Lambda Successor Return Error

Anthony GX-Chen
Mila - Quebec AI Institute
McGill University

Veronica Chelu
Mila - Quebec AI Institute
McGill University

Blake Richards
Mila - Quebec AI Institute
McGill University

Joelle Pineau
Mila - Quebec AI Institute
McGill University
Facebook AI Research

Abstract

We introduce the λ *Successor Return Error* (λ-SRE), by factorizing λ-return error into one-step temporal difference errors, and a successor-like representation (SR). This return can be computed independent of the parameterization of the value function. We introduce the λ-SRE algorithm and demonstrate a proof of concept in the tabular random chain environment, showing its advantageous performance to both the offline λ-return, and the SR algorithm (with factorized reward representations) for policy evaluation. We conclude by discussing this perspective in the context of the recent neuroscience hypothesis of the brain using successor-like representations.

1 Introduction

In reinforcement learning (RL), the successor representation (SR) is a way of representing each state as the total future visitations to other states [1] under a given policy. Once learned, the SR can be combined with any instantaneous reward function to compute the value function in "one shot", assuming stationary environment dynamics. In neuroscience, many works have discussed the use of SR, particularly in relation to *place fields* [2] in the *hippocampus* [3, 4, 5, 6]. Specifically, physiological studies show place fields as skewed in a policy and transition-dependent way [5, 7, 8, 9], and human behavioural data appears to be well-fitted by a combination of SR and model-based algorithms [3]. Such a "predictive representation" may also shed light on canonical psychology results, such as Tolman's *latent learning* [10]. While the neuro-SR literature have theorized a number of potential use-cases (see: [5, 11, 12]), the predominant theory is that SR allows for one-shot computation of the value function [1], allowing for a "middle-ground" between model-free efficiency and model-based flexibility [4].

While having a well-learned SR can be used for transfer and discovering structures in the environment dynamics [13, 14, 15], it is less clear that SR is useful in the case of tabula rasa *single-policy evaluation*. As [16] speculates and [17] empirically demonstrates, decoupling the value function into SR and reward can result in value estimates that are quite poor until the learning is complete. Furthermore, the ability for SR to transfer well may be limited to cases where the optimal policy is similar [17].

One family of methods for efficient tabula rasa policy evaluation is the λ-return [18]. This can be viewed through the lens of *eligibility traces* (via the "backward view", or TD(λ)), which keep track of previously activated parameters to quickly update them with new rewards. Alternatively, it can be viewed through the lens of the λ-return (the "forward view"), by constructing an exponentially weighted average over all future (n-step) value predictions. TD(λ) allows for online updates of value

Preprint. Accepted to NeurIPS 2020, Workshop on Biological and Artificial Reinforcement Learning.

functions, while the λ-return is a "theoretical" view that requires the full length of a future trajectory, though it is more easily adopted to episodic settings and settings that do training over trajectories (i.e. recurrent neural net training) [19].

We focus on the single-policy evaluation case and propose a method to learns SR-*like* representations *for single policy evaluation*. We show specifically that the forward view λ-return error, in expectation, can be decomposed into expected (one-step) temporal difference errors, and a SR-like quantity denoting discounted future occupancy. We refer to the SR-weighted return as the λ *Successor Return*. Our contributions are as follows: (i) we show an interesting connection between λ-return and SR-like representations in the form of λ successor return for temporal credit assignment; (ii) we translate these insights into an algorithm and empirically illustrate that it outperforms λ-return and the SR in a tabular toy domain; and (iii) we interpret the proposed mechanism in the neuroscience context.

2 Background

We use the standard RL setup, formalized using a Markov Decision Process (MDP) defined by states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, transition probabilities $\mathcal{P} = P(S_{t+1}|S_t, A_t)$, reward $\mathcal{R}$ and a discount factor $0 \leq \gamma \leq 1$. $\pi : \mathcal{S} \to \mathcal{A}$ is the policy, and $P_\pi(S'|S) = \mathcal{P}(S'|S, \pi(S))$ is the state transition probabilities under policy π. We want to learn the value function v_π, which we approximate: $v_\theta(s_t) \approx v_\pi(s_t) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s_t]$. Note we sometimes use $v_{\theta,t} = v_\theta(s_t)$ for brevity.

