Accelerating Learning in Constructive Predictive Frameworks with the Successor Representation

Craig Sherstan[1], Marlos C. Machado[1], Patrick M. Pilarski[1,2]

Abstract— **We propose using the *Successor Representation* (SR) to accelerate learning in a constructive knowledge system based on *General Value Functions* (GVFs). In real-world settings, like robotics for unstructured and dynamic environments, it is impossible to model all meaningful aspects of a system and its environment by hand. Instead, robots must learn and adapt to changes in their environment and task, incrementally constructing models from their own experience. GVFs, taken from the field of reinforcement learning (RL), are a way of modeling the world as predictive questions. One approach to such models proposes a massive network of interconnected and interdependent GVFs, which are incrementally added over time. It is reasonable to expect that new, incrementally added predictions can be learned more swiftly if the learning process leverages knowledge gained from past experience. The SR provides a means of capturing regularities that can be reused across multiple GVFs by separating the dynamics of the world from the prediction targets. As a primary contribution of this work, we show that using the SR can improve sample efficiency and learning speed of GVFs in a continual learning setting where new predictions are incrementally added and learned over time. We analyze our approach in a grid-world and then demonstrate its potential on data from a physical robot arm.**

I. INTRODUCTION

A long standing goal in the pursuit of artificial general intelligence is that of knowledge construction—incrementally modeling and explaining the world and the agent's interaction with it directly from the agent's own experience [1]. This is particularly important in fields such as continual learning [2] and developmental robotics [3], where we expect agents to be capable of learning, dynamically and incrementally, to interact and succeed in complex environments.

One proposed approach for representing such world models is a collection of *General Value Functions* (GVFs) [4], which models the world as a set of predictive questions each defined by a policy of interest, a target signal, and a timescale (discounting schedule) for accumulating the signal of interest. For example, a GVF on a mobile robot could pose the question "How much current will my wheels consume over the next second if I drive straight forward?" Such predictive models should enable a robot to understand its environment, its body and the consequences of its actions. As these models continually improve so too the robot's ability to act should also improve.

GVF questions are typically answered using temporal-difference (TD) methods [5] from the field of reinforcement learning (RL) [6]. A learned GVF approximates the expected future value of a signal of interest, directly representing the relationship between the environment, policy, timescale, and target signal as the output of a single predictive unit.

Nevertheless, despite the success RL algorithms have achieved recently (e.g., [7], [8]), methods for answering multiple predictive questions from a single stream of experience (critical in a robotic setting) are known to exhibit sample inefficiency. In our setting of interest, where multitudes of GVFs are learned in an incremental, sample by sample way, this problem is multiplied. Ultimately, the faster an agent can learn to approximate a new GVF, the better.

This paper focuses on the problem of model construction. Specifically, we show how one can accelerate learning in a constructive knowledge system based on GVFs by sharing the environment dynamics across the different predictors. This is done with the successor representation (SR) [9], which allows us to learn the world dynamics under a policy independently of any signal being predicted.

We empirically demonstrate the effectiveness of our approach on both a tabular representation and on a robot arm which uses function approximation. We evaluate our algorithm in the continual learning setting where it is not possible to specify all GVFs *a priori*, but rather GVFs are added incrementally during the course of learning. As a key result, we show that using a learned SR enables an agent to learn newly added GVFs faster than when learning the same GVFs in the standard fashion without the use of the SR.

II. BACKGROUND

We consider an agent interacting with the environment sequentially. We use the standard notation in the reinforcement learning (RL) literature [6], modeling the problem as a Markov Decision Process. Starting from state $S_0 \in \mathcal{S}$, at each timestep the agent chooses an action, $A_t \in \mathcal{A}$, according to the policy distribution $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$, and transitions to state $S_{t+1} \in \mathcal{S}$ according to the probability transition $p(\cdot|S_t, A_t)$. For each transition $S_t \xrightarrow{A_t} S_{t+1}$ the agent receives a reward, R_t, from the reward function $R(S_t, A_t, S_{t+1}) \in \mathbb{R}$.

In this paper we focus on the prediction problem in RL, in which the agent's goal is to predict the value of a signal from its current state (e.g., the cumulative sum of future rewards). Note that throughout this paper we use upper case letters to indicate random variables.

A. General Value Functions (GVFs)

The most common prediction made in RL is about the expected return. The return is defined to be the sum of future

[1] University of Alberta, Canada
{sherstan, machado, pilarski}@ualberta.ca
[2] DeepMind

discounted rewards under policy π starting from state s. Formally, $G_t = \sum_{t=0}^{T} \gamma^t R_t$, with $\gamma \in [0,1]$[1] being the discount factor and T being the final timestep, where $T = \infty$ denotes a continuing task. The function encoding the prediction about the return is known as the value function $v_\pi(s) = \mathbb{E}_\pi[G_t | S_0 = s]$.

