the states have different SFs. The optimal policy is to select action a at state A, and action b at state B and then collect a reward of one at state D by repeating action a. If an agent uses a reward-predictive state representation, then the optimal policy could be recovered. If an agent uses a value-predictive state representation, the agent would be constrained to not distinguish between states A and B and cannot recover an optimal policy.

where $\boldsymbol{\theta}$ is a real-valued weight vector that does not depend on a state s or action a. Linear TD-learning learns the parameter vector $\boldsymbol{\theta}$ by minimizing the mean squared value error

$$\mathrm{VE}(\boldsymbol{\theta}) = \sum_{s,a,r,s'} \mu(s,a,r,s') \left(Q^{\pi}_{\boldsymbol{\theta}}(s,a;\boldsymbol{\theta}) - y_{s,a,r,s'}\right)^2. \tag{40}$$

Equation (40) averages prediction errors with respect to some distribution μ with which transitions (s,a,r,s') are sampled. The prediction target

$$y_{s,a,r,s'} = r + \gamma \sum_{a'} b(s',a') Q^{\pi}(s',a';\boldsymbol{\theta}) \tag{41}$$

varies by which function b is used. For example, to find the optimal policy linear Q-learning uses an indicator function $b(s,a) = \mathbf{1}[a = \arg\max_a Q^{\pi}(s,a;\boldsymbol{\theta})]$ so that $y_{s,a,r,s'} = r + \gamma \max_{a'} Q^{\pi}(s',a';\boldsymbol{\theta})$. For Expected SARSA (Sutton and Barto, 2018, Chapter 6.6), which evaluates a fixed policy π, the target can be constructed using $b(s,a) = \pi(s,a)$, where $\pi(s,a)$ specifies the probability with which a is selected at state s. When computing a gradient of $\mathrm{VE}(\boldsymbol{\theta})$ the *prediction target* $y_{s,a,r,s'}$ is considered a constant. For an observed transition (s,a,r,s'), the parameter vector is updated using the rule

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \left(Q^{\pi}(s,a;\boldsymbol{\theta}_t) - y_{s,a,r,s'}\right) \boldsymbol{\xi}_{s,a}, \tag{42}$$

where α is a learning rate and the subscript t tracks the update iteration. A SF-learning algorithm can be derived by defining the mean squared SF error (Lehnert et al., 2017)

$$\mathrm{SFE}(\boldsymbol{\psi}^{\pi}_{\mathrm{SA}}) = \sum_{s,a,r,s'} \mu(s,a,r,s') ||\boldsymbol{\psi}^{\pi}_{\mathrm{SA}}(s,a) - \vec{\boldsymbol{y}}_{s,a,r,s'}||^2. \tag{43}$$

Because the SF $\boldsymbol{\psi}^{\pi}_{\mathrm{SA}}(s,a)$ is a vector of dimension m, the target

$$\vec{\boldsymbol{y}}_{s,a,r,s'} = \boldsymbol{\xi}_{s,a} + \gamma \sum_{a'} b(s',a') \boldsymbol{\psi}^{\pi}_{\mathrm{SA}}(s',a') \tag{44}$$

is also a vector but can be constructed similarly to the usual value-prediction target $y_{s,a,r,s'}$. Assuming that SFs are approximated linearly using the basis function ξ,

$$\boldsymbol{\psi}^{\pi}_{\mathrm{SA}}(s,a;\boldsymbol{G}) = \boldsymbol{G}\boldsymbol{\xi}_{s,a}, \tag{45}$$

where $\boldsymbol{F}$ is a square matrix. Computing the gradient of $\mathrm{SFE}(\boldsymbol{\psi}^{\pi})$ with respect to $\boldsymbol{F}$ results in an update rule similar to linear TD-learning:

$$\boldsymbol{G}_{t+1} = \boldsymbol{G}_t + \alpha \left(\boldsymbol{\psi}^{\pi}_{\mathrm{SA}}(s,a;\boldsymbol{G}_t) - \vec{\boldsymbol{y}}_{s,a,r,s'}\right) \boldsymbol{\xi}^{\top}_{s,a}. \tag{46}$$

Assuming the reward condition in Equation (33) holds, both linear TD-learning and SF-learning produce the same value-function sequence.

Proposition 1 (Linear TD-learning and SF-learning Equivalence)**.** *Consider an MDP and a basis function ξ such that $r(s,a) = \boldsymbol{\xi}^{\top}_{s,a}\boldsymbol{w}$ for all states s and actions a. Suppose both iterates in Equation* (42) *and in Equation* (46) *use the same function b to construct prediction targets and are applied for the same trajectory $(s_1, a_1, r_1, s_2, a_2, ...)$. If $\boldsymbol{\theta}_0 = \boldsymbol{G}_0\boldsymbol{w}$, then*

$$\forall t > 0,\ \boldsymbol{\theta}_t = \boldsymbol{G}_t\boldsymbol{w}. \tag{47}$$

Proposition 1 proves that both linear TD-learning and linear SF-learning generate identical value-function estimates on the same trajectory. Appendix B proves Proposition 1. Because linear TD-learning need not converge to an optimal solution, the SF iterate in Equation (46) also need not converge to an optimal solution. The tabular case, in which convergence can be guaranteed, is a sub-case of the presented analysis: For finite state and action spaces, a basis function ξ can be constructed that outputs a one-hot bit vector of dimension n, where n is the number of all state and action pairs. In this case, each weight in the parameter vector $\boldsymbol{\theta}$ corresponds to the Q-value for a particular state and action pair. Similarly, each row in the matrix $\boldsymbol{F}$ corresponds to the SF vector for a particular state and action pair. In this light, learning SFs is akin to learning a value function in model-free RL.

5.2 Generalization Across Transition and Reward Functions

One key distinction between value- and reward-predictive state representations is their ability to generalize across different transition and reward functions. While prior work on SFs (Barreto et al., 2016) and adversarial IRL (Fu et al., 2018) separately model the reward function from the transition function and observed policy, reward-predictive state representations only model equivalence relations between states separately from the transition and reward model. Consequently, reward-predictive state representations extract equivalences between different state's transitions and one-step rewards, reward-predictive state representations can be reused across tasks that vary in their transition and reward functions (Lehnert et al., 2019). Figure 8 presents a transfer experiment highligting that reusing a previously learned reward-predictive state representation allows an intelligent agent to learn an optimal policy using less data.

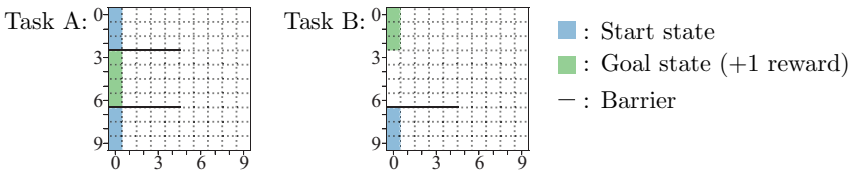
This experiment uses two grid-world tasks (Figure 8(a)): For Task A, a transition data set $\mathcal{D}_A$ is collected. A reward-predictive state representation is learned using an LSFM and a value-predictive state representation is learned using a form of Fitted Q-iteration (Riedmiller, 2005). These state representations are then reused without modification to learn an optimal policy for Task B given a data set $\mathcal{D}_B$ collected from Task B. Both data sets are generated by performing a random walk from a uniformly sampled start state to one of the rewarding goal states. In both tasks, the agent can transition between adjacent grid cells by moving up, down, left, or right, but cannot transition across a barrier. Transitions are probabilistic, because, with a 5% chance, the agent does not move after selecting any action.

Figure 8(b) presents two heuristics for clustering all 100 states into 50 latent states. The first heuristic constructs a reward-predictive state representation by joining states into the same latent state partition if they are directly connected to another. Because both tasks are navigation tasks, partitioning the state space in this way leads to approximately preserving the agent's location in the grid. The second heuristic constructs a value-predictive state representation by joining states that have approximately the same optimal Q-values. Because Q-values are discounted sums of rewards, Q-values decay as one moves further away from a goal cell. This situation leads to different corners being merged into the same state partition (for example grid cell $(0, 0)$ is merged with $(0, 9)$) and an overall more fragmented partitioning that does not perserve the agent's location. Because both state representations are computed for Task A, both state representations can be used to compute an optimal pol-

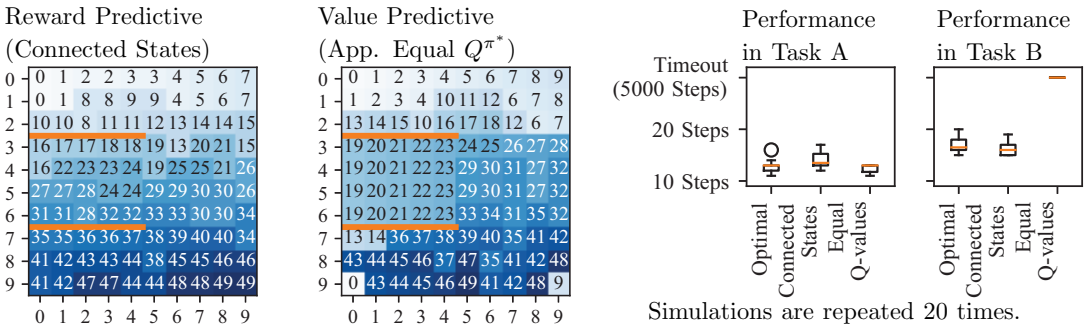
icy for Task A. For Task B, an optimal policy cannot be obtained using the value-predictive state representation. For example, both grid cells at $(0, 0)$ and $(0, 9)$ have different optimal actions in Task B but are mapped to the same latent state. Consequently, an optimal action cannot be computed using the previously learned value-predictive state representation. Because each grid cell has a different optimal action, an optimal abstract policy mapping each latent state to an optimal action cannot be found and the navigation Task B cannot be completed within 5000 time steps if the value-predictive state representation is used (Figure 8(b), right panel). In contrast, the reward-predictive state representation can be used, because it approximately models the grid-world topology. For Task B, each latent state has to be associated with different one-step rewards and latent transitions, but it is still possible to obtain an optimal policy using this state representation and complete the navigation task quickly.

Figure 8(c) repeats a similar experiment, but learns a state representation using either an LSFM to find a reward-predictive state representation or a modification of Fitted Q-iteration to find a value-predictive state representation. The two left panels in Figure 8(c) plot a partitioning of the state space that was obtained by clustering all latent state feature vectors using agglomerative clustering. In this experiment, the latent feature space was set to have 50 dimensions. One can observe that the state representation learned using an LSFM qualitatively corresponds to clustering connected grid cells. The learned value-predictive state representation qualitatively resembles a clustering of states by their optimal Q-values, because Fitted Q-iteration optimizes this state representation to predict the optimal value function as accurately as possible. Both state representations are tested on Task B using the following procedure: First, a data set $\mathcal{D}_B$ was collected of a fixed size. Then, Fitted Q-iteration was used to compute the optimal policy for Task B using the previously learned state representation as a basis function such that $Q^{\pi^*}(s, a) \approx \boldsymbol{\phi}_s^\top \boldsymbol{q}_a$ where $\boldsymbol{q}_a$ is a weight vector and $\phi(s) = \boldsymbol{\phi}_s$. The state representation ϕ trained on Task A is not modified to obtain an optimal policy in Task B. If the training data set $\mathcal{D}_B$ obtained from Task B is too small, then the data set may not provide enough information to find an optimal policy. In this case, Fitted Q-iteration converged to a sub-optimal policy. Because the data sets $\mathcal{D}_B$ are generated at random, one may find that sampling a data set of 2000 transitions may lead to an optimal solution often, but not all the time.