We learn v_θ by constructing *returns* as update targets. The n-step return at time t is defined as

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^{n-1} R_{t+n} + \gamma^n v_\theta(S_{t+n}). \tag{1}$$

The lambda (λ-) return is an exponentially weighted average over all n-step returns.

$$G_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}. \tag{2}$$

We minimize the mean squared loss (MSE) loss between the λ-return and v_θ: $\mathcal{L}_t = \frac{1}{2}[G_t^\lambda - v_\theta(s_t)]^2$. We have the update equation:

$$\theta_{n+1} = \theta_n + \alpha[G_t^\lambda - v_{\theta,t}] \frac{\partial v_{\theta,t}}{\partial \theta_n}, \tag{3}$$

where α is the learning rate, or step size. We note we can write the λ-*error* term in Equation 3 as:

$$G_t^\lambda - v_{\theta,t} = \sum_{k=t}^{\infty} (\lambda\gamma)^{k-t} \delta_k, \tag{4}$$

where $\delta_k = R_{k+1} + \gamma v_\theta(s_{k+1}) - v_\theta(s_k)$ is the TD(0) error at timesteps k, $k \geq t$. The equality above is exact in the case of offline (i.e. end-of-episode) updates, and becomes approximate if updates to θ are made online. This equivalency can be demonstrated by expanding the lambda return and re-arranging the order of summations.

3 The Lambda Successor Return

Let us first define the lambda successor representation (λ-SR):

$$m_\pi(s, s') = \sum_{n=0}^{\infty} (\lambda\gamma)^n P_\pi(S_{t+n} = s'|S_t = s), \tag{5}$$

where $P_\pi(S_{t+n} = s'|S_t = s)$ is the probability of arriving in state s' after n steps when starting from state s and following policy π. We note λ-SR is equivalent to the successor representation (SR) when $\lambda = 1$ [1]:

$$M(s, s') = (I - \gamma P_\pi)^{-1}(s, s') = \sum_{n=0}^{\infty} \gamma^n P_\pi(S_n = s'|S_0 = s) \tag{6}$$

3.1 Relationship to the lambda-return

We write the λ-return error of Eq.4 in expectation:

$$\mathbb{E}[G_t^\lambda - v_\theta(s)|S_t = s] = \mathbb{E}\left[\sum_{k=0}^{\infty}(\lambda\gamma)^k\ \delta_\theta(S_{t+k})\middle|S_t = s\right], \tag{7}$$

$$= \sum_{k=0}^{\infty}\sum_{s'}(\lambda\gamma)^k P_\pi(S_{t+k} = s'|S_t = s)\ \mathbb{E}[\delta_\theta(s')], \tag{8}$$

$$= \sum_{s'}\mathbb{E}[\delta_\theta(s')]\sum_{k=0}^{\infty}(\lambda\gamma)^k P_\pi(S_{t+k} = s'|S_t = s), \tag{9}$$

$$= \sum_{s'} m_\pi(s, s')\ \mathbb{E}[\delta_\theta(s')], \qquad \text{[definition of } m_\pi\text{]} \tag{10}$$

$$= \mathbb{E}_Q\left[\frac{m(S, S')}{Q(S')}\ \mathbb{E}[\delta_\theta(S')]\middle|S = s\right], \tag{11}$$

where $\mathbb{E}[\delta_\theta(s')] = \mathbb{E}[\delta_\theta(S_{t+k})|S_{t+k} = s'] = \sum_{s''} P_\pi(s''|s')[r(s'') + \gamma v_\theta(s'')] - v_\theta(s')$ is the expected one-step TD error from state s' following policy π, which works via the Markov property and can be estimated as long as we have on-policy transitions (note we sub-scripted with θ to make clear that the expected one-step TD error will depend on the model parameters). With reference to Eq.10, the error from λ-return in expectation can be factorized into the λ-SR (m_π), and the expected error at each state of the MDP.