GVFs [4] extend the notion of predictions to different signals in the environment. This is done by replacing the reward signal, R_t, by any other target signal, which we refer to as the cumulant, C_t, and by allowing a state-dependent discounting function, $\gamma_t = \gamma(S_t)$, instead of using a fixed discounting factor. The general value of state s under policy π is defined as:

$$v_{\pi:\gamma}(s) = \bar{c}_\pi + \bar{\gamma}_\pi \sum_{s'} p_\pi(s'|s) v_\pi(s'), \tag{1}$$

where $\bar{c}_\pi$ is the average cumulant from state s, $\bar{\gamma}_\pi$ is the average γ from state s, and $p_\pi(s'|s)$ is the probability of transitioning from state s to s' under policy π. This can also be written in matrix form: $\mathbf{v}_{\pi:\gamma} = \bar{\mathbf{c}}_\pi + \bar{\boldsymbol{\gamma}}_\pi \odot P_\pi \mathbf{v}_{\pi:\gamma}$, where $\odot$ denotes element-wise multiplication. Such an equation, when solved, gives us

$$\mathbf{v}_{\pi:\gamma} = (I - \bar{\boldsymbol{\gamma}}_\pi \odot P_\pi)^{-1} \bar{\mathbf{c}}_\pi, \tag{2}$$

where I is the identity matrix, $\mathbf{v}, \bar{\mathbf{c}}, \bar{\boldsymbol{\gamma}} \in \mathbb{R}^{|\mathcal{S}| \times 1}$, and $P_\pi \in [0,1]^{|\mathcal{S}| \times |\mathcal{S}|}$ is a probability matrix such that $[P_\pi]_{ij} = \mathbb{E}[p_\pi(S_{t+1} = s_j | S_t = s_i)]$.

B. The Successor Representation (SR)

The successor representation [9] was initially proposed as a representation capable of capturing state similarity in terms of time. It is formally defined, for a fixed $\gamma < 1$, as:

$$\psi_\pi(s, s') = \mathbb{E}_\pi\Big[\sum_{t=0}^{\infty} \gamma^t \mathbb{1}_{\{S_t = s'\}} | S_0 = s\Big].$$

In words, the SR encodes the expected number of times the agent will visit a particular state when the sum is discounted by γ over time[2]. It can be re-written in matrix form as

$$\Psi_\pi = \sum_{t=0}^{\infty} (\gamma P_\pi)^t = (I - \gamma P_\pi)^{-1}. \tag{3}$$

Importantly, the SR can be easily computed, incrementally, by standard RL algorithms such as TD learning [5], since its primary modification is to replace the reward signal by a state visitation counter. Nevertheless, despite its simplicity, the SR holds important properties we leverage in this paper.

The SR, in the limit, for a constant γ (see Eq. 3), corresponds to the first factor of the solution in Eq. 2. Thus, the SR can be seen as encoding the dynamics of the Markov chain induced by the policy π and by the environment's transition probability p. If the agent has access to the SR, it can accurately predict the (discounted) accumulated value of any signal, from any state, by simply learning the expected *immediate* value, $\bar{\mathbf{c}}_\pi$, of that signal in each state. On the other hand, if the agent does not use the SR, the agent must also deal with the problem of credit assignment, having to look at n-step returns to control for *delayed* consequences. Importantly, the dynamics encoded by the SR are the same for all signals learned under the same policy and discount function. This factorization of the solution is the main property we use in our work, described in the next section.

III. METHODS

As aforementioned, we are interested in the problem of knowledge acquisition in the continual learning setting [2], where knowledge is encoded as predictive questions (GVFs). In this setting it is not possible to specify all GVFs ahead of time. Instead, GVFs must be added incrementally by some, as yet unknown, mechanism. The standard approach would be to learn each newly added prediction from scratch. In this section we discuss how we can use the SR to accelerate learning by taking advantage of the factorization shown in Eq. 2. Our method leverages the fact that the SR is independent of the target signal being predicted, learning the SR separately and re-using it when learning to predict new signals.

In the previous section, for clarity, we discussed the main concepts in the tabular case. In real world applications, where the state space is too large, assuming states can be uniquely identified is not often feasible. Instead, we generally represent states as a set of features $\boldsymbol{\phi}(s) \in \mathbb{R}^d$ where $d \ll |\mathcal{S}|$. Because both ψ and $\bar{c}$ are a function of feature vector $\boldsymbol{\phi}(S)$, they can easily be represented using function approximation and learned using TD algorithms. In order to present a more general version of our algorithm, we introduce it here using the function approximation notation.