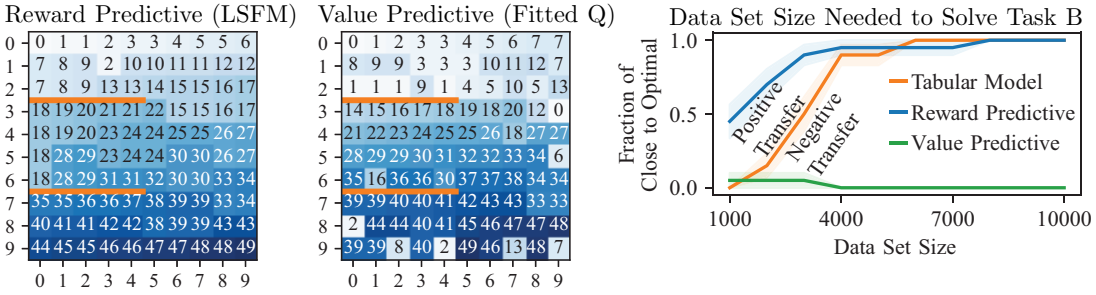
The right panel in Figure 8(c) plots the dependency of being able to find an optimal policy as a function of the transition data set. For each data set size, twenty different data sets were sampled and the y-axis plots the fraction (with standard error of measure) of how often using this data set leads to a close-to-optimal policy. A close-to-optimal policy solves the navigation task in at most 22 time steps. The orange curve is computed using a tabular model-based agent, which constructs a transition and reward table using the sampled data set and solves for an optimal policy using value iteration. Reusing a reward-predictive state representation in Task B often leads to finding an optimal policy for small data set sizes (the blue curve in Figure 8(c), right panel). Because the training data set $\mathcal{D}_B$ is only used to inform different latent transitions and rewards, this agent can generalize across different states and reuse what it has learned without having to observe every possible transition. This behavior leads to better performance than the tabular model-based baseline algorithm, which does not generalize across different states and constructs a transition table for Task B and computes an optimal policy using value iteration (Sutton and Barto, 2018, Chapter



(a) Maps of Transfer Grid Worlds



(b) State representations obtained using a clustering heuristic or optimal Q-values of Task A



(c) State partitions obtained through learning on a transition data set $\mathcal{D}$.

Figure 8: Reward-predictive representations generalize across variations in transitions and rewards. 8(b): The left panels plot state partitions obtained by clustering connected states or states with equal optimal Q-values in Task A (8(a)). The right panels plot the number of times steps a policy, which uses each representation, needs to complete Task B (8(a)). 8(c): The left panels plot partitions obtained by clustering latent states of a reward-predictive and value-predictive representation. The right panel plots how often one out of 20 transition data sets can be used to find an optimal policy as a function of the data set size. By reusing the learned reward-predictive representation, an agent can generalize across states and compute an optimal policy using less data than a tabular model-based RL agent. Reusing a value-predictive representation leads to poor performance, because this representation is only predictive of Task A's optimal policy.

4.4). Reusing the learned value-predictive state representation leads to finding a sub-optimal policy in almost all cases (green curve in Figure 8(c), right panel). The value-predictive state representation is optimized to predict the Q-value function of the optimal policy in Task A. Because Task B has a different optimal policy, reusing this representation does not lead to an optimal policy in Task B, because the previously learned representation is explicitly tied to Task A's optimal policy. Note that any trial that did not find a close-to-optimal policy that completes the task in 22 time steps did also not finish the task and hit the timeout threshold of 5000 time steps. Appendix C.3 presents additional details on the experiments conducted in Figure 8.

5.3 Reward-Predictive Representations Encode Task Relevant State Information

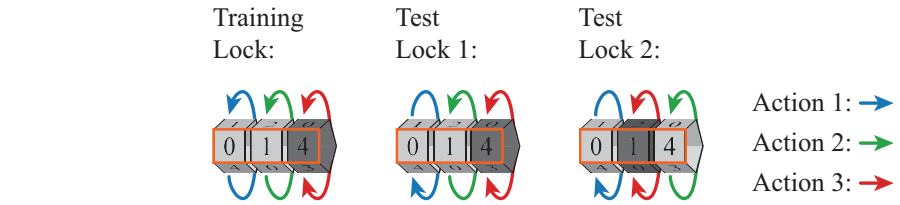
In this section, we present the last simulation result illustrating which aspect of an MDP reward-predictive state representations encode. Figure 9(a) illustrates the combination lock task, where an agent rotates three different numerical dials to obtain a rewarding number combination. In this task, the state is defined as a number triple and each dial has five different positions labelled with the digits zero through four. For example, if Action 1 is chosen at state $(0, 1, 4)$, then the left dial rotates by one step and the state changes to $(1, 1, 4)$. A dial that is set to four will rotate to zero. For example, selecting Action 2 at state $(0, 4, 4)$ will result in a transition to state $(0, 0, 4)$.

In the Training Lock task (Figure 9(a), left schematic), the right dial is "broken" and spins randomly after each time step. Consequently, Action 3 (red arrow) only causes a random change in the right dial and the agent can only use Action 1 (blue arrow) or Action 2 (green arrow) to manipulate the state of the lock. When the agent enters a combination where the left and middle dials are set to four, a reward of $+1$ is given, otherwise the agent receives no reward.[7] While the combination lock task has $5 \cdot 5 \cdot 5 = 125$ different states, the state space of the Training Lock can be compressed to $5 \cdot 5 = 25$ latent states by ignoring the position of the randomly changing right dial, because the right dial is neither relevant for maximizing reward nor predicting reward sequences.

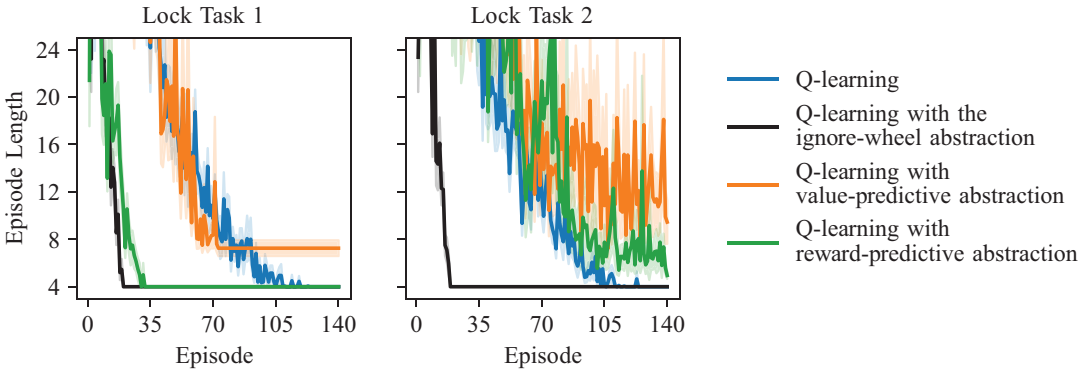
The Test Lock 1 and Test Lock 2 tasks (Figure 9(a), center and right schematics) differ from the training task in that the rewarding combination is changed and the rotation direction of the left dial is reversed (Action 1, blue arrow), resembling a change in both transition and reward functions. While in Test Lock 1, the right dial still spins at random, in Test Lock 2 the middle dial rotates at random instead and the right dial becomes relevant for maximizing reward. Both test tasks can also be compressed into 25 latent states by ignoring the position of the randomly rotating dial, but in Test Lock 2 this compression would be constructed differently than in Test Task 1 or in the Training Lock.

To determine if a reward- or value-predictive state representation can be re-used to accelerate learning in a previously unseen task, we compute a two-state representations of each type for the Training Lock MDP. To assess if each state abstraction can be re-used, they are both tested by compressing the state space of a Q-learning agent (Watkins and Dayan, 1992) to learn an optimal policy in Test Lock 1 and Test Lock 2. Any resulting performance changes are then indicative of the state abstraction's ability to generalize from

7. Specifically, in the Training Lock task, the rewarding states are $(4, 4, 0), (4, 4, 1), \ldots (4, 4, 4)$.



(a) Combination Lock Tasks



(b) Performance of Each Model at Transfer

Figure 9: Combination Lock Transfer Experiment. 9(a): In the combination lock tasks, the agent decides between three different actions to rotate each dial by one digit. Each dial has five sides labelled with the digits zero through four. The dark gray dial is "broken" and spins at random at every time step. In the training task, any combination setting the left and middle dial to four are rewarding. In Test Lock 1, setting the left dial to two and the middle dial to three is rewarding and simulations were started by setting the left dial to two and the middle dial to four. In Test Lock 2, setting the left dial to two and the right dial to three is rewarding and simulations were started by setting the left dial to two and the right dial to four. 9(b): Each panel plots the episode length of the Q-learning algorithm on Lock Task 1 and Lock Task 2 averaged over 20 repeats. Note that Q-learning with the ignore-wheel abstraction uses a different abstraction in Test Lock 1 and Test Lock 2. In Test Lock 1, the ignore-wheel abstraction ignores the right dial. In Test Lock 2, the ignore-wheel abstraction ignores the middle dial. Please refer to Appendix C.3 for a detailed description of the experiment implementation.

the Training Lock task to Test Lock 1 and Test Lock 2. If a state representation can generalize information from the training to the test task, then Q-learning with a state abstraction should converge to an optimal policy faster than a Q-learning agent that does not use any state abstraction. To combine the tabular Q-learning algorithm with a state abstraction, we assume in this section that a state abstraction function maps each state to a set of discrete latent states. Furthermore, before updating the Q-learning agent with a transition (s, a, r, s'), this transition is mapped to a latent space transition $(\phi(s), a, r', \phi(s))$ using the respective state abstraction function ϕ. Because the latent state space is smaller than the task's original state space, one would expect that using a state abstraction function results in faster convergence, because any Q-value update is generalized across all states that are mapped to the same latent state.

The reward-predictive state representation is computed using the training task's transition and reward table with the same procedure used for the column-world task presented in Figure 5. (Please also refer to Appendix C.3 for implementation details.) To obtain a state-representation function mapping states to discrete latent states, the real-valued state representation function $\phi_{\text{real-valued}} : \mathcal{S} \to \mathbb{R}^n$ is then further refined by clustering the set of feature vectors $\{\phi_{\text{real-valued}}(s) | s \in \mathcal{S}\}$ into discrete clusters using agglomerative clustering. Each cluster then becomes a separate latent state in the resulting latent state space.

The value-predictive state representation is computed by associating two states with the same latent state if the optimal policy has equal Q-values at both states and for each action. This type of state abstraction has been previously introduced by Li et al. (2006) as a Q^*-irrelevance state abstraction and is predictive of Q-values because each latent state is associated with a different set of Q-values.

Figure 9(b) plots the episode length of four different Q-learning agent configurations. The first configuration (blue curves in Figure 9(b)) forms a baseline and simulates the Q-learning algorithm on both test tasks, without using any state abstraction. The second configuration, called "Q-learning with ignore dial abstraction" (black curves in Figure 9(b)) simulates the Q-learning algorithm with a manually coded state abstraction that ignores the right dial in Test Lock 1 and the middle dial in Test Lock 2. Because this state abstraction compresses the 125 task states into 25 latent states by removing the digit from the state triplet not relevant for maximizing reward, this variation converges significantly faster than the baseline algorithm. The third variation, called "Q-learning with value-predictive abstraction" (orange curves in Figure 9(b)) simulates Q-learning with the value-predictive state abstraction. In both Test Lock 1 and Test Lock 2, this variation converges more slowly than using Q-learning without any state abstraction. As discussed in Section 5.2, a value-predictive state abstraction is constructed using the Q-values of the policy that is optimal in the Training Lock. Because each combination lock MDP has a different optimal policy, a value-predictive state abstraction cannot be transferred across any of the two tasks. Consequently, the "Q-learning with a value-predictive abstraction" agent does not converge more quickly than the baseline agent. In these simulations, the Q-learning algorithm is not capable of finding an optimal policy, as outlined previously in Figure 7. The green curves in Figure 9(b) plot the average episode length when Q-learning is combined with a reward-predictive state abstraction. In Lock Task 1, this agent converges almost as fast as the agent using the manually coded state abstraction (black curve, left panel). Only the position of the left and middle dials are relevant for predicting reward sequences $r_1, ..., r_t$ that are gen-

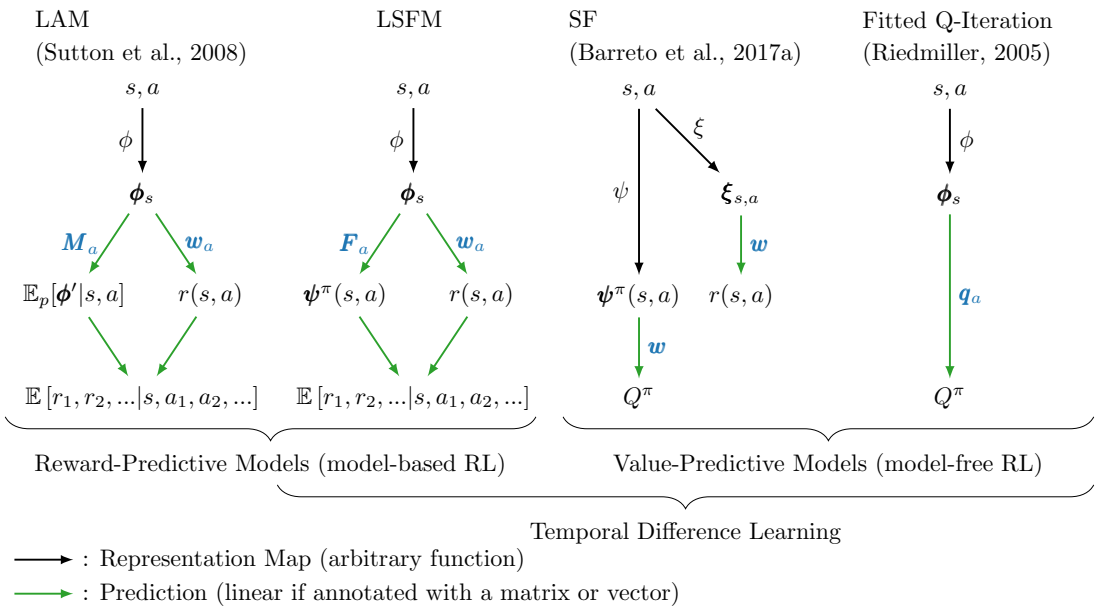


Figure 10: Comparison of Presented State-Representation Models.

erated by following an arbitrary action sequence $a_1, ..., a_t$ from an arbitrary start state s in the Training Lock MDP and in the Test Lock 1 MDP. Consequently, a reward-predictive state abstraction can compress the 125 task states into 25 different latent states by ignoring the right dial, resulting in a significant performance improvement in Test Lock 1. Note that LSFMs only approximate reward-predictive state representations resulting in slightly slower convergence in comparison to the black curve in the left panel of Figure 9(b).