We refer to Eq.10 as the *lambda successor return error* (λ-SRE). We emphasize it is a quantity independent of the value function parameterization, much like the λ-return is. That is, Eq.10 can be used with any kind of parameterization of the value function. While the summation in Eq.10 is typically intractable, equation 11 suggests a family of algorithms, as long as they define a sampling distribution, Q, over the state space to allow for an unbiased estimate of the λ-SRE. For example, in the case of uniform sampling of s', the error at state s will be directly proportional to $m(s, s')\delta_\theta(s')$.

3.2 The offline lambda-SRE algorithm

As a proof of concept and for a fair comparison with the offline λ-return algorithm, we propose an offline algorithm using states sampled only from a single episode of on-policy trajectory. We note that λ-SRE does not need to be an offline algorithm, but our main hypothesis is that simply *having* the λ-SRE (regardless of its implementation detail) *is better than using λ-return*, specifically by removing the trajectory dependence of λ-return via factoring out the entire time component into the λ-SR [1], which we hypothesize reduces trajectory variance (at the potential cost of increased bias in using m_π). Specifically, we use the on-policy distribution as Q to approximate Eq.11 without correcting for the frequency of states encountered. This is a naive approach, albeit we found that it worked reasonably well in the restricted set of empirical studies we performed. We leave the development of more principled methods of correcting the return for future work. This gives rise to our offline λ-SRE algorithm (Algorithm 1).

Note algorithm 1 is independent of the value function parameterization θ as long as we can compute $\partial v_\theta(s_t)/\partial\theta$ (e.g. via backpropagation in a neural network), and assuming that we can learn (or are given) a λ-SR function m_π (Alg. 1, line 5).

3.3 Learning the λ-SR

We now examine the question of learning the λ-SR. Similar to the successor representation, λ-SR has a Bellman form:

$$m_\pi(s_t, s') = P_\pi(S_0 = s'|S_0 = s) + (\lambda\gamma)\mathbb{E}_\pi\left[m_\pi(s_{t+1}, s')\right]. \tag{12}$$

Therefore in theory we can learn it using any RL methods [1, 16]. However, we note that learning m_π in the functional approximation (non-tabular) case is non-trivial. We discuss this further in the future works section below. For now, we use a tabular algorithm for learning m_π for line 5 of Algo.1. We note how to best do SR learning is not the focus of this work (but rather that it is beneficial to learn

Algorithm 1: Offline λ-SRE algorithm using single episode samples

Given: value function v_θ with parameter θ;
Given: the λ-SR function, $m_\pi(\cdot,\cdot) \to \mathbb{R}$ and a way to learn it;
while *repeat episodes forever* **do**
  Collect end-of-episode on-policy trajectory $\tau_\pi = (s_0, a_0, r_1, s_1, ..., r_T)$;
  Update λ-SR function, m_π ;
  for $t \leftarrow 0, ..., T-1$ **do**
    $\delta_t = r_{t+1} + \gamma v_\theta(s_{t+1}) - v_\theta(s_t)$
  end
  for $t \leftarrow 0, ..., T-1$ **do**
    $\Delta\theta = \left[\frac{1}{T-t}\sum_{k=t}^{T-1} m_\pi(s_t, s_k)\delta_k\right] \frac{\partial v_\theta(s_t)}{\partial\theta}$;
    $\theta \leftarrow \theta + \alpha\,\Delta\theta$;
  end
end

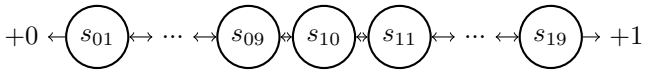

Figure 1: The 19-state random chain. Agent start in the centre and transition randomly to adjacent states until reaching the end to receive the indicated reward

this quantity), so we trivially adopt the tabular update algorithm from [14], providing it in Appendix A.1 for completeness.