The first step in our algorithm is to compute the one-step average cumulant, which we do with TD error:

$$\delta_t = C_{t+1} - \bar{c}\big(\boldsymbol{\phi}(S_t)\big). \tag{4}$$

If we use linear function approximation to estimate $\bar{c}$ then $\bar{c}\big(\boldsymbol{\phi}(s)\big) = \boldsymbol{\phi}(s)^\top \mathbf{w}$. The TD error for ψ is given as (note that $\boldsymbol{\delta}$ is a vector of length d)

$$\boldsymbol{\delta}_t = \boldsymbol{\phi}(S_t) + \gamma_{t+1} \psi\big(\boldsymbol{\phi}(S_{t+1})\big) - \psi\big(\boldsymbol{\phi}(S_t)\big). \tag{5}$$

This generalization of the SR to the function approximation case is known as successor features [10].

If we use linear function approximation to estimate ψ then $\psi\big(\boldsymbol{\phi}(s)\big) = M^\top \boldsymbol{\phi}(s)$, where $M \in \mathbb{R}^{d \times d}$. Using the usual semi-gradient method,[3] often used with TD, we derive a TD(0) stochastic gradient descent update as:

$$\begin{aligned} M_{t+1} &= M_t - \frac{1}{2} \alpha \nabla_M (\boldsymbol{\delta}_t^2) \\ &= M_t + \alpha \nabla_M \big(\psi(S_t)\big) \otimes \boldsymbol{\delta}_t \\ &= M_t + \alpha \boldsymbol{\phi}(S_t) \otimes \boldsymbol{\delta}_t, \end{aligned}$$

[1]Note that $\gamma = 1$ is only valid when termination is guaranteed as in the episodic case.

[2]Note that Dayan describes the SR as predicting future state visitation from time t onward. This is non-standard in RL as we typically describe the return as predicting the signal from $t+1$ onward.

[3]In semi-gradient methods, the effect of M on the prediction target $\boldsymbol{\phi}(S_{t+1}) + \gamma \psi\big(\boldsymbol{\phi}(S_{t+1})\big)$ is ignored when computing the gradient.

Algorithm 1 GVF prediction with the SR

```
Input: Feature representation $\phi$, policy $\pi$, discount function $\gamma$, and step-sizes $\alpha_C$, $\alpha_{\text{SR}}$
Output: Matrix $M$ and vectors $\bar{c}_i$ as predictors of $C_i$

Initialize $\mathbf{w}$ and $M$ arbitrarily
while $S'$ is not terminal do
    Observe state $S$, take action $A$ selected according to $\pi(S)$, and observe a next state $S'$ and the cumulants $C_i$
    $\delta_{\text{SR}} = \phi(S) + \gamma(S')M^\top \phi(S') - M^\top \phi(S)$
    $M \leftarrow M + \alpha_{\text{SR}}\phi(S) \otimes \delta_{\text{SR}}$
    for each cumulant $C_i$ do
        $\delta_{C_i} = C_i - \phi(S)^\top \mathbf{w}_i$
        $\mathbf{w}_i \leftarrow \mathbf{w}_i + \alpha_{C_i}\phi(S)\delta_{C_i}$
    end for
end while
$\delta_{\text{SR}} = \phi(S') - M^\top \phi(S')$
$M \leftarrow M + \alpha_{\text{SR}}\phi(S') \otimes \delta_{\text{SR}}$
```

where ∇_M is the gradient with respect to M and $\otimes$ is the outer product. Based on this derivation, as well as Eq. 4 and 5, we obtain Algorithm 1. Note that the last two lines of Algorithm 1, which update the SR for the state S', are only required for the episodic case.

This algorithm allows us to predict the cumulant C_i, in state S, using the current estimate of the matrix M and the weights $\mathbf{w}_i$. We can then obtain the final prediction by simply computing $\psi\big(\phi(S)\big)^\top \mathbf{w} = \big(M^\top \boldsymbol{\phi}(s)\big)^\top \mathbf{w} = \boldsymbol{\phi}(s)^\top M\mathbf{w}$.

This algorithm accelerates learning because, generally, learning to estimate $\bar{c}$ is faster than learning the GVF directly. This is exactly what our algorithm does. When predicting a new signal, it starts with its current estimate for the SR, Ψ. At the end, the multiplication $\Psi\bar{c}$ is simply a weighted average of the one-step predictions across all states, weighted by the likelihood they will be visited. We provide empirical evidence supporting this claim in the next sections.

IV. Evaluation in Dayan's Grid World

We first evaluated our algorithm in a tabular grid world. Its simplicity allowed us to analyze our method more thoroughly since we were not bounded by the speed and complexity of physical robots. The grid world we used was inspired by Dayan's [9] (see Figure 1a). Four actions are available in this environment: up, down, left, right. Taking an action into a wall (blue) results in no change in position. Transitions are deterministic, i.e., a move in any direction moves the agent to the next cell in the given direction, except when moving into a wall. For each episode the agent spawns at location **S** and the episode terminates when the agent reaches the goal **G**.