This behaviour changes in Test Lock 2, because in Test Lock 2 a different dial moves at random, changing how the state space should be compressed. Because the right dial is relevant for predicting expected reward sequences in Test Lock 2, the reward-predictive state representation learned in the Training Lock task is no longer reward-predictive in Test Lock 2 and cannot be re-used without modifications. Consequently, the "Q-learning with reward-predictive abstraction" agent exhibits worse performance (Figure 9(b), right panel, green curve) than using Q-learning without any state abstraction (Figure 9(b), right panel, blue curve).

The results presented in Figure 9 demonstrate that reward-predictive state representations encode which state information is relevant for predicting reward sequences and maximizing reward in a task. Because reward-predictive state representations only model this aspect of an MDP, this type of state representation generalizes across tasks that preserve these state equivalences but differ in their transitions and rewards otherwise. In contrast, value-predictive state representations "overfit" to a specific MDP and the MDP's optimal policy, resulting in negative transfer and possibly prohibiting an agent from finding an optimal policy at transfer.

6. Discussion

This article presents a study of how successor features combine aspects of model-free and model-based RL. Connections are drawn by analyzing which properties different latent state spaces are predictive of. The schematic in Figure 10 illustrates the differences between the presented models. By introducing LSFMs, SFs are tied to learning state representations that are predictive of future expected reward outcomes. This model ties successor features to model-based reinforcement learning, because an agent that has learned an LSFM can predict expected future reward outcomes for any arbitrary action sequence. While this connection to model-based RL has been previously hypothesized (Russek et al., 2017; Momennejad et al., 2017), LSFMs formalize this connection. Because SFs obey a fixed-point equation similar to the Bellman fixed-point equation, SFs can also be linked to temporal-difference learning. Similar to LAMs, LSFMs are a "strict" model-based architecture and are distinct from model-based and model-free hybrid architectures that iteratively search for an optimal policy and adjust their internal representation (Oh et al., 2017; Weber et al., 2017; François-Lavet et al., 2019; Gelada et al., 2019). LSFMs only evaluate SFs for a fixed target policy that selects actions uniformly at random. The learned model can then be used to predict the value function of any arbitrary policy, including the optimal policy. In contrast to model-based and model-free hybrid architectures, the learned state representation does not have to be adopted to predict an optimal policy and generalizes across all latent policies. How to learn neural networks mapping inputs to latent feature spaces that are predictive of future reward outcomes is beyond the scope of this article and is left for future work.

Similar to reward-predictive state representations, the Predictron architecture (Silver et al., 2017) and the MuZero algorithm (Schrittwieser et al., 2019) use or construct a state representation to predict reward sequences. In contrast to reward-predictive state representations, these other architectures predict reward sequences for k time steps and then use the value function for one (or multiple) policies to predict the return obtained after k time steps. This distinction is key in learning reward-predictive state representations with LSFMs, which do not include a value prediction module, because if a state representation is designed to predict the value function of a policy, then the resulting state representation would be (to some extent) value predictive. As outlined in Section 5, this change would compromise the resulting state abstraction's ability to generalize across different transition and reward functions.

In contrast to the SF framework introduced by Barreto et al. (2017a), the connection between LSFMs and model-based RL is possible because the same state representation ϕ is used to predict its own SF (Figure 10 center column). While the deep learning models presented by Kulkarni et al. (2016) and Zhang et al. (2017) also use one state representation to predict SFs and one-step rewards, these models are also constrained to predict image frames. LSFMs do not use the state representation to reconstruct actual states. Instead, the state space is explicitly compressed, allowing the agent to generalize across distinct states.

Table 1 summarizes different properties of the presented state representations. Bisimulation relations (Givan et al., 2003) preserve most structure of the original MDP and latent transition probabilities match with transition probabilities in the original MDP (Section 4.1). Reward-predictive state representations do not preserve the transition probabil-

Model	Predicts Trained Policy	Generalizes to Variations in Rewards	Generalizes to Variations in Transitions	Generalizes Across All Policies	Predicts Transition Probabilities
Bisimulation (Givan et al., 2003)	yes	yes	yes	yes	yes
Reward-Predictive (LSFM or LAM)	yes	yes	yes	yes	no
Successor Features (Barreto et al., 2017a)	yes	yes	no	no	no
Value-Predictive (Fitted Q-iteration)	yes	no	no	no	no

Table 1: Summary of Generalization Properties of Presented State Representations

ities of the original task (Figure 4) but construct a latent state space that is predictive of expected future reward outcomes. These two representations generalize across all abstract policies, because they can predict reward outcomes for arbitrary action sequences. Successor features are equivalent to value-predictive state representations, which construct a latent state space that is predictive of a policy's value function (Section 5). Because the value function can be factored into SFs and a reward model (Equation (37)), SFs are robust to variations in reward functions. Reward-predictive state representations remove previous limitations of SFs and generalize across variations in transition functions. This property stands in contrast to previous work on SFs, which demonstrate robustness against changes in the reward function only (Barreto et al., 2017a, 2018; Kulkarni et al., 2016; Zhang et al., 2017; Stachenfeld et al., 2017; Momennejad et al., 2017; Russek et al., 2017). In all cases, including reward-predictive state representations, the learned models can only generalize to changes that approximately preserve the latent state space structure. For example, in Figure 8 positive transfer is possible because both tasks are grid worlds and only the locations of rewards and barriers is changed. If the same representation is used on a completely different randomly generated finite MDP, then positive transfer may not be possible because both tasks do not have a latent state structure in common.

In comparison to previous work on state abstractions (Even-Dar and Mansour, 2003; Li et al., 2006; Abel et al., 2016, 2018, 2019), this article does not consider state representations that compress the state space as much as possible. Instead, the degree of compression is set through the hyper-parameter that controls the dimension of the constructed latent state space. The presented experiments demonstrate that these state representations compress the state space and implicitly convey information useful for transfer. This formulation of state representations connects ideas from state abstractions to models that analyze linear basis functions (Parr et al., 2008; Sutton, 1996; Konidaris et al., 2011) or learn linear representations of the transition and reward functions (Song et al., 2016). Recently, Ruan et al. (2015) presented algorithms to cluster approximately bisimilar states. Their method relies on bisimulation metrics (Ferns et al., 2004), which use the Wasserstein metric to assess if two state transitions have the same distribution over next state clusters. In contrast to their approach, we phrase learning reward-predictive state representations as an energy-minimization problem and remove the requirement of computing the Wasserstein metric.

The presented experiments learn state representations by generating a transition data set covering all states of an MDP. Complete coverage is obtained on the grid world tasks by

generating a large enough transition data set. Because this article focuses on drawing connections between different models of generalization across states, combining the presented algorithms with efficient exploration algorithms (Jin et al., 2018) or obtaining sample complexity or regret bounds similar to prior work (Jaksch et al., 2010; Azar et al., 2017; Osband et al., 2013) is left to future studies.

7. Conclusion

This article presents an analysis of which latent representations an intelligent agent can construct to support different predictions, leading to new connections between model-based and model-free RL. By introducing LSFMs, the presented analysis links learning successor features to model-based RL and demonstrates that the learned reward-predictive state representations are suitable for transfer across variations in transitions and rewards. The presented results outline how different models of generalization are related to another and proposes a model for phrasing model-based learning as a representation-learning problem. These results motivate the design and further investigation of new approximate model-based RL algorithms that learn state representations instead of one-step reward and transition models.

Acknowledgments

We would like to thank Prof. Michael J. Frank for many insightful discussions that benefited the development of the presented work. This project was supported in part by the ONR MURI PERISCOPE project and in part by the NIH T32 Training Grant on Interactionist Cognitive Neuroscience.

Appendix A. Proofs of Theoretical Results

This section lists formal proofs for all presented theorems and propositions.

A.1 Bisimulation Theorems

For an equivalence relation $\sim$ defined on a set $\mathcal{S}$, the set of all partition is denoted with $\mathcal{S}/\sim$. Each partition $[s] \in \mathcal{S}/\sim$ is a subset of $\mathcal{S}$ and $s \in [s]$.

Definition 4 (Bisimilarity (Ferns et al., 2011))**.** *For an MDP $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ where $\langle \mathcal{S}, \Sigma, p \rangle$ is a measurable space with σ-algebra Σ and p is a Markov kernel labelled for each action $a \in \mathcal{A}$. Consider an equivalence relation $\sim_b$ on the state space $\mathcal{S}$ such that each state partition $[s']$ also lies in the σ-algebra and $\forall [s'] \in \mathcal{S}/\sim_b,\ [s'] \in \Sigma$. The equivalence relation $\sim_b$ is a bisimulation if*

$$s \sim_b \tilde{s} \iff \forall a \in \mathcal{A},\ \mathbb{E}_p\left[r(s,a,s')\middle|s,a\right] = \mathbb{E}_p\left[r(\tilde{s},a,s')\middle|\tilde{s},a\right] \tag{48}$$

$$\text{and } \forall [s'] \in \mathcal{S}/\sim_b,\ p([s']|s,a) = p([s']|\tilde{s},a). \tag{49}$$

Using this definition, Theorem 1 can be proven.