4 Experiments

For our experiments, we focus on the tabular setting as a proof-of-concept for λ-SRE, as it allows for tractable computation of m_π and for us to easily solve for the true values of the MDP to compare against the agents' estimates. We evaluate the algorithm on the canonical 19 states random chain task (Fig.1, also see [18], Figure 12.3). We average over 50 seeds for all experiments.

4.1 Policy Evaluation

We compare all algorithms in the offline, on-policy setting with access to the full trajectory of the most recent episode. We compare three algorithms learning from scratch: (i) offline lambda return, (ii) our offline λ-SRE, and (iii) a SR algorithm with decoupled SR and reward representations. We give the SR algorithm the *true* reward function such that it only needs to learn the SR matrix. Despite this, the SR algorithm had a large (> 1) error. We therefore excluded it from figure 2.

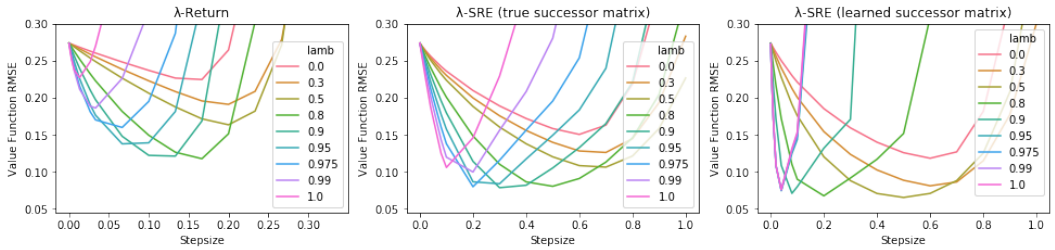

Figure 2: Value root mean squared error (RMSE) at the end of the 10-th episode. Left: offline λ-return. Centre: offline λ-SRE (given true m_π), Right: offline λ-SRE (concurrently learned m_π).

We see the λ-SRE agent performs similarly whether the λ-SR matrix is given or learned from scratch (Fig.2, centre and right), albeit the concurrently learned λ-SR agent shows more instability across the learning rates. Both λ-SRE algorithms consistently out-perform the traditional λ-return algorithm

(Fig.2, left) while having access to the exactly same information. Figure 3 further demonstrates this by plotting the 100 episodes learning curves for the three algorithms with best parameter settings.

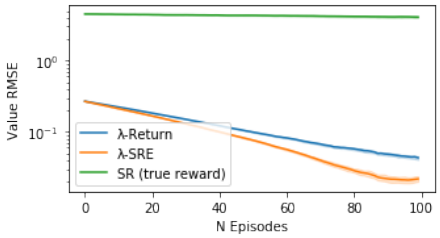

Figure 3: Value root mean squared error (RMSE), 100 episodes of training (log scale)

4.2 Successor-like representations

Having shown the benefit of using λ-SRE, we turn to the representation learned. We compare the learning between the λ-SR and the SR (given the ground truth reward function) directly by setting $\lambda = 1$ for the λ-SRE agent such that both agents are learning identical SR matrices.

Figure 4: Root mean square error (RMSE) of the value function and successor matrix. Left: value function error over training. Middle: successor matrix error over training. Right: correlation of value function error with successor matrix error.

We observe that while the successor matrix error is identical for both algorithms throughout training (Fig.4, middle), the λ-SRE agent converges much more quickly than the SR agent (Fig.4, left). We also see that the λ-SRE agent is robust to a much higher successor matrix error while the SR agent requires a highly accurate successor matrix for good value prediction (Fig.4, right).