We generated fifty different signals for the agent to predict. They were generated randomly from a collection of primitives enumerated in Table I. They are composed of two different primitives, one for each axis, like so: $(sig_x(x + offset_x) + bias_x) * (sig_y(y + offset_y) + bias_y)$.

TABLE I: Signal Primitives

Primitive	Parameters
Fixed value	value $\in [-2.0, 2.0)$
Square wave	period $\in [2, 40)$, invert $\in [True, False]$
Sin wave	period $\in [2, 40)$
Random binary	Fixed random binary string generated over the length of an axis.
Random float	Fixed random floats $\in [0, 1.0)$ generated over the length of an axis.
Unit	Fixed value of 1.0
Shortest path	transition_cost $\in [-10.0, -1.0)$, goal_reward $\in [1.0, 10.0)$

The bias and offset were drawn from $[-2.0, 2.0)$ and $[0, 10)$, respectively. Offset and bias were not applied to either the unit or shortest path primitives. Further, the shortest path primitive was not combined with a second signal, but was used on its own. Gaussian noise with a standard deviation of 0.3 was applied on top of each signal. The shortest path signal is inspired by a common reward function used in RL where each transition has a cost (a negative reward) meant to push the agent to completing a task in a timely manner and reaching the goal produces a positive reward signal.

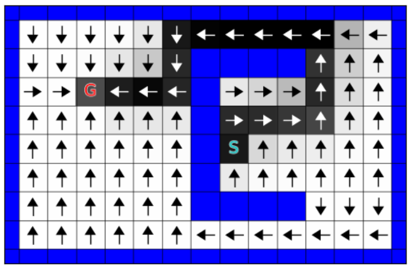
Our agent selects actions using ϵ-greedy action selection where, at each timestep, with probability $1 - \epsilon$, it uses the action specified by a hand-coded policy (see Figure 1a), and otherwise chooses randomly from all four actions ($\epsilon = 0.3$ in our experiments). A tabular representation is used with each grid cell uniquely represented by a one-hot encoding.

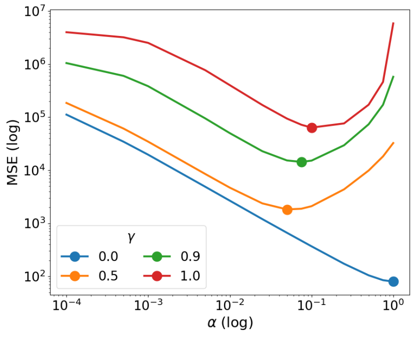
In this set of experiments we can compute the ground-truth predictors for the SR and the signal predictors. This is done by taking the average return observed from each state. The SR reference was averaged over 30,000 episodes and the signal predictor references were averaged over 10,000 episodes. Each episode started at the start state and followed the ϵ-greedy policy already described.

We first evaluated the predictive performance of our SR learning algorithm with respect to the step-size for a variety of γ values. We report the average over 30 trials of 10,000 episodes. We initialize the SR weights to 0.0. The squared Euclidean distance was calculated between the predicted SR and the reference SR for each timestep. These values were summed over the run and the average was taken across the runs. These averages are shown in Figure 1b.

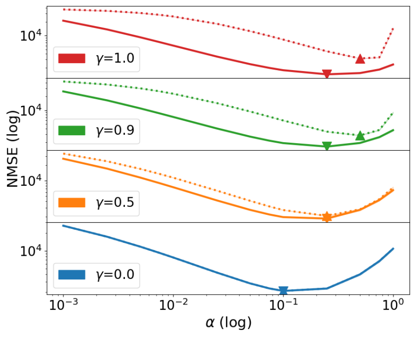
Using the results in Figure 1b we evaluated the performance of the two signal prediction approaches by sweeping across step-sizes for various γ. For each experimental run, learning of a new signal was enabled incrementally every 50 episodes. This produced runs with a total length of 2,500 episodes where the first pair of GVFs added (the direct and one-step predictors) were trained for 2,500 episodes and the last added GVFs were trained for 50 episodes. Further, for each run, the order in which the signals were added was randomized. Thirty runs were performed. The weights of the predictors and of the SR were initialized to 0.0. Notice that the SR was being learned at the same time as the direct and one-step predictions.