Proof of Theorem 1. Consider any two states s and $\tilde{s}$ such that $\boldsymbol{\phi}_s = \boldsymbol{\phi}_{\tilde{s}}$. For both s and $\tilde{s}$ we have

$$\mathbb{E}_p\left[r(s,a,s')\middle|s,a\right] = \boldsymbol{\phi}_s^\top \boldsymbol{w}_a = \boldsymbol{\phi}_{\tilde{s}}^\top \boldsymbol{w}_a = \mathbb{E}_p\left[r(\tilde{s},a,s')\middle|\tilde{s},a\right], \tag{50}$$

and the bisimulation reward condition in Equation (48) holds. To show that also the bisimulation transition condition in Equation (49) holds, observe that

$$\boldsymbol{\phi}_s^\top = \boldsymbol{\phi}_{\tilde{s}}^\top \iff \tag{51}$$

$$\boldsymbol{\phi}_s^\top \boldsymbol{M}_a = \boldsymbol{\phi}_{\tilde{s}}^\top \boldsymbol{M}_a \iff \tag{52}$$

$$\mathbb{E}_p\left[\boldsymbol{\phi}_{s'}|s,a\right] = \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}|\tilde{s},a\right] \iff \tag{53}$$

$$\sum_{i=1}^{n} p(s,a,[s_i])\boldsymbol{e}_i = \sum_{i=1}^{n} p(\tilde{s},a,[s_i])\boldsymbol{e}_i, \tag{54}$$

where $[s_i] \subset \mathcal{S}$ are all states that are mapped to the one-hot feature vector $\boldsymbol{e}_i$. Each side of the identity (54) computes an expectation over one-hot bit vectors and thus the ith entry of $\sum_{i=1}^{n} p(s,a,[s_i])\boldsymbol{e}_i$ contains the probability value $p(s,a,[s_i])$. Hence both s and $\tilde{s}$ have equal probabilities of transitioning into each state partition that is associated with $\boldsymbol{e}_i$. Define an equivalence relation $\sim_\phi$ such that

$$\forall s, \tilde{s} \in \mathcal{S},\ \boldsymbol{\phi}_s = \boldsymbol{\phi}_{\tilde{s}} \iff s \sim_\phi \tilde{s}. \tag{55}$$

Because all feature vectors $\boldsymbol{\phi}_s$ are one-hot bit vectors, there are at most n partitions and the set of all state partitions has size $|\mathcal{S}/\sim_\phi| = n$. Combining these observations, Equation (54) can be rewritten as

$$\forall [s'] \in \mathcal{S}/\sim_\phi,\ p([s']|s,a) = p([s']|\tilde{s},a). \tag{56}$$

By lines (50) and (56), the equivalence relation $\sim_\phi$ is a bisimulation relation and if $\boldsymbol{\phi}_s = \boldsymbol{\phi}_{\tilde{s}}$ then both s and $\tilde{s}$ are bisimilar. □

Lemma 2. *Assume an MDP, state representation $\phi : \mathcal{S} \to \{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$, LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, and arbitrary policy $\pi \in \Pi_\phi$. Let $\boldsymbol{F}^\pi$ be an $n \times n$ real valued matrix with each row $\boldsymbol{F}^\pi(i) = \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top \boldsymbol{F}_a \middle| s\right]$, then*

$$\mathbb{E}_\pi\left[\boldsymbol{\phi}_s^\top \boldsymbol{F}_a \middle| s\right] = \boldsymbol{e}_i^\top \boldsymbol{F}^\pi, \tag{57}$$

where $\boldsymbol{\phi}_s = \boldsymbol{e}_i$ for some i. For a LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, let $\boldsymbol{M}^\pi$ be a $n \times n$ real-valued matrix with each row $\boldsymbol{M}^\pi(i) = \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top \boldsymbol{M}_a \middle| s\right]$, then

$$\mathbb{E}_\pi\left[\boldsymbol{\phi}_s^\top \boldsymbol{M}_a \middle| s\right] = \boldsymbol{e}_i^\top \boldsymbol{M}^\pi. \tag{58}$$

Proof of Lemma 2. The first identities in (57) and (58) hold because $\boldsymbol{\phi}_s = \boldsymbol{e}_i$ for some i. Then,

$$\mathbb{E}_\pi\left[\boldsymbol{\phi}_s^\top \boldsymbol{F}_a \middle| s\right] = \sum_a \pi(s,a)\boldsymbol{\phi}_s \boldsymbol{F}_a = \sum_a \pi(s,a)\boldsymbol{e}_i^\top \boldsymbol{F}_a = \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top \boldsymbol{F}_a \middle| s\right] = \boldsymbol{F}^\pi(i) = \boldsymbol{e}_i^\top \boldsymbol{F}^\pi$$

and

$$\mathbb{E}_\pi\left[\boldsymbol{\phi}_s^\top \boldsymbol{M}_a \middle| s\right] = \sum_a \pi(s,a)\boldsymbol{\phi}_s \boldsymbol{M}_a = \sum_a \pi(s,a)\boldsymbol{e}_i^\top \boldsymbol{M}_a = \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top \boldsymbol{M}_a \middle| s\right] = \boldsymbol{M}^\pi(i) = \boldsymbol{e}_i^\top \boldsymbol{M}^\pi.$$

□

Definition 5 (Weighting Function)**.** *For an MDP $M = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma\rangle$, let $\sim$ be an equivalence relation on the state space $\mathcal{S}$, creating a set of state partitions $\mathcal{S}/\sim$. Assume that each state partition $[s]$ is a measurable space $\langle [s], \Sigma_{[s]}, \omega_{[s]}\rangle$, where $\omega_{[s]}$ is a probability measure indexed by each partition $[s]$ with σ-algebra $\Sigma_{[s]}$. The function ω is called the weighting function.*

Lemma 3. *Assume an MDP, state representation $\phi : \mathcal{S} \to \{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$, LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, and arbitrary abstract policy $\pi \in \Pi_\phi$. Then,*

$$\forall s, \forall a\ \boldsymbol{\phi}_s^\top \boldsymbol{F}_a = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p,\pi}\left[\boldsymbol{\phi}_{s'} \boldsymbol{F}_{a'} | s, a\right] \implies \forall s, \forall a, \exists \boldsymbol{M}_a \text{ such that } \boldsymbol{F}_a = \boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi, \tag{59}$$

where the matrix $\boldsymbol{F}^\pi$ is constructed as described in Lemma 2.

Proof of Lemma 3. Consider an equivalence relation $\sim_\phi$ that is constructed using the state representation ϕ and

$$\forall s, \tilde{s} \in \mathcal{S},\ \boldsymbol{\phi}_s = \boldsymbol{\phi}_{\tilde{s}} \iff s \sim_\phi \tilde{s}. \tag{60}$$

The weighting function ω models a probability distribution of density function of visiting a state s that belongs to a state partition $[s] \in \mathcal{S}/\sim_\phi$. Because all states $s \in [s]$ are mapped to the same feature vector $\boldsymbol{\phi}_s$, we have that

$$\mathbb{E}_{\omega_{[s]}}\left[\boldsymbol{\phi}_s\right] = \boldsymbol{\phi}_s. \tag{61}$$

The stochastic matrix $\boldsymbol{M}_a$ is defined for every action a as

$$\boldsymbol{M}_a(i,j) = \mathbb{E}_{\omega_{[s]}}\left[\Pr\left\{s \xrightarrow{a} \boldsymbol{e}_j\right\}\right] \text{ with } \boldsymbol{\phi}_s = \boldsymbol{e}_i, \tag{62}$$

where $\Pr\left\{s \xrightarrow{a} \boldsymbol{e}_j\right\}$ is the probability of transitioning into the state partition associated with the latent state $\boldsymbol{e}_j$ and

$$\Pr\left\{s \xrightarrow{a} \boldsymbol{e}_j\right\} = p(s, a, [s_i]) \text{ such that } \forall s \in [s_i], \phi(s) = \boldsymbol{e}_j. \tag{63}$$

The identity in Equation (59) can be re-written as follows:

$$\begin{aligned}
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p,\pi}\left[\boldsymbol{\phi}_{s'} \boldsymbol{F}_{a'} | s, a\right] && \iff \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p}\left[\boldsymbol{\phi}_{s'} | s, a\right] \boldsymbol{F}^\pi && \iff \text{(by Lemma 2)} \\
\mathbb{E}_{\omega_{[s]}}\left[\boldsymbol{\phi}_s^\top \boldsymbol{F}_a\right] &= \mathbb{E}_{\omega_{[s]}}\left[\left(\boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p}\left[\boldsymbol{\phi}_{s'} | s, a\right]\right) \boldsymbol{F}^\pi\right] && \iff \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{\omega_{[s]}}\left[\mathbb{E}_{p}\left[\boldsymbol{\phi}_{s'} | s, a\right]\right] \boldsymbol{F}^\pi && \iff \text{(by (61))} \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{\omega_{[s]}}\left[\sum_{j=1}^{n} \Pr\left\{s \xrightarrow{a} \boldsymbol{e}_j\right\} \boldsymbol{e}_j^\top\right] \boldsymbol{F}^\pi && \iff \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \underbrace{\left[\mathbb{E}_{\omega_{[s]}}\left[\Pr\left\{s \xrightarrow{a} \boldsymbol{e}_1\right\}\right], ..., \mathbb{E}_{\omega_{[s]}}\left[\Pr\left\{s \xrightarrow{a} \boldsymbol{e}_n\right\}\right]\right]}_{n\text{-dimensional row vector, because } \boldsymbol{e}_j \text{ is one-hot}} \boldsymbol{F}^\pi && \iff \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \left[\boldsymbol{M}_a(i,1), ..., \boldsymbol{M}_a(i,n)\right] \boldsymbol{F}^\pi && \iff \text{(by (62))} \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{F}^\pi && \iff \text{(by } \boldsymbol{\phi}_s = \boldsymbol{e}_i\text{)} \\
\boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top \left(\boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi\right) && \qquad (64)
\end{aligned}$$

Equation (64) holds for any arbitrary state s and because each state is mapped to a one of the one-hot vectors $\{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$,

$$\begin{aligned}
\forall i \in \{1, ..., n\},\ \boldsymbol{e}_i^\top \boldsymbol{F}_a &= \boldsymbol{e}_i^\top \left(\boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi\right) && \iff \\
\boldsymbol{F}_a &= \boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi.
\end{aligned}$$

□

Lemma 4. *Consider a state representation $\phi : \mathcal{S} \to \{\boldsymbol{e}_1, ..., \boldsymbol{e}_n\}$, an arbitrary abstract policy $\pi \in \Pi_\phi$, an LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ and LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ where each transition matrix $\boldsymbol{M}_a$ is stochastic. Then,*

$$\boldsymbol{F}_a = \boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi \implies \exists \left(\boldsymbol{F}^\pi\right)^{-1} \text{ and } \left(\boldsymbol{F}^\pi\right)^{-1} = \boldsymbol{I} - \gamma \boldsymbol{M}^\pi, \tag{65}$$

where the matrix $\boldsymbol{M}^\pi$ is constructed as described in Lemma 2.

Proof of Lemma (4). By Lemma 2, one can write for any arbitrary i

$$\begin{aligned}
\boldsymbol{e}_i^\top \boldsymbol{F}^\pi &= \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top \boldsymbol{F}_a\right] && \iff \\
\boldsymbol{e}_i^\top \boldsymbol{F}^\pi &= \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top (\boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi)\right] && \iff \text{(by Lemma 3)} \\
\boldsymbol{e}_i^\top \boldsymbol{F}^\pi &= \boldsymbol{e}_i \boldsymbol{I} + \gamma \mathbb{E}_\pi\left[\boldsymbol{e}_i^\top \boldsymbol{M}_a\right] \boldsymbol{F}^\pi && \iff \\
\boldsymbol{e}_i^\top \boldsymbol{F}^\pi &= \boldsymbol{e}_i \boldsymbol{I} + \gamma \boldsymbol{e}_i^\top \boldsymbol{M}^\pi \boldsymbol{F}^\pi && \iff \text{(by Lemma 2)} \\
\boldsymbol{e}_i^\top \boldsymbol{F}^\pi &= \boldsymbol{e}_i \left(\boldsymbol{I} + \gamma \boldsymbol{M}^\pi \boldsymbol{F}^\pi\right) && \qquad (66)
\end{aligned}$$

Because Equation (66) holds for every i,

$$\begin{aligned} \boldsymbol{F}^\pi &= \boldsymbol{I} + \gamma \boldsymbol{M}^\pi \boldsymbol{F}^\pi && \iff \\ \boldsymbol{F}^\pi - \gamma \boldsymbol{M}^\pi \boldsymbol{F}^\pi &= \boldsymbol{I} && \iff \\ (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi)\boldsymbol{F}^\pi &= \boldsymbol{I}. \end{aligned}$$

Because $\boldsymbol{M}^\pi$ is stochastic, it has a spectral radius of at most one and all its eigenvalues $\lambda_j \leq 1$. Thus the matrix $\boldsymbol{I} - \gamma \boldsymbol{M}^\pi$ is invertible because

$$\det(\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) \geq \det(\boldsymbol{I}) + (-\gamma)^n \det(\boldsymbol{M}^\pi) \geq 1 - \gamma^n \det(\boldsymbol{M}^\pi) = 1 - \gamma^n \underbrace{\prod_j \lambda_j}_{\leq 1} > 0. \tag{67}$$

Hence $\boldsymbol{F}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi)^{-1} \iff (\boldsymbol{F}^\pi)^{-1} = \boldsymbol{I} - \gamma \boldsymbol{M}^\pi$. □

Using these lemmas, Theorem 2 can be proven.