5 Related Work

The closest work to ours is Source Traces [20], which is a "backward view" version of λ-SRE. While we update a current value function by a future error, $\Delta v(s_t) = \sum_{s'} m(s_t, s')\delta_\theta(s')$ (tabular update), [20] updates *past* value functions using the present TD error: $\Delta v(s') = m(s', s_t)\delta(s_t)\ \forall s'$. The trace perspective is explored further in [21]. More generally, the SR is a versatile representation that was proposed for state generalization [1], has been applied to transfer [13, 22], exploration [14], and more generally for uncovering structures in the environment [23, 15].

6 Discussion

6.1 The λ successor return error

Consider the tabular successor representation matrix $\mathbf{M} \in \mathbb{R}^{|S|\times|S|}$. Each row, $\mathbf{M}_{i,}$, is the SR for state i, representing the total future (discounted) visitations to all other states [1]. We can similarly view the SR as the degree of "contribution" from state i to all other states. Consequently, the expected *error* received at state i from other states is proportional to its successor representation. Interestingly, we can also view the matrix *columns*, $\mathbf{M}_{,j}$, as how error at state j should be distributed to all other states - this is known as the *ideal source trace* and explored extensively in [20].

In λ-return, the λ term gives us additional flexibility to tune how much importance we wish to assign to future *errors*, which is expressed via the λ-SR matrix, $\mathbf{m}$. Similar to how the value is the expected discounted sum of future (instantaneous) rewards, the λ-return error is the discounted sum of future (one-step) errors (Eq.4). It is perhaps unsurprising that just as one can compute the value for a state in one step using the SR and rewards, we can compute the expected λ-return error in one step using the λ-SR and TD errors. We can interpret the λ-successor return similarly to the λ-return. In the case of $\lambda = 0$, we have the lambda successor matrix $\mathbf{m} = \mathbf{I}$, meaning we only care about the *immediate* TD errors and do TD(0). In the case of $\lambda = 1$ we recover the successor representation ($\mathbf{m} = \mathbf{M}$), weighing future errors proportional to their actual (γ-discounted) occupancy.

We note that the γ term in SR is *behaviourally relevant*: different γ can result in different levels of myopia. Unlike γ dependent processes like SR learning, the λ term is a purely learning parameter, and we are free to tune the discounting freely in λ-SR without changing the policy myopia. It is interesting to observe that lower λ tends to perform better in the λ-SRE agent as compared to the λ-return agent (Fig.2), potentially pointing to lower λ resulting in a more stable learning problem.

6.2 Neuroscience

We propose a relationship between hippocampal *representation* [2, 5, 7, 8, 9], and the hippocampus as a system for fast learning (through replay [24, 25, 26], complementary learning [27, 28], and/or episodic control [29, 30, 31]). Through the λ-successor representation, we demonstrate a representation *to support fast learning* (of a value function) through better credit assignment. Notably, the physiological evidence supporting place fields as successor representations equally support the λ-SR hypothesis (as SR can be seen as a special case of λ-SR). We further speculate that learning the value function through λ-SRE naturally fit with methods such as experience replay (or a model), specifically through the need to sample the state space to estimate Eq.11. It may be interesting to re-interpret hippocampal replay data in light of λ-SRE.

We reiterate a small but important nuance between the traditional SR and λ-SR: SR is a direct (factorized) representation of the value function, while λ-SR weighs the TD errors to *update* a (separate) value function. We demonstrate through simulations that the latter can result in both faster learning and is more robust to imperfections in the learning process - a potentially desirable biological property that complements the strength of the traditional SR (e.g. for transfer). Subscribing to the λ-SR as a model for place fields, we would expect place cells to be important *during* learning. Some works indeed hint at this, such as the *transient* involvement of the dorsal hippocampus for action acquisition [32]; and dopamine release (a quantity typically associated with the one-step TD error [33]) in the dorsal hippocampus promoting spatial learning [34]. Future experiments can help delineate the interactions between reward prediction error, place fields, and learning speed, through the lens of λ-SRE.