For each run a cumulative MSE for each signal i was calculated according to Eq. 6. This equation computes the


(a)



(b)



(c)

Fig. 1: **a)** Dayan's grid world. Arrows indicate the hand-coded policy (from the start state the policy is to go up). Black squares indicate the SR prediction given from the starting state, S, for $\gamma = 1.0$; the darker the square the higher the expected visitation. Notice the graying around the central path caused by the ϵ-greedy action selection. **b)** MSE of the SR as a function of step-size α for different values of γ. Lowest error is indicated by the markers. **c)** A comparison of the NMSE of the direct (dashed lines) and SR-based (solid lines) predictions as a function of fixed step-size α and discount factor γ summed across all signals. Lowest cumulative error is indicated by up arrows (direct predictions) and down arrows (SR-based predictions). Note, that although difficult to see, confidence intervals of 95% are included in both b and c.

total squared error between the predictor's estimate, V and the reference predictor's estimate, V^*. For each episode, E, the error of the current and previous episodes is averaged. Then, for each signal, the maximum error for a given γ, either in the direct or SR-based predictions, is found and used to normalize the errors in the signal across that particular value of γ (see Eq. 7). In this way we attempt to treat the error of each signal equally. If this is not done the errors of large magnitude signals dominate the results. These normalized values are then summed across the signals and the averages across all 30 runs are plotted in Figure 1c.

$$MSE_i = \frac{1}{E}\sum_{e_0}^{E}\sum_{t}^{T}(V_{i:t} - V^*_{i:t})^2 \tag{6}$$

$$NMSE_{i:\alpha:\gamma} = \frac{MSE_{i:\alpha:\gamma}}{\max_\alpha(MSE_{i:\gamma:direct}, MSE_{i:\gamma:SR})} \tag{7}$$

The advantage of the SR-based method is clear as γ increases. This is to be expected since for $\gamma = 0.0$ both methods are making one-step predictions. In the experiment of Figure 1c, the SR performs better in the vast majority of the signals, as shown in Table II. For all step-sizes not listed the SR-based method was better on all signals. Analysis of these cases where the direct method did better reveal that some of the target signals have very small magnitudes, suggesting the SR-based approach may be more susceptible to signal-to-noise ratio. Further analysis remains to be done.

Finally, we analyzed how the prediction error of our systems evolve with time. This is demonstrated in Figure 2 where we selected the best step-sizes for $\gamma = 0.9$ and

TABLE II: Signal performance for $\gamma = 0.9$ of Figure 1c.

α	**Direct Better**	**SR-Based Better**
0.25	3	47
0.5	5	45
0.75	4	46
1.0	1	49

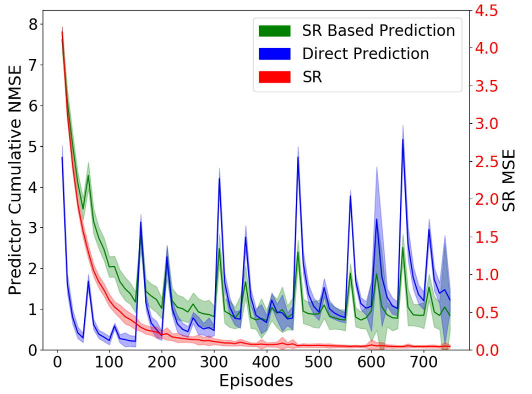


Fig. 2: All predictors learn from scratch with new predictors added in every 50 episodes. As the SR error (red, right axis) goes low the SR-based predictors (green) are able to learn faster than their direct (blue) counterparts. Shading indicates a 95% confidence interval.

plotted the performance over time across 30 different runs. In this case the order of the signals remained fixed so that sensible averages could be plotted for each signal. Signal performance was normalized as before and summed across all active signals. As expected, we clearly see that the SR-based predictions (green) start with much higher error than the direct (blue), but as the error of the SR (red) drops low the newly added SR-based predictors are able to learn quicker, with less peak and overall error than the direct predictor.

In the continual learning setting we never have the opportunity to tune for optimal step-sizes as we did in our evaluation. Practically, fixed step-sizes are used for many robotics settings in RL, but, in order to ensure stable learning, small step-sizes are chosen. As we saw in Figure 1c, the advantage of using the SR-based predictions is enhanced with smaller step-sizes. Ideally, however, we would imagine that a fully developed system would use some method of adapting step-sizes, such as ADADELTA [11].

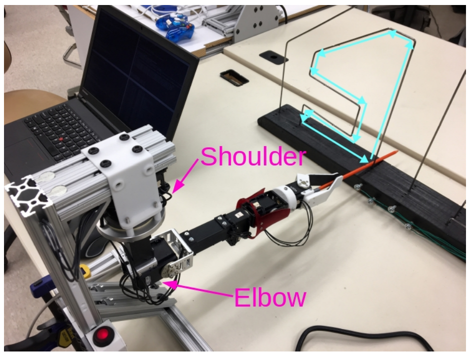


Fig. 3: The user controls the robot arm using a joystick to trace the inside of the wire maze in a counter-clockwise direction. Circuit path shown in blue.