Proof of Theorem 2. The proof is by reducing Equation (18) to Equation (16) using the previously established lemmas and then applying Theorem 1. Equation (18) can be re-written as follows:

$$\begin{aligned} \boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p,\pi} \left[\boldsymbol{\phi}_{s'} \boldsymbol{F}_{a'} | s, a \right] && \Leftrightarrow \\ \boldsymbol{\phi}_s^\top \boldsymbol{F}_a &= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] \boldsymbol{F}^\pi && \Leftrightarrow \text{(Lem. 2)} \\ \boldsymbol{\phi}_s^\top \boldsymbol{F}_a (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) &= \boldsymbol{\phi}_s^\top (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) + \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] \boldsymbol{F}^\pi (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) && \Leftrightarrow \text{(Lem. 4)} \\ \boldsymbol{\phi}_s^\top \boldsymbol{F}_a (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) &= \boldsymbol{\phi}_s^\top (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) + \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] \boldsymbol{F}^\pi \left(\boldsymbol{F}^\pi \right)^{-1} && \Leftrightarrow \text{(Lem. 4)} \\ \boldsymbol{\phi}_s^\top \boldsymbol{F}_a (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) &= \boldsymbol{\phi}_s^\top (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) + \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] && \Leftrightarrow \\ \boldsymbol{\phi}_s^\top (\boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi)(\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) &= \boldsymbol{\phi}_s^\top (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) + \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] && \Leftrightarrow \text{(Lem. 3)} \\ \boldsymbol{\phi}_s^\top \gamma \boldsymbol{M}_a \boldsymbol{F}^\pi (\boldsymbol{I} - \gamma \boldsymbol{M}^\pi) &= \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] && \Leftrightarrow \\ \boldsymbol{\phi}_s^\top \gamma \boldsymbol{M}_a &= \gamma \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] && \Leftrightarrow \text{(Lem. 4)} \\ \boldsymbol{\phi}_s^\top \boldsymbol{M}_a &= \mathbb{E}_p \left[\boldsymbol{\phi}_{s'} | s, a \right] && (68) \end{aligned}$$

Using the reward condition stated in Equation (18) and Equation (68) Theorem 1 can be applied to conclude the proof.

To prove the last claim of Theorem 2, assume that

$$\boldsymbol{\phi}_s^\top \boldsymbol{F}_a = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p,\pi} \left[\boldsymbol{\phi}_{s'} \boldsymbol{F}_{a'} | s, a \right] \tag{69}$$

for some policy $\pi \in \Pi_\phi$. Consider an arbitrary distinct policy $\tilde{\pi} \in \Pi_\phi$, then the fixed-point Eq. (69) can be re-stated in terms of then policy $\tilde{\pi}$:

$$\boldsymbol{\phi}_s^\top \boldsymbol{F}_a = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p,\pi}\left[\boldsymbol{\phi}_{s'} \boldsymbol{F}_{a'} | s, a\right] \qquad \Leftrightarrow \tag{70}$$

$$\boldsymbol{\phi}_s^\top \boldsymbol{M}_a = \mathbb{E}_p\left[\boldsymbol{\phi}_{s'} | s, a\right] \qquad \Leftrightarrow \text{(Eq. (68))} \tag{71}$$

$$\gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{F}^{\tilde{\pi}} = \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'} | s, a\right] \boldsymbol{F}^{\tilde{\pi}} \qquad \Leftrightarrow \text{(multiply with } \boldsymbol{F}^{\tilde{\pi}} \text{ and } \gamma) \tag{72}$$

$$\boldsymbol{\phi}_s^\top + \gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{F}^{\tilde{\pi}} = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'} | s, a\right] \boldsymbol{F}^{\tilde{\pi}} \qquad \Leftrightarrow \text{(add } \boldsymbol{\phi}_s) \tag{73}$$

$$\boldsymbol{\phi}_s^\top \left(\boldsymbol{I} + \gamma \boldsymbol{M}_a \boldsymbol{F}^{\tilde{\pi}}\right) = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'} | s, a\right] \boldsymbol{F}^{\tilde{\pi}} \qquad \Leftrightarrow \tag{74}$$

$$\boldsymbol{\phi}_s^\top \boldsymbol{F}_a = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'} | s, a\right] \boldsymbol{F}^{\tilde{\pi}} \qquad \Leftrightarrow \text{(Lem. 3)} \tag{75}$$

$$\boldsymbol{\phi}_s^\top \boldsymbol{F}_a = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_{p,\tilde{\pi}}\left[\boldsymbol{\phi}_{s'} \boldsymbol{F}_a | s, a\right] \qquad \Leftrightarrow \text{(Lem. 2)} \tag{76}$$

This argument shows that if Eq. (69) holds for a policy π, then Eq. (76) also holds for other arbitrary policy $\tilde{\pi}$. □

A.2 Approximate Reward-Predictive State Representations

This section presents formal proofs for Theorem 3 and 4. While the following proofs assume that the matrix $\overline{\boldsymbol{F}}$ is defined as stated in Equation (25), these proofs could be generalized to different definitions of $\overline{\boldsymbol{F}}$, assuming that the matrix $\overline{\boldsymbol{F}}$ is not a function of the state s and only depends on the matrices $\{\boldsymbol{F}_a\}_{a \in \mathcal{A}}$.

Lemma 5. *For an MDP, a state representation ϕ, a LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ and a LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ where $\Delta = 0$,*

$$\varepsilon_p \leq \varepsilon_\psi \frac{1 + \gamma M}{\gamma}. \tag{77}$$

Proof of Lemma 5. The proof is by manipulating the definition of ε_ψ and using the fact that $\Delta = 0$ and $\boldsymbol{F}_a = \boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}}$. Let $\overline{M} = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \boldsymbol{M}_a$, then

$$\overline{\boldsymbol{F}} = \boldsymbol{I} + \gamma \overline{\boldsymbol{M}} \overline{\boldsymbol{F}} \iff \boldsymbol{I} = (\boldsymbol{I} - \gamma \overline{\boldsymbol{M}}) \overline{\boldsymbol{F}} \tag{78}$$

Hence the square matrix $\boldsymbol{I} - \gamma \overline{\boldsymbol{M}}$ is a left inverse of the square matrix $\overline{\boldsymbol{F}}$. By associativity of matrix multiplication, $\boldsymbol{I} + \gamma \overline{\boldsymbol{M}}$ is also a right inverse of $\overline{\boldsymbol{F}}$ and $\overline{\boldsymbol{F}}(\boldsymbol{I} - \gamma \overline{\boldsymbol{M}})$.[8] Consequently, the norm of $\overline{\boldsymbol{F}}^{-1}$ can be bounded with

$$\left|\left|\overline{\boldsymbol{F}}^{-1}\right|\right| = \left|\left|(\boldsymbol{I} - \gamma \overline{\boldsymbol{F}})\right|\right| \leq 1 + \gamma M. \tag{79}$$

For an arbitrary state and action pair s, a,

$$\boldsymbol{\delta}_{s,a}^\top = \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \overline{\boldsymbol{F}} \middle| s, a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{F}_a \tag{80}$$

$$= \boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \overline{\boldsymbol{F}} \middle| s, a\right] - \gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \overline{\boldsymbol{F}} + \gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{\phi}_s^\top \boldsymbol{F}_a \tag{81}$$

$$= \boldsymbol{\phi}_s^\top + \gamma \left(\mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \middle| s, a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a\right) \overline{\boldsymbol{F}} + \gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{\phi}_s^\top \boldsymbol{F}_a \tag{82}$$

8. If $\overline{\boldsymbol{F}}^{-1} \overline{\boldsymbol{F}} = \boldsymbol{I}$ then $\overline{\boldsymbol{F}} = \overline{\boldsymbol{F}} \boldsymbol{I} = \overline{\boldsymbol{F}} (\overline{\boldsymbol{F}}^{-1} \overline{\boldsymbol{F}}) = (\overline{\boldsymbol{F}} \overline{\boldsymbol{F}}^{-1}) \overline{\boldsymbol{F}}$. If $\overline{\boldsymbol{F}}^{-1} \overline{\boldsymbol{F}} \neq \boldsymbol{I}$ were true, then it would contradict $\overline{\boldsymbol{F}} = (\overline{\boldsymbol{F}} \overline{\boldsymbol{F}}^{-1}) \overline{\boldsymbol{F}}$. Hence $\overline{\boldsymbol{F}} \overline{\boldsymbol{F}}^{-1} = \boldsymbol{I}$ and the right inverse exists.

Let $\boldsymbol{\varepsilon}_{s,a}^\top = \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \middle| s,a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a$. Re-arranging the identity in (82) results in

$$\begin{aligned}
\gamma\boldsymbol{\varepsilon}_{s,a}^\top\overline{\boldsymbol{F}} &= \boldsymbol{\delta}_{s,a}^\top - \boldsymbol{\phi}_s^\top - \gamma\boldsymbol{\phi}_s^\top\boldsymbol{M}_a\overline{\boldsymbol{F}} + \boldsymbol{\phi}_s^\top\boldsymbol{F}_a && \iff \\
\gamma\boldsymbol{\varepsilon}_{s,a}^\top &= \boldsymbol{\delta}_{s,a}^\top\overline{\boldsymbol{F}}^{-1} - \boldsymbol{\phi}_s^\top\overline{\boldsymbol{F}}^{-1} - \gamma\boldsymbol{\phi}_s^\top\boldsymbol{M}_a + \boldsymbol{\phi}_s^\top\boldsymbol{F}_a\overline{\boldsymbol{F}}^{-1} && \iff \text{(by (78))} \\
\gamma\boldsymbol{\varepsilon}_{s,a}^\top &= \boldsymbol{\delta}_{s,a}^\top\overline{\boldsymbol{F}}^{-1} - \boldsymbol{\phi}_s^\top\overline{\boldsymbol{F}}^{-1} - \gamma\boldsymbol{\phi}_s^\top\boldsymbol{M}_a + \boldsymbol{\phi}_s^\top(\boldsymbol{I} + \gamma\boldsymbol{M}_a\overline{\boldsymbol{F}})\overline{\boldsymbol{F}}^{-1} && \iff \text{(by } \Delta = 0\text{)} \\
\gamma\boldsymbol{\varepsilon}_{s,a}^\top &= \boldsymbol{\delta}_{s,a}^\top\overline{\boldsymbol{F}}^{-1} - \boldsymbol{\phi}_s^\top\overline{\boldsymbol{F}}^{-1} - \gamma\boldsymbol{\phi}_s^\top\boldsymbol{M}_a + \boldsymbol{\phi}_s^\top\overline{\boldsymbol{F}}^{-1} + \gamma\boldsymbol{\phi}_s^\top\boldsymbol{M}_a && \iff \\
\gamma\boldsymbol{\varepsilon}_{s,a}^\top &= \boldsymbol{\delta}_{s,a}^\top\overline{\boldsymbol{F}}^{-1} && \iff \\
\gamma\left|\left|\boldsymbol{\varepsilon}_{s,a}^\top\right|\right| &\leq \left|\left|\boldsymbol{\delta}_{s,a}^\top\right|\right| \left|\left|\overline{\boldsymbol{F}}^{-1}\right|\right| && \iff \\
\gamma\left|\left|\boldsymbol{\varepsilon}_{s,a}^\top\right|\right| &\leq \varepsilon_\psi \left|\left|\overline{\boldsymbol{F}}^{-1}\right|\right| && \iff \text{(by (26))} \\
\left|\left|\boldsymbol{\varepsilon}_{s,a}^\top\right|\right| &\leq \varepsilon_\psi(1+\gamma M)/\gamma && \iff \text{(by (79))} \qquad (83)
\end{aligned}$$

Note that the bound in Equation (83) does not depend on the state and action pair s, a and thus

$$\forall s,a,\ \left|\left|\mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top\middle|s,a\right] - \boldsymbol{\phi}_s^\top\boldsymbol{M}_a\right|\right| \leq \varepsilon_\psi(1+\gamma M)/\gamma \implies \varepsilon_p \leq \varepsilon_\psi(1+\gamma M)/\gamma. \qquad (84)$$

□

The following lemma is a restatement of Lemma 1 in the main paper.