6.3 Future work

We identify two main limitations to the current work: (i) proposing a sampling distribution, Q, in Eq.11; and (ii) having a principled method of learning the λ-SR function, m_π in high dimensional feature spaces. Point (i) is well suited to be combined with an experience replay buffer, and/or a (generative) model of states and state transitions. On the other hand, point (ii) is much more difficult, in particular to calcualte the $P_\pi(S_0 = s'|S_0 = s)$ term of Eq.12. While heuristic methods can be used to estimate the proximity of features, it is unclear currently how this can be done in a principled way. It may be possible to explore a factorized form of the λ-SRE to allow the use of successor features [17, 13], and in general this is a promising direction for future research, as the biological evidence also points to such "successor-like" representations being present in the brain, which works exclusively in high-dimensional, partially observable settings.

6.4 Conclusion

We showed an interesting connection between SR - a representation typically used for transfer, and λ-return - a quantity for temporal credit assignment. We show our method works in tabular settings and complements the current neuroscientific theories and observations, demonstrating its potential to be further investigated for future research at the intersection of biological and artificial RL.

Broader Impact

Our results are mainly theoretical, providing a connection between two theoretical concepts in reinforcement learning: the successor representation and the λ-return. We hope our work can improve current reinforcement learning algorithms, as well as theories for neuroscience that let us better understand the brain. We note we are contributing to advancement of reinforcement learning as a whole, and the potential creation of autonomous intelligence agents. To this larger scope, we are mindful that should this work become more practically applied to real world situations, we take into consideration the biases of real world data-sets and data collection processes. Furthermore, we are mindful to minimize the usage of our algorithms for poor intents, such as in military settings.

Acknowledgments and Disclosure of Funding

This work is generally supported by the NSERC master's scholarship, FRQNT master's research scholarship, and the UNIQUE Healthy Brains for Healthy Lives excellence scholarship. We are very grateful of various members of Mila for their insightful discussion on reinforcement learning, neuroscience, and general helpfulness: Annik Carson, Surya Penmetsa, Colleen Gillon, Arna Ghosh, Mandana Samiei, and Emmanuel Bengio. We would also like to thank Leo Cheong for his insights on the rodent experimental paradigm. Finally, we would like to thank the reviewers for their very constructive feedback.

References

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

[2] John O'Keefe and Jonathan Dostrovsky. The hippocampus as a spatial map: Preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 1971.

[3] Ida Momennejad, Evan M Russek, Jin H Cheong, Matthew M Botvinick, Nathaniel Douglass Daw, and Samuel J Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680–692, 2017.

[4] Samuel J Gershman. The successor representation: its computational logic and neural substrates. *Journal of Neuroscience*, 38(33):7193–7200, 2018.

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

[6] Ida Momennejad. Learning structures: Predictive representations, replay, and generalization. *Current Opinion in Behavioral Sciences*, 32:155–166, 2020.

[7] Mayank R Mehta, Michael C Quirk, and Matthew A Wilson. Experience-dependent asymmetric shape of hippocampal receptive fields. *Neuron*, 25(3):707–715, 2000.

[8] Alice Alvernhe, Etienne Save, and Bruno Poucet. Local remapping of place cell firing in the tolman detour task. *European Journal of Neuroscience*, 33(9):1696–1705, 2011.

[9] Stig A Hollup, Sturla Molden, James G Donnett, May-Britt Moser, and Edvard I Moser. Accumulation of hippocampal place fields at the goal location in an annular watermaze task. *Journal of Neuroscience*, 21(5):1635–1644, 2001.

[10] Edward Chace Tolman and Charles H Honzik. Introduction and removal of reward, and maze performance in rats. *University of California publications in psychology*, 1930.

[11] Samuel J Gershman. Predicting the past, remembering the future. *Current opinion in behavioral sciences*, 17:7–13, 2017.

[12] Ida Momennejad and Marc W Howard. Predicting the future with multi-scale successor representations. *BioRxiv*, page 449470, 2018.