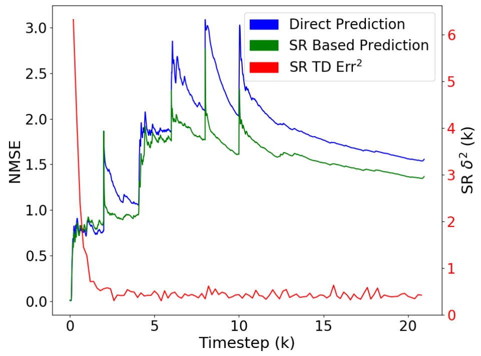


Fig. 4: A 12 minute run tracing the maze circuit. A new predictor is added every 2,000 timesteps. NMSE errors are summed across all predictors.

V. Evaluation on a Robot Arm

Tabular settings like Dayan's grid world are useful for enabling analysis and providing insight into the behavior of our method. However, our goal is to accelerate learning on a real robot where states are not fully observed and cannot be represented exactly; instead we must use function approximation. Here we demonstrate our approach using a robot arm and learning sensorimotor predictions with respect to a human-generated policy. In our task a user controls a robot arm via joystick to trace a counter-clockwise circuit through the inside of the wire maze (see Figure 3) with a rod held in the robot's gripper. The user performed this task for approximately 12 minutes completing around 50 circuits.

In this experiment we used six different prediction targets: the current, position and speed of both the shoulder rotation and elbow flexion joints. A new predictor was activated every 2,000 timesteps (Note that the robot reports sensor updates at 30 timesteps/s). For this demonstration a discount factor of $\gamma = 0.95$ was used. Four signals were used as input to our function approximator: the current position and a decaying trace of the position for the shoulder and elbow joints. The decaying trace for joint j was calculated as $tr_{j:t+1} = 0.8 * tr_{j:t} + 0.2 * pos_{j:t+1}$. These inputs were normalized over the joint ranges observed in the experiment and passed into a 4-dimensional tilecoding [6] with 100 tilings of width 1.0 and a total memory size of 2048. Additionally a bias unit was added, resulting in a binary feature vector of length 2049 with a maximum of 101 active features on each timestep (hashing collisions can reduce this number). We use a decaying step-size for all the predictors where the step-size starts at 0.1 and decays linearly to zero over the entire dataset. At each timestep this step-size is further divided by the number of active features in $\phi(S_t)$. Finally, for each predictor, this step-size is offset such that the step-size starts at 0.1 when it is first activated and decays at the same rate as all the other predictors.

To compare the prediction error we compute a running MSE for each signal according to Eq. 8, where at each timestep t the sum is taken over all previous timesteps. Unlike the previous tabular domain, we do not have the ideal estimator to compare against and instead compare the predictions, V, against the actual return, G. In order to treat each signal equally we further normalize these errors according to Eq. 9. Note that the NMSE allows us to compare the predictions of a single signal between the two methods, but does not tell us how accurate the predictions are nor does it allow comparison between signals.

$$MSE_{i:t} = \frac{1}{T}\sum_{t=t_0}^{T}(V_{i:t} - G_{i:t})^2 \tag{8}$$

$$NMSE_i = \frac{MSE_i}{\max(MSE_{i:direct}, MSE_{i:SR})} \tag{9}$$

Figures 4 and 5 show a single run, approximately 12 minutes in length. A single ordering of the predictors was used. Figure 4 shows the error across all predictors while Figure 5 separates out each predictor. Here we see a clear advantage to using the SR-based predictions for most of the signals. Unlike the previous tabular results, there is little difference on the performance of the first predictor (shoulder current), even while the SR is being learned. To investigate, we ran experiments where each signal was learned from the beginning of the run. We observed that performance was rarely worse and sometimes even better when using the SR-based method. This suggests the SR-based approach is more robust than expected, but further experimentation is needed.

VI. Further Advantages When Scaling

While this paper analyzed single policies and discount functions, this is not the setting in which the GVF framework is proposed to be used. Rather, it is imagined that massive numbers of GVFS over many policies and timescales will be used represent complex models of the world [4], [12]. In this setting we note that using SR-based predictions can offer additional benefits, allowing the robot to do more with less. Consider, for a single policy π, a collection of SRs learned for f discount functions and h one-step predictors. We can then represent fh predictions using $f + h$ predictors. A first advantage is that far fewer GVFs need to be updated on each timestep, saving computational costs. As a second benefit, there is potential to reduce the number of weights used by the system. For example, consider learning in a tabular setting, with $|\mathcal{S}|$ states, using linear estimators. For fh predictions