Lemma 6. *For an MDP, a state representation ϕ, a LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ and a LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ where $\Delta \geq 0$,*

$$\varepsilon_p \leq \varepsilon_\psi \frac{1+\gamma M}{\gamma} + C_{\gamma,M,N}\Delta, \qquad (85)$$

where $C_{\gamma,M,N} = \frac{(1+\gamma)(1+\gamma M)N}{\gamma(1-\gamma M)}$

Proof. The proof reuses and extends the bound shown in Lemma 5. Using the LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$, construct an LSFM $\{\boldsymbol{F}_a^*, \boldsymbol{w}_a^*\}_{a\in\mathcal{A}}$ such that

$$\boldsymbol{F}_a^* = \boldsymbol{I} + \gamma\boldsymbol{M}_a \underbrace{\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\boldsymbol{F}_a^*}_{=\overline{\boldsymbol{F}}^*}. \qquad (86)$$

If

$$\varepsilon_\psi^* = \sup_{s,a}\left|\boldsymbol{\phi}_s^\top + \gamma\mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top\overline{\boldsymbol{F}}^*\middle|s,a\right] - \boldsymbol{\phi}_s^\top\boldsymbol{F}_a^*\right|, \qquad (87)$$

then

$$\varepsilon_p \leq \varepsilon_\psi^* \frac{1+\gamma M}{\gamma}, \qquad (88)$$

by Lemma 5. By linearity of the expectation operator, the SF-error for the LSFM $\{\boldsymbol{F}^*_a, \boldsymbol{w}^*_a\}_{a\in\mathcal{A}}$ and LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ can be founded for any arbitrary state and action pair s, a with

$$\Big|\Big| \underbrace{\left(\boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \overline{\boldsymbol{F}}^* \middle| s, a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{F}^*_a\right)}_{=\boldsymbol{\delta}^*_{s,a}} - \underbrace{\left(\boldsymbol{\phi}_s^\top + \gamma \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \overline{\boldsymbol{F}} \middle| s, a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{F}_a\right)}_{\boldsymbol{\delta}_{s,a}} \Big|\Big| \tag{89}$$

$$\leq \gamma N \left(\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}\right) + N\left(\boldsymbol{F}^*_a - \boldsymbol{F}_a\right). \tag{90}$$

As stated in Equation (89), the SF errors for the LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ are defined as $\boldsymbol{\delta}_{s,a}$ and for the LSFM $\{\boldsymbol{F}^*_a, \boldsymbol{w}^*_a\}_{a\in\mathcal{A}}$ as $\boldsymbol{\delta}^*_{s,a}$. Because the LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ has a $\Delta > 0$, we define

$$\overline{\boldsymbol{\Delta}} = \frac{1}{|\mathcal{A}|}\sum_a \boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a \tag{91}$$

$$= \boldsymbol{I} + \gamma \overline{\boldsymbol{M}\boldsymbol{F}} - \overline{\boldsymbol{F}}. \tag{92}$$

By Equation (91), $||\overline{\boldsymbol{\Delta}}|| \leq \frac{1}{|\mathcal{A}|}\sum_a ||\boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a|| \leq \Delta$ (by triangle inequality). Reusing Equation (92) one can write,

$$\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}} = \boldsymbol{I} + \gamma \overline{\boldsymbol{M}\boldsymbol{F}}^* - \boldsymbol{I} - \gamma \overline{\boldsymbol{M}\boldsymbol{F}} + \overline{\boldsymbol{\Delta}} \tag{93}$$

$$= \gamma \overline{\boldsymbol{M}}(\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}) + \overline{\boldsymbol{\Delta}} \tag{94}$$

$$\iff ||\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}|| \leq \gamma M ||\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}|| + \Delta \tag{95}$$

$$\iff ||\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}|| \leq \frac{\Delta}{1-\gamma M} \tag{96}$$

Similarly, define

$$\boldsymbol{\Delta}_a = \boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a, \tag{97}$$

then,

$$\boldsymbol{F}^*_a - \boldsymbol{F}_a = \boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}}^* - \boldsymbol{I} - \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} + \boldsymbol{\Delta}_a \tag{98}$$

$$= \gamma \boldsymbol{M}_a(\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}) + \boldsymbol{\Delta}_a \tag{99}$$

$$\iff ||\boldsymbol{F}^*_a - \boldsymbol{F}_a|| \leq \gamma M ||\overline{\boldsymbol{F}}^* - \overline{\boldsymbol{F}}|| + \Delta \tag{100}$$

$$\leq \gamma M \frac{\Delta}{1-\gamma M} + \Delta \tag{101}$$

$$= \frac{\Delta}{1-\gamma M} \tag{102}$$

Substituting lines (96) and (102) into (90),

$$\left|\left|\boldsymbol{\delta}^*_{s,a} - \boldsymbol{\delta}_{s,a}\right|\right| \leq \frac{(1+\gamma)N\Delta}{1-\gamma M}. \tag{103}$$

For both LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ and $\{\boldsymbol{F}^*_a, \boldsymbol{w}^*_a\}_{a\in\mathcal{A}}$, the worst case SF prediction errors ε_ψ and ε^*_ψ are defined as

$$\varepsilon_\psi = \sup_{s,a} ||\boldsymbol{\delta}_{s,a}|| \text{ and } \varepsilon^*_\psi = \sup_{s,a} ||\boldsymbol{\delta}^*_{s,a}||. \tag{104}$$

To find a bound on $|\varepsilon_\psi - \varepsilon^*_\phi|$, the maximizing state and action pairs are defined as

$$s_{\text{sup}}, a_{\text{sup}} = \arg\sup_{s,a} ||\boldsymbol{\delta}_{s,a}|| \text{ and } s^*_{\text{sup}}, a^*_{\text{sup}} = \arg\sup_{s,a} ||\boldsymbol{\delta}^*_{s,a}||. \tag{105}$$

If $(s_{\text{sup}}, a_{\text{sup}}) = (s^*_{\text{sup}}, a^*_{\text{sup}})$ then

$$\left|\varepsilon_\psi - \varepsilon^*_\psi\right| \leq \frac{(1+\gamma)N\Delta}{1-\gamma M}. \qquad \text{(by (103))} \tag{106}$$

If $(s_{\text{sup}}, a_{\text{sup}}) \neq (s^*_{\text{sup}}, a^*_{\text{sup}})$ and $\varepsilon_\psi \geq \varepsilon^*_\psi$, then

$$\begin{aligned}
\varepsilon_\psi - \varepsilon^*_\psi &= \left|\left|\boldsymbol{\delta}_{s_{\text{sup}},a_{\text{sup}}}\right|\right| - \left|\left|\boldsymbol{\delta}^*_{s^*_{\text{sup}},a^*_{\text{sup}}}\right|\right| && (107)\\
&\leq \left|\left|\boldsymbol{\delta}_{s_{\text{sup}},a_{\text{sup}}}\right|\right| - \left|\left|\boldsymbol{\delta}^*_{s_{\text{sup}},a_{\text{sup}}}\right|\right| && \text{(by (105))} \quad (108)\\
&\leq \left|\left|\boldsymbol{\delta}_{s_{\text{sup}},a_{\text{sup}}} - \boldsymbol{\delta}^*_{s_{\text{sup}},a_{\text{sup}}}\right|\right| && \text{(by inv. triangle in eq.)} \quad (109)\\
&\leq \frac{(1+\gamma)N\Delta}{1-\gamma M}. && \text{(by (103))} \quad (110)
\end{aligned}$$

If $(s_{\text{sup}}, a_{\text{sup}}) \neq (s^*_{\text{sup}}, a^*_{\text{sup}})$ and $\varepsilon^*_\psi \geq \varepsilon_\psi$, then

$$\begin{aligned}
\varepsilon^*_\psi - \varepsilon_\psi &= \left|\left|\boldsymbol{\delta}^*_{s^*_{\text{sup}},a^*_{\text{sup}}}\right|\right| - \left|\left|\boldsymbol{\delta}_{s_{\text{sup}},a_{\text{sup}}}\right|\right| && (111)\\
&\leq \left|\left|\boldsymbol{\delta}^*_{s^*_{\text{sup}},a^*_{\text{sup}}}\right|\right| - \left|\left|\boldsymbol{\delta}_{s^*_{\text{sup}},a^*_{\text{sup}}}\right|\right| && \text{(by (105))} \quad (112)\\
&\leq \left|\left|\boldsymbol{\delta}^*_{s^*_{\text{sup}},a^*_{\text{sup}}} - \boldsymbol{\delta}_{s^*_{\text{sup}},a^*_{\text{sup}}}\right|\right| && \text{(by inv. triangle ineq.)} \quad (113)\\
&\leq \frac{(1+\gamma)N\Delta}{1-\gamma M}. && \text{(by (103))} \quad (114)
\end{aligned}$$

By lines (106), (110), and (114),

$$\left|\varepsilon_\psi - \varepsilon^*_\psi\right| \leq \frac{(1+\gamma)N\Delta}{1-\gamma M} \implies \varepsilon^*_\psi \leq \varepsilon_\psi + \frac{(1+\gamma)N\Delta}{1-\gamma M}. \tag{115}$$

Substituting (115) into (88) results in the desired bound:

$$\varepsilon_p \leq \varepsilon^*_\psi \frac{1+\gamma M}{\gamma} \leq \left(\varepsilon_\psi + \frac{(1+\gamma)N\Delta}{1-\gamma M}\right)\frac{1+\gamma M}{\gamma} = \varepsilon_\psi \frac{1+\gamma M}{\gamma} + \frac{(1+\gamma)(1+\gamma M)N}{\gamma(1-\gamma M)}\Delta. \tag{116}$$

□

Using these lemmas, Theorem 3 can be proven.

Proof of Theorem 3. The proof is by induction on the sequence length T.
Base Case: For $T = 1$,

$$\left|\boldsymbol{\phi}_s^\top \boldsymbol{w}_{a_1} - \mathbb{E}_p\left[r_1 | s, a_1\right]\right| = \left|\boldsymbol{\phi}_s^\top \boldsymbol{w}_{a_1} - r(s, a_1)\right| \leq \varepsilon_r. \tag{117}$$

Induction Step: Assume that the bound (29) holds for T, then for $T+1$,

$$\left|\boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} - \mathbb{E}_p\left[r_{T+1} | s, a_1, ..., a_{T+1}\right]\right| \tag{118}$$

$$= \left|\boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} - \mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} \middle| s, a_1\right]\right.$$
$$\left. + \mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} \middle| s, a_1\right] - \mathbb{E}_p\left[r_{T+1} | s, a_1, ..., a_{T+1}\right]\right| \tag{119}$$

$$\leq \left|\left(\boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} - \mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \middle| s, a_1\right]\right) \boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}}\right|$$
$$\left. + \left|\mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} \middle| s, a_1\right] - \mathbb{E}_p\left[r_{T+1} | s, a_1, ..., a_{T+1}\right]\right|\right. \tag{120}$$

$$\leq \left|\left|\boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} - \mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \middle| s, a_1\right]\right|\right| \cdot \left|\left|\boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}}\right|\right|$$
$$\left. + \left|\mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} \middle| s, a_1\right] - \mathbb{E}_p\left[r_{T+1} | s, a_1, ..., a_{T+1}\right]\right|\right.$$

$$\leq \left|\left|\boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} - \mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \middle| s, a_1\right]\right|\right| \cdot \left|\left|\boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}}\right|\right|$$
$$\left. + \left|\mathbb{E}_p\left[\boldsymbol{\phi}_{s_2}^\top \boldsymbol{M}_{a_2} \cdots \boldsymbol{M}_{a_T} \boldsymbol{w}_{a_{T+1}} - \mathbb{E}_p\left[r_{T+1} | s_1, a_2 ..., a_{T+1}\right] \middle| s, a_1\right]\right|\right. \tag{121}$$

$$\leq \varepsilon_p M^{T-1} W$$
$$+ \varepsilon_p \sum_{t=1}^{T-1} M^t W + \varepsilon_r \tag{122}$$

$$= \varepsilon_p \sum_{t=1}^{(T+1)-1} M^t W + \varepsilon_r. \tag{123}$$

□

Theorem 5. *For an MDP, state representation $\phi : \mathcal{S} \to \mathbb{R}^n$, and for all $T \geq 1, s, a_1, ..., a_T$,*

$$\left|\boldsymbol{\phi}_s^\top \boldsymbol{M}_{a_1} \cdots \boldsymbol{M}_{a_{T-1}} \boldsymbol{w}_{a_T} - \mathbb{E}_p\left[r_T | s, a_1, ..., a_T\right]\right| \leq \left(\varepsilon_\psi \frac{1+\gamma M}{\gamma} + C_{\gamma,M,N} \Delta\right) \sum_{t=1}^{T-1} M^t W + \varepsilon_r.$$

Proof of Theorem 5. The proof is by reusing the bound in Theorem 3 and substituting ε_p with the bound presented in Lemma 1. □

Theorem 4, which is stated in the main paper, can be proven as follows.