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

[14] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. *arXiv preprint arXiv:1807.11622*, 2018.

[15] Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017.

[16] Lisa M White. *Temporal difference learning: eligibility traces and the successor representation for actions.* Citeseer, 1996.

[17] 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.

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

[19] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. *arXiv preprint arXiv:2003.13350*, 2020.

[20] Silviu Pitis. Source traces for temporal difference learning. *arXiv preprint arXiv:1902.02907*, 2019.

[21] Hado van Hasselt, Sephora Madjiheurem, Matteo Hessel, David Silver, André Barreto, and Diana Borsa. Expected eligibility traces. *arXiv preprint arXiv:2007.01839*, 2020.

[22] 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.

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

[24] Anoopum S Gupta, Matthijs AA van der Meer, David S Touretzky, and A David Redish. Hippocampal replay is not a simple function of experience. *Neuron*, 65(5):695–705, 2010.

[25] Marcelo G Mattar and Nathaniel D Daw. Prioritized memory access explains planning and hippocampal replay. *Nature neuroscience*, 21(11):1609–1617, 2018.

[26] Ida Momennejad, A Ross Otto, Nathaniel D Daw, and Kenneth A Norman. Offline replay supports planning in human reinforcement learning. *Elife*, 7:e32548, 2018.

[27] James L McClelland, Bruce L McNaughton, and Randall C O'Reilly. Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419, 1995.

[28] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in cognitive sciences*, 20(7):512–534, 2016.

[29] Máté Lengyel and Peter Dayan. Hippocampal contributions to control: the third way. In *Advances in neural information processing systems*, pages 889–896, 2008.

[30] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

[31] Samuel J Gershman and Nathaniel D Daw. Reinforcement learning and episodic memory in humans and animals: an integrative framework. *Annual review of psychology*, 68:101–128, 2017.

[32] Laura A Bradfield, Beatrice K Leung, Susan Boldt, Sophia Liang, and Bernard W Balleine. Goal-directed actions transiently depend on dorsal hippocampus. *Nature Neuroscience*, pages 1–4, 2020.

[33] Wolfram Schultz. Dopamine reward prediction error coding. *Dialogues in clinical neuroscience*, 18(1):23, 2016.

[34] Kimberly A Kempadoo, Eugene V Mosharov, Se Joon Choi, David Sulzer, and Eric R Kandel. Dopamine release from the locus coeruleus to the dorsal hippocampus promotes spatial learning and memory. *Proceedings of the National Academy of Sciences*, 113(51):14835–14840, 2016.

A Appendix

A.1 Tabular algorithm for learning successor representation

Here we show the algorithm used to learn the λ-successor representation in a tabular setting. We update at the end of episode to make it compatible with Algorithm 1.

Algorithm 2: Offline update for learning tabular λ-SR

Given tabular λ-SR matrix, m_π, of size $|S| \times |S|$;
Given a on-policy trajectory of states from the latest episode: $s_0, s_1, ..., s_{T-1}$;
for s_t *where* $t \leftarrow 0, ..., T-1$ **do**
    while *time allows* **do**
        Sample $s_k \sim \{s_0, s_1, ..., s_{T-1}\}$ from the trajectory randomly with replacement;
        $\Delta m(s_t, s_k) \leftarrow \mathbf{1}\{s_t = s_k\} + (\gamma\lambda)\, m_\pi(s_{t+1}, s_k) - m_\pi(s_t, s_k)$;
        $m(s_t, s_k) \leftarrow m(s_t, s_k) + \alpha \Delta m(s_t, s_k)$;
    end
end

For all experiments, in the inner loop of algorithm 2, we sample only $n \approx (0.05 \cdot T)$ number of states (s_k) (T is trajectory length). While larger samples can help with faster learning, small samples worked well in our task, and had better computational efficiency.