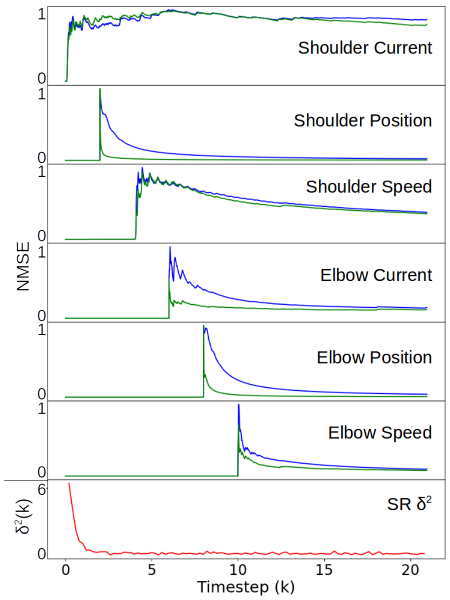


Fig. 5: The same results as Figure 4, but with the NMSE for the individual predictors. Each is normalized from 0 to 1.

the number of weights needed is : $|w|_{direct} = fh|\mathcal{S}|$, $|w|_{SR-based} = f|\mathcal{S}|^2 + h|\mathcal{S}|$. It can be shown that for a fixed f and $\mathcal{S}$ the total number of weights used by the direct prediction approach is greater when $h > \frac{f|\mathcal{S}|}{f-1}$.

VII. Related Work

The SR was originally introduced as a representation method for policy learning [9], but has recently been applied to other settings. It has been used, for instance, in transfer learning problems allowing agents to generalize better across different tasks [10], [13]–[18]. Particularly, the work of Zhang et al. [16] showed how the SR could be used for transfer in robot navigation tasks. The robot first learned to navigate a simulated 3D maze environment. Next, the robot was moved to a real maze and showed transfer by learning the real-world navigation task faster than learning from scratch. Zhu et al. [17] demonstrated the use of the SR in transfer with a traditional STRIPS style planning algorithm. In their setting a simulated robot planned over high-level actions with camera images as input. Their approach showed better ability to complete previously unseen tasks than several other well-known algorithms.

To learn optimal policies an RL agent must sufficiently explore its state space. In complex, real-world scenarios, such as those faced by robots, the state-action space is incredibly large and exploration by random primitive actions, as is often done in simple grid-world domains, is not feasible. The work of Machado et al. [19] uses the SR to define intrinsic rewards in an option discovery algorithm. By exploring with these options agents are able to improve the efficiency of their exploration. Such an approach might be applied beneficially to real-world robots.

GVFs were originally proposed as a method for building an agent's overall knowledge in a modular and hierarchical way, representing state as a collection of predictions [4]. To date they have been used with hand-coded fixed policies [12], [20] and as state for learned policies [21]. GVFs have also been used to track internal statistics such as the variance of the return [22] and state visitation [23]. The UNREAL agent [24] is a powerful demonstration of the usefulness of multiple predictions; pre-defined auxiliary tasks, which can be viewed as GVFs, are shown to accelerate and improve the robustness of learning. Work on the construction of GVF models is in its infancy, with the primary approach explored being random generation [25]. However, there are related approaches using other predictive frameworks including TD-networks [26] and PSRs [27].

There are several ways in which our own results might be improved and extended. Alternative algorithms have been suggested for learning with the SR [28], [29]. In particular, Pitis [29] shows potential alternatives to: 1) accelerate the learning of value estimates by using the SR and 2) make the one-step signal predictors more robust to noise. These algorithms may further improve the performance of the SR-based prediction approach. Finally, a limitation of the SR is that it is tied to a specific policy. This limitation was relaxed by combining the concept of Universal Value Functions (UVFAs) [30] with the SR in the Universal Successor Representation (USR) [18]. The USR enables SR-based predictions to generalize across goals (and their corresponding policies). Combining our approach with the USR could further improve learning efficiency and generality.

VIII. Conclusions

In this paper we showed how the successor representation (SR), although originally introduced for another purpose, can be used to accelerate learning in a continual learning setting in which a robot incrementally constructs models of its world as a collection of predictions known as general value functions (GVFs). The SR enables a given prediction to be modularized into two components, one representing the dynamics of the environment (the SR) and the other representing the target signal (one-step signal prediction). This allows a robot to reuse its existing knowledge when adding a new prediction target, speeding up learning of the new predictor. We demonstrated this behaviour in both a tabular grid world and on a robot arm. These results suggest an effective method for improving the learning rate and sample efficiency for robots learning in the real world. There are several clear opportunities for further research on this topic. The first is to provide greater understanding into why, for a given fixed step-size, some (few) signals are better predicted directly rather than through the SR. Further, the work in using the SR with function approximation is preliminary and more insight can yet be gained in this setting. Another opportunity for research is to explore using SR-based predictions with state-dependent discount functions. Finally, we suggest that SR-based predictions with deep feature learning [13], [19] and an incrementally constructed architecture would be a very powerful tool to support continual or developmental learning in robotic domains with widespread real-world applications.