Proof of Theorem 4. The value error term can be upper-bounded with

$$\left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| \leq \sum_{a \in \mathcal{A}} \pi(s,a) \left|r(s,a) + \gamma \mathbb{E}_p\left[V^\pi(s') \middle| s, a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{w}_a - \gamma \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{v}^\pi\right| \tag{124}$$

$$\leq \sum_{a \in \mathcal{A}} \pi(s,a) \left|r(s,a) - \boldsymbol{\phi}_s^\top \boldsymbol{w}_a\right| + \gamma \left|\mathbb{E}_p\left[V^\pi(s') \middle| s, a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{v}^\pi\right| \tag{125}$$

The second term in Equation (125) is bounded by

$$\begin{aligned}
&\left|\mathbb{E}_p\left[V^\pi(s')\middle|s,a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{v}^\pi\right| \\
&= \left|\mathbb{E}_p\left[V^\pi(s')\middle|s,a\right] - \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \boldsymbol{v}^\pi\middle|s,a\right] + \mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \boldsymbol{v}^\pi\middle|s,a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{v}^\pi\right| \\
&= \sup_s \left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| + \left|\mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top \boldsymbol{v}^\pi\middle|s,a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{v}^\pi\right| \\
&= \sup_s \left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| + \left|\left|\mathbb{E}_p\left[\boldsymbol{\phi}_{s'}^\top\middle|s,a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{M}_a\right|\right| ||\boldsymbol{v}^\pi|| \\
&= \sup_s \left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| + \varepsilon_p ||\boldsymbol{v}^\pi|| \qquad (126)
\end{aligned}$$

Substituting (126) into (125) results in

$$\left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| \leq \sum_{a\in\mathcal{A}} \pi(s,a)\left(\left|r(s,a) - \boldsymbol{\phi}_s^\top \boldsymbol{w}_a\right| + \gamma\left(\sup_s \left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| + \varepsilon_p ||\boldsymbol{v}^\pi||\right)\right). \qquad (127)$$

Let $B = \sup_s \left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right|$, then

$$\begin{aligned}
\left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| &\leq \sum_{a\in\mathcal{A}} \pi(s,a)(\varepsilon_r + \gamma\left(B + \varepsilon_p ||\boldsymbol{v}^\pi||\right)) \\
&= \varepsilon_r + \gamma B + \gamma\varepsilon_p ||\boldsymbol{v}^\pi|| \qquad (128)
\end{aligned}$$

The bound in Equation (128) does not depend on any particular state and action pair s, a and thus

$$\begin{aligned}
\forall s,a,\ \left|V^\pi(s) - \boldsymbol{\phi}_s^\top \boldsymbol{v}^\pi\right| \leq \varepsilon_r + \gamma B + \gamma\varepsilon_p ||\boldsymbol{v}^\pi|| &\implies B \leq \varepsilon_r + \gamma B + \gamma\varepsilon_p ||\boldsymbol{v}^\pi|| \\
&\implies B \leq \frac{\varepsilon_r + \gamma\varepsilon_\psi ||\boldsymbol{v}^\pi||}{1-\gamma}. \qquad (129)
\end{aligned}$$

To bound the Q-value function,

$$\left|Q^\pi(s,a) - \boldsymbol{\phi}_s^\top \boldsymbol{q}_a\right| \leq \left|r(s,a) + \gamma\mathbb{E}_p\left[V^\pi(s')\middle|s,a\right] - \boldsymbol{\phi}_s^\top \boldsymbol{w}_a - \gamma\boldsymbol{\phi}_s^\top \boldsymbol{M}_a \boldsymbol{v}^\pi\right|, \qquad (130)$$

which is similar to Equation (125) and the proof proceeds in the same way. The LSFM bound

$$\frac{\varepsilon_r + \gamma\varepsilon_p ||\boldsymbol{v}^\pi||}{1-\gamma} \leq \frac{\varepsilon_r + \varepsilon_\psi(1+\gamma M) ||\boldsymbol{v}^\pi|| + \gamma C_{\gamma,M,N}\Delta ||\boldsymbol{v}^\pi||}{1-\gamma} \qquad (131)$$

follows by Lemma 1. □

A.3 Bound on Error Term Δ

The following proposition formally proofs the bound presented in Equation (32).

Proposition 2. *For a data set* $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^D$,

$$\Delta \leq \max_a ||\boldsymbol{\Phi}_a^+||_2^2 \mathcal{L}_\psi, \qquad (132)$$

where each row of $\boldsymbol{\Phi}_a$ *is set to a row-vector* $\boldsymbol{\phi}_s$ *for a transition* $(s,a,r,s') \in \mathcal{D}$ *that uses action* a*, and* $\boldsymbol{\Phi}_a^+$ *is the pseudo-inverse of* $\boldsymbol{\Phi}_a$*.*

Proof of Proposition 2. For a data set $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^{D}$, construct the matrix $\boldsymbol{\Phi}_a$ and similarly construct the matrix $\boldsymbol{\Phi}'_a$ where each row of $\boldsymbol{\Phi}'_a$ is set to a row-vector $\boldsymbol{\phi}_{s'}$ for a transition $(s, a, r, s') \in \mathcal{D}$ that uses action a. The transition matrix of a LAM can be obtained using a least squares regression and

$$\boldsymbol{M}_a = \arg\min_{\boldsymbol{M}} ||\boldsymbol{\Phi}_a \boldsymbol{M} - \boldsymbol{\Phi}'_a||_2^2 \implies \boldsymbol{M}_a = \boldsymbol{\Phi}_a^+ \boldsymbol{\Phi}'_a, \tag{133}$$

where $\boldsymbol{\Phi}_a^+$ is the pseudo-inverse of $\boldsymbol{\Phi}_a$. Using this notation, one can write

$$\boldsymbol{\Phi}_a + \gamma \boldsymbol{\Phi}'_a \overline{\boldsymbol{F}} - \boldsymbol{\Phi}_a \boldsymbol{F}_a = \boldsymbol{L}_a \qquad \iff \tag{134}$$

$$\boldsymbol{\Phi}_a^+ \boldsymbol{\Phi}_a + \gamma \boldsymbol{\Phi}_a^+ \boldsymbol{\Phi}'_a \overline{\boldsymbol{F}} - \boldsymbol{\Phi}_a^+ \boldsymbol{\Phi}_a \boldsymbol{F}_a = \boldsymbol{\Phi}_a^+ \boldsymbol{L}_a \qquad \iff \tag{135}$$

$$\boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a = \boldsymbol{\Phi}_a^+ \boldsymbol{L}_a \qquad \iff \tag{136}$$

$$||\boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a||_2^2 \leq ||\boldsymbol{\Phi}_a^+||_2^2 ||\boldsymbol{L}_a||_2^2. \tag{137}$$

Note that $\mathcal{L}_\psi = \sum_{a \in \mathcal{A}} \boldsymbol{L}_a$, and thus

$$\Delta = \max_a ||\boldsymbol{I} + \gamma \boldsymbol{M}_a \overline{\boldsymbol{F}} - \boldsymbol{F}_a||_2^2 \leq \max_a ||\boldsymbol{\Phi}_a^+||_2^2 \mathcal{L}_\psi. \tag{138}$$

□

Appendix B. Connection Between Q-learning and SF-learning

Proof of Proposition 1. Before proving the main statement, we first make the following observation. Assuming that for some t, $\boldsymbol{\theta}_t = \boldsymbol{F}_t \boldsymbol{w}$, then

$$\boldsymbol{w}^\top \boldsymbol{y}_{s,a,r,s'} = \boldsymbol{w}^\top \left(\boldsymbol{\xi}_{s,a} + \gamma \sum_{a'} b(s', a') \boldsymbol{\psi}^\pi_{s',a'} \right) \tag{139}$$

$$= r(s, a) + \gamma \sum_{a'} b(s', a') \boldsymbol{w}^\top \boldsymbol{\psi}^\pi_{s',a'} \tag{140}$$

$$= r(s, a) + \gamma \sum_{a'} b(s', a') \boldsymbol{w}^\top \boldsymbol{F}_t \boldsymbol{\xi}_{s',a'} \tag{141}$$

$$= r(s, a) + \gamma \sum_{a'} b(s', a') \boldsymbol{\theta}_t^\top \boldsymbol{\xi}_{s',a'} \tag{142}$$

$$= y_{s,a,r,s'}. \tag{143}$$

Equation (141) follows by substitution with Equation (45). The proof of the main statement is by induction on t.
Base Case: For $t = 1$, assume $\boldsymbol{\theta}_0 = \boldsymbol{F}_0 \boldsymbol{w}$. Then

$$\boldsymbol{w}^\top \boldsymbol{F}_1 = \boldsymbol{w}^\top \left(\boldsymbol{F}_0 + \alpha_\psi \left(\boldsymbol{F}_0 \boldsymbol{\xi}_{s,a} - \boldsymbol{y}_{s,a,r,s'} \right)^\top \boldsymbol{\xi}_{s,a} \right) \tag{144}$$

$$= \boldsymbol{w}^\top \boldsymbol{F}_0 + \alpha_\psi \left(\boldsymbol{w}^\top \boldsymbol{F}_0 \boldsymbol{\xi}_{s,a} - \boldsymbol{w}^\top \boldsymbol{y}_{s,a,r,s'} \right)^\top \boldsymbol{\xi}_{s,a} \tag{145}$$

$$= \boldsymbol{\theta}_0 + \alpha_\psi \left(\boldsymbol{\theta}_0^\top \boldsymbol{\xi}_{s,a} - y_{s,a,r,s'} \right)^\top \boldsymbol{\xi}_{s,a} \tag{146}$$

$$= \boldsymbol{\theta}_1. \tag{147}$$

Equation (146) us obtained by substituting the identity in Equation (143) for $t = 0$. Equation (147) is obtained by substituting the linear TD iterate from Equation (42).
Induction Step: Assuming the hypothesis $\boldsymbol{w}^\top \boldsymbol{F}_t = \boldsymbol{\theta}_t^\top$ holds for t and proceeding as in the base case, then

$$\boldsymbol{w}^\top \boldsymbol{F}_{t+1} = \boldsymbol{w}^\top \left(\boldsymbol{F}_t + \alpha_\psi \left(\boldsymbol{F}_t \boldsymbol{\xi}_{s,a} - \boldsymbol{y}_{s,a,r,s'} \right)^\top \boldsymbol{\xi}_{s,a} \right) \tag{148}$$

$$= \boldsymbol{w}^\top \boldsymbol{F}_t + \alpha_\psi \left(\boldsymbol{w}^\top \boldsymbol{F}_t \boldsymbol{\xi}_{s,a} - \boldsymbol{w}^\top \boldsymbol{y}_{s,a,r,s'} \right)^\top \boldsymbol{\xi}_{s,a} \tag{149}$$

$$= \boldsymbol{\theta}_t + \alpha_\psi \left(\boldsymbol{\theta}_t^\top \boldsymbol{\xi}_{s,a} - y_{s,a,r,s'} \right)^\top \boldsymbol{\xi}_{s,a} \tag{150}$$

$$= \boldsymbol{\theta}_{t+1}. \tag{151}$$

Hence for all t, $\boldsymbol{w}^\top \boldsymbol{F}_t = \boldsymbol{\theta}_t$, as desired.

Note that this proof assumes that both iterates are applied for exactly the same transitions. This assumption is not restrictive assuming that control policies are constructed using the current parameters $\boldsymbol{\theta}_t$ in the case for TD-learning or the parameters $\boldsymbol{F}_t$ and $\boldsymbol{w}$ in the case for SF-learning. Even in the control case, where an ε-greedy exploration strategy is used, for example, both algorithms will produce an identical sequence of value functions and will chose actions with equal probability. □

Appendix C. Experiment Design

The presented experiments are conducted on finite MDPs and use a state representation function

$$\phi : s \mapsto \boldsymbol{\Phi}(s, :), \tag{152}$$

where $\boldsymbol{\Phi}$ is a $\mathcal{S} \times n$ matrix and $\boldsymbol{\Phi}(s, :)$ is a row with state index s. The feature dimension n is a fixed hyper parameter for each experiment.