We note that in the tabular case, the successor representation is upper-bounded by $m_\pi(s, s') \leq \sum_{n=0}^{\infty} (\lambda\gamma)^n = \frac{1}{1-\gamma\lambda}, \forall s, s'$. As a heuristic we therefore initialize the successor matrix to $m_{\text{init}}(\cdot, \cdot) = 0.5 \cdot \frac{1}{1-\min(\gamma\lambda, 0.95)}$. The 0.5 constant is akin to saying any state has a 0.5 chance of reaching any other states at any times, and the min is used to prevent the values from blowing up to infinity. We emphasize this is a rough heuristic we used in our experiments and we did not explore extensively the effect of initialization on learning the tabular SR matrix as this is not the main focus of this work.

A.2 Experiment details

For all random chain experiments, we initialize the tabular value function (for the offline λ-return and the λ-SRE algorithms which contains an explicitely learned value function) to 0.5, the average value of the random chance MDP. This follows from the method used in Chapter 12 of [18]. For all experiments we run 50 seeds and average over them for the results shown. Error bars denote 95% confidence interval (standard error).

Section 4.1 Experiments Again following the set-up in [18], we use no discounting ($\gamma = 1$) and evaluate the root mean squared error (RMSE) of each agent's value function after 10 episodes' worth of experience. We also use the same stepsize for all learning procedures (e.g. for value learning and for SR learning in the λ-SRE agent).

For the SR (with decoupled reward function) agent, we initially attempted to learn both the SR matrix and reward function from the on-policy distribution (as the reward function can be learned with supervised learning). However, we noticed in exploratory experiments that the learning rate needs to be set differently, specifically the reward learning seem to prefer a very small learning rate, while the SR learning preferred larger ones. Using the same stepsizes resulted in either a complete lack of learning or divergence. Thus, for simplicity, we opted to just give the SR agent the true reward function and let it learn *only* the SR matrix.

We also show the 100 episode learning curve for the SR agent exclusively here. Combining Fig.3 and Fig.5, we observe that λ-SRE has lower value error as compared to the optimal λ-return at all points of learning. The failure of SR to learn might be due to multiple reasons. Initialization of the SR matrix plays a potential role, though this is not a problem with the λ-SRE. The undiscounted case may also be difficult to learn in as future occupancy would have high value. This may also be due to small errors in the SR matrix resulting in large errors in the value function [16].

Section 4.2 experiments Here we used learning rate of $\alpha = 0.1$ for all experiments and run for 3000 episodes, with discounting $\gamma = 0.8$ (which was chosen to make the learning process easier for

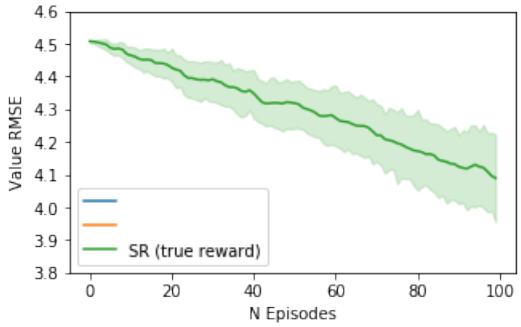

Figure 5: Value root mean squared error (RMSE), SR agent (with true reward function given), 100 episodes of training

the SR agent as high γ may result in more unstable learning), and similarly averaging over 50 seeds. Note that we *learn* from scratch the SR matrix for both the λ-SRE and SR algorithms here, rather than using a pre-computed SR matrix.

A.3 Further comparison with Source Traces

Comparing our current work to [20], both works require learning the SR function m_π, which is a mutual weaknesses. However, even if m_π can be learned, the backward view has two additional weaknesses for non-tabular cases: (i) it assumes access to the entire state space; and (ii) it is unclear how to interpret the SR function as an eligibility trace, as m_π maps to a scalar, while (non-tabular) eligibility traces require one to keep track of all model parameters. We note this latter perspective is explored further in [21], which can be viewed as "back-in-time" successor *features* for linear value functions.