References

[1] G. L. Drescher, *Made-up Minds: A Constructivist Approach to Artificial Intelligence*. MIT Press, 1991.

[2] M. B. Ring, "Continual Learning in Reinforcement Environments," Ph.D. dissertation, The University of Texas at Austin, 1994.

[3] P. Oudeyer, F. Kaplan, and V. V. Hafner, "Intrinsic Motivation Systems for Autonomous Mental Development," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 2, pp. 265–286, 2007.

[4] R. S. Sutton, J. Modayil, M. Delp, T. Degris, P. M. Pilarski, A. White, and D. Precup, "Horde: a Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction," in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011, pp. 761–768.

[5] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level Control through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[8] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the Game of Go Without Human Knowledge," *Nature*, vol. 550, pp. 354–359, 2017.

[9] P. Dayan, "Improving Generalization for Temporal Difference Learning: The Successor Representation," *Neural Computation*, vol. 5, no. 4, pp. 613–624, 1993.

[10] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Žídek, and R. Munos, "Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement," in *International Conference on Machine Learning (ICML)*, 2018.

[11] M. D. Zeiler, "ADADELTA: An Adaptive Learning Rate Method," *arXiv*, vol. 1212.5701, 2012.

[12] J. Modayil, A. White, and R. S. Sutton, "Multi-Timescale Nexting in a Reinforcement Learning Robot," *Adaptive Behavior*, vol. 22, no. 2, pp. 146–160, 2014.

[13] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep Successor Reinforcement Learning," *arXiv*, vol. 1606.02396, 2016.

[14] L. Lehnert, S. Tellex, and M. L. Littman, "Advantages and Limitations of Using Successor Features for Transfer in Reinforcement Learning," *arXiv*, vol. 1708.00102, 2017.

[15] H. Yao, C. Szepesvári, R. Sutton, J. Modayil, and S. Bhatnagar, "Universal Option Models," in *Neural Information Processing Systems (NIPS)*, 2014.

[16] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep Reinforcement Learning with Successor Features for Navigation across Similar Environments," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2371–2378.

[17] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual Semantic Planning using Deep Successor Representations," in *International Conference on Computer Vision (ICCV)*, vol. 2, no. 4, 2017, p. 7.

[18] C. Ma, J. Wen, and Y. Bengio, "Universal Successor Representations for Transfer Reinforcement Learning," *arXiv*, vol. 1804.03758, 2018.

[19] M. C. Machado, C. Rosenbaum, X. Guo, M. Liu, G. Tesauro, and M. Campbell, "Eigenoption Discovery through the Deep Successor Representation," in *International Conference on Learning Representations (ICLR)*, 2018.

[20] C. Sherstan, J. Modayil, and P. M. Pilarski, "A Collaborative Approach to the Simultaneous Multi-joint Control of a Prosthetic Arm," in *International Conference on Rehabilitation Robotics (ICORR)*, 2015, pp. 13–18.

[21] P. M. Pilarski, T. B. Dick, and R. S. Sutton, "Real-Time Prediction Learning for the Simultaneous Actuation of Multiple Prosthetic Joints," in *International Conference on Rehabilitation Robotics (ICORR)*, 2013, pp. 1–8.

[22] C. Sherstan, B. Bennett, K. Young, D. R. Ashley, and R. S. Sutton, "Directly Estimating the Variance of the λ -Return Using Temporal-Difference Methods," in *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

[23] C. Sherstan, M. C. Machado, A. White, and P. M. Pilarski, "Introspective Agents: Confidence Measures for General Value Functions," in *International Conference on Artificial General Intelligence (AGI)*, 2016, pp. 258–261.

[24] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu, "Reinforcement Learning with Unsupervised Auxiliary Tasks," in *International Conference on Learning Representations (ICLR)*, 2017.

[25] M. Schlegel, A. White, A. Patterson, and M. White, "General Value Function Networks," *arXiv*, vol. 1807.06763, 2018.

[26] T. Makino and T. Takagi, "On-line Discovery of Temporal-Difference Networks," in *International Conference on Machine Learning (ICML)*, 2008.

[27] P. McCracken and M. Bowling, "Online Discovery and Learning of Predictive State Representations," in *Neural Information Processing Systems (NIPS)*, 2006, pp. 875–882.

[28] C. A. Gehring, "Approximate Linear Successor Representation," in *Reinforcement Learning Decision Making (RLDM)*, 2015.

[29] S. Pitis, "Source Traces for Temporal Difference Learning," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[30] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal Value Function Approximators," in *International Conference on Machine Learning (ICML)*, 2015, pp. 1312–1320.