C.1 Matrix Optimization in Column World

The column world experiment (Figure 5) learns a state representation using the full transition and reward tables. Assume that the transition table of the column world task is stored as a set of stochastic transition matrices $\{\boldsymbol{P}_a\}_{a \in \mathcal{A}}$ and the reward table as a set of reward vectors $\{\boldsymbol{r}_a\}_{a \in \mathcal{A}}$. The one-step reward prediction errors and linear SF prediction errors are minimized for the LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ using the loss objective

$$\mathcal{L}_{\text{LSFM-mat}} = \sum_{a \in \mathcal{A}} ||\boldsymbol{\Phi} \boldsymbol{w}_a - \boldsymbol{r}_a||_2^2 + \alpha_\psi ||\boldsymbol{\Phi} + \gamma \boldsymbol{P}_a \boldsymbol{\Phi} \overline{\boldsymbol{F}} - \boldsymbol{\Phi} \boldsymbol{F}_a||_2^2. \tag{153}$$

For $\alpha_\psi = 1$, the loss objective $\mathcal{L}_{\text{LSFM-mat}}$ is optimized with respect to all free parameters $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ and $\boldsymbol{\Phi}$. Similarly, a LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ is computed using the loss objective

$$\mathcal{L}_{\text{LAM-mat}} = \sum_{a \in \mathcal{A}} ||\boldsymbol{\Phi} \boldsymbol{w}_a - \boldsymbol{r}_a||_2^2 + ||\boldsymbol{\Phi} \boldsymbol{M}_a - \boldsymbol{P}_a \boldsymbol{\Phi}||_2^2. \tag{154}$$

This loss objective is optimized with respect to all free parameters $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a \in \mathcal{A}}$ and $\boldsymbol{\Phi}$. Both experiments used the Adam optimizer (Kingma and Ba, 2014) with a learning rate of

Hyper-Parameter	LAM	LSFM	Tested Values
Learning Rate	0.0005	0.0005	0.0001, 0.0005, 0.001, 0.005
α_ψ	-	0.01	0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0
α_p	1.0	-	0.0001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0
α_N	0.1	0.0	0.0, 0.0001, 0.001, 0.01, 0.1
Feature Dimension	80	80	
Batch Size	50	50	
Number of Training Transitions	10000	10000	
Number of Gradient Steps	50000	50000	

Table 2: Hyper-Parameter for Puddle-World Experiment

0.1 and Tensorflow (Abadi et al., 2015) default parameters. Optimization was initialized by sampling entries for $\boldsymbol{\Phi}$ uniformly from the interval $[0, 1]$. The LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ or LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ was initialized using a least squares solution for the initialization of $\boldsymbol{\Phi}$.

C.2 Puddle-World Experiment

In the puddle world MDP transitions are probabilistic, because with a 5% chance, the agent does not move after selecting any action. The partition maps presented in Figures 6(b) and 6(c) were obtained by clustering latent state vectors using agglomerative clustering. A finite data set of transitions $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}_{i=1}^D$ was collected by selecting actions uniformly as random. Given such a data set $\mathcal{D}$, the loss objective

$$\mathcal{L}_{\text{LAM}} = \underbrace{\sum_{i=1}^{D} \left(\boldsymbol{\phi}_{s_i}^\top \boldsymbol{w}_{a_i} - r_i\right)^2}_{=\mathcal{L}_r} + \alpha_p \underbrace{\sum_{i=1}^{D} \left|\left|\boldsymbol{\phi}_{s_i}^\top \boldsymbol{M}_a - \boldsymbol{\phi}_{s'_i}^\top\right|\right|_2^2}_{=\mathcal{L}_p} + \alpha_N \underbrace{\sum_{i=1}^{D} \left(\left|\left|\boldsymbol{\phi}_{s_i}\right|\right|_2^2 - 1\right)^2}_{=\mathcal{L}_N},$$

is used to approximate a reward-predictive state representation using a LAM. Optimization was initialized by each entry of the matrix $\boldsymbol{\Phi}$ uniformly at random and then finding a LAM $\{\boldsymbol{M}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ for this initialized representation using least squares regression.

For the LSFM experiment, the matrix $\boldsymbol{\Phi}$ was also initialized using values sampled uniformly at random. The LSFM $\{\boldsymbol{F}_a, \boldsymbol{w}_a\}_{a\in\mathcal{A}}$ was set to zero matrices and vectors at initialization. Both loss objective functions were optimized using the Adam optimizer with Tensorflow's default parameters. Table 2 lists the hyper-parameter that were found to work best for each model. Figures 6(h) and 6(i) are plotted by first evaluating an ε-greedy policy using the full transition and reward tables of the task. Then the state representation is used to find an approximation of the value functions for each ε setting using least-squares linear regression. Each curve then plots the maximum value prediction error.

C.3 Transfer Experiments

For the transfer experiment presented in Section 5.2, a training data set of 10000 transitions was collected from Task B. The LSFM was trained using the hyper-parameter listed in Table 3.

Hyper-Parameter	LSFM	Tested Values
Learning Rate	0.001	0.0001, 0.0005, 0.001, 0.005
α_ψ	0.0001	0.001, 0.001, 0.01, 0.1, 1.0, 10.0, 100.0
α_N	0.0	0.0, 0.0001, 0.001, 0.01, 0.1
Feature Dimension	50	
Batch Size	50	
Number of Training Transitions	10000	
Number of Gradient Steps	50000	

Table 3: Hyper-Parameter for LSFM on Task A

Value-predictive state representations are learned using a modified version of Neural Fitted Q-iteration (Riedmiller, 2005). The Q-value function is computed with

$$Q(s, a; \mathbf{\Phi}, \{\boldsymbol{q}_a\}_{a\in\mathcal{A}}) = \boldsymbol{\phi}_s^\top \boldsymbol{q}_a, \tag{155}$$

where the state features $\boldsymbol{\phi}_s$ are computed as shown in Equation (152). To find a value-predictive state representation, the loss function

$$\mathcal{L}_q(\mathbf{\Phi}, \{\boldsymbol{q}_a\}_{a\in\mathcal{A}}) = \sum_{i=1}^{D} (Q(s_i, a_i; \mathbf{\Phi}, \{\boldsymbol{q}_a\}_{a\in\mathcal{A}}) - y_{s_i,a_i,r_i,s_i'})^2 \tag{156}$$

is minimized using stochastic gradient descent on a transition data set $\mathcal{D} = \{(s_i, a_i, r_i, s_i')\}_{i=1}^D$. When differentiating the loss objective $\mathcal{L}_q$ with respect to its free parameters $\mathbf{\Phi}, \{\boldsymbol{q}_a\}_{a\in\mathcal{A}}$, the prediction target

$$y_{s,a,r,s'} = r + \gamma \max_{a'} Q(s', a'; \mathbf{\Phi}, \{\boldsymbol{q}_a\}_{a\in\mathcal{A}}) \tag{157}$$

is considered a constant and no gradient of $y_{s,a,r,s'}$ is computed. A value-predictive state representation is learned for Task A by optimizing over all free parameters $\mathbf{\Phi}, \{\boldsymbol{q}_a\}_{a\in\mathcal{A}}$. Table 4 lists the used hyper-parameter. For Task A the best hyper-parameter setting was obtained by testing the learned state representation on Task A and using the model that produced the shortest episode length, averaged over 20 repeats.

On Task B, a previously learned state representation is evaluated using the same implementation of Fitted Q-iteration, but the previously learned state representation is re-used and considered a constant. At transfer, gradients of $\mathcal{L}_q$ are only computed with respect to the vector set $\{\boldsymbol{q}_a\}_{a\in\mathcal{A}}$ and the feature matrix $\mathbf{\Phi}$ is held constant.

The tabular model first compute a partial transition and reward table of Task B by averaging different transitions and reward using the given data set. If no transition is provided for a particular state and action pair, uniform transitions are assumed. If no reward is provided for a particular state and action pair, a reward value is sampled uniformly at random from the interval $[0, 1]$. Augmenting a partial transition and reward table is equivalent to providing the agent with a uniform prior over rewards and transitions. The tabular model's optimal policy is computed using value iteration.

To plot the right panel in Figure 8(c), twenty different transition data sets of a certain fixed size were collected and the Fitted Q-iteration algorithm was used to approximate the optimal value function. For both tested state representations and data sets, a small

Hyper-Parameter	Fitted Q-iteration, learning on Task A	Fitted Q-iteration, evaluation on Task B	Tested Values
Learning Rate	0.001	0.00001	0.00001, 0.0001, 0.001, 0.01
Feature Dimension	50	50	
Batch Size	50	50	
Training Transitions	10000	Varies	
Gradient Steps	50000	20000	

Table 4: Hyper-Parameter used for Fitted Q-iteration

enough learning rate was found to guarantee that Fitted Q-iteration converges. The found solutions were evaluated twenty times, and if all evaluation completed the navigation task within 22 time steps (which is close to optimal), then this data set is considered to be optimally solved. Note that all tested evaluation runs either complete within 22 time steps or hit the timeout threshold of 5000 time steps. Table 4 lists the hyper-parameters used for Fitted Q-iteration to obtain the right panel in Figure 8(c). For transfer evaluation, the hyper-parameter setting was used that approximated the Q-values optimal in Task B with the lowest error.

C.4 Combination Lock Experiment

For the combination lock simulations presented in Figure 9, each Q-learning configuration was tested independently on each task with learning rates 0.1, 0.5, and 0.9. Because Q-values were initialized optimistically with a value of 1.0, each plot in Figure 9 uses a learning rate of 0.9.

To find a reward-predictive state representation, the LSFM loss objective $\mathcal{L}_{\text{LSFM-mat}}$ (Equation (153)) was optimized 100000 iterations using Tensorflow's Adam optimizer with a learning rate of 0.005 and $\alpha_\psi = 0.01$.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

David Abel, David Hershkowitz, and Michael Littman. Near optimal behavior via approximate state abstraction. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2915–2923, 2016.

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.

David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L. Littman, and Lawson L.S. Wong. State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

Kavosh Asadi, Dipendra Misra, and Michael Littman. Lipschitz continuity in model-based 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 264–273, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/asadi18a.html.

Mohammad Gheshlaghi Azar, Ian Osband, and Rémi Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 263–272. JMLR.org, 2017.

André Barreto, Rémi Munos, Tom Schaul, and David Silver. Successor features for transfer in reinforcement learning. *CoRR*, abs/1606.05312, 2016. URL http://arxiv.org/abs/1606.05312.

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, 2017a.

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, 2017b.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. 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 501–510, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/barreto18a.html.

Richard E Bellman. *Adaptive Control Processes: A Guided Tour*, volume 2045. Princeton University Press, 1961.

Dimitri P Bertsekas. Dynamic programming and optimal control 3rd edition, volume ii. *Belmont, MA: Athena Scientific*, 2011.

Justin A Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. In *Advances in Neural Information Processing Systems*, pages 369–376, 1995.

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

Eyal Even-Dar and Yishay Mansour. Approximate equivalence of Markov decision processes. In *Learning Theory and Kernel Machines*, pages 581–594. Springer, 2003.

Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 162–169. AUAI Press, 2004.

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

Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined reinforcement learning via abstract representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3582–3589, 2019.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adverserial inverse reinforcement learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=rkHywl-A-`.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179, 2019.

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

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is Q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

George Konidaris, Sarah Osentoski, and Philip Thomas. Value function approximation in reinforcement learning using the Fourier basis. *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages pages 380–385, August 2011.

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

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.

Lucas Lehnert, Michael J Frank, and Michael L Littman. Reward predictive representations generalize across tasks in reinforcement learning. *BioRxiv*, page 653493, 2019.

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

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2018.

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.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *arXiv preprint arXiv:1707.03497*, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) efficient reinforcement learning via posterior sampling. In *Advances in Neural Information Processing Systems*, pages 3003–3011, 2013.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 752–759. ACM, 2008.

Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Sherry Shanshan Ruan, Gheorghe Comanici, Prakash Panangaden, and Doina Precup. Representation discovery for mdps using bisimulation metrics. In *AAAI*, pages 3578–3584, 2015.

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.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pages 3191–3199. PMLR, 2017.

Zhao Song, Ronald E Parr, Xuejun Liao, and Lawrence Carin. Linear feature encoding for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4224–4232, 2016.

Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643, 2017.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, pages 1038–1044, 1996.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book. MIT Press, Cambridge, MA, 1 edition, 1998.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2018.

Richard S. Sutton, Csaba Szepesvári, Alborz Geramifard, and Michael Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, 2008.

Erik Talvitie. Learning the reward function for a misspecified model. 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 4838–4847, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/talvitie18a.html`.

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

Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

Hengshuai Yao and Csaba Szepesvári. Approximate policy iteration with linear action models. In *AAAI*, 2012.

Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378. IEEE, 2017.