Appendix: A State Representation for Diminishing Rewards

GIFs of navigation agents can be found at lambdarepresentation.github.io **and in the supplementary material.**

A Derivation of λR Recursion

We provide a step-by-step derivation of the λR recursion in Eq. (4.3):

$$
\begin{aligned}
\Phi^{\pi}_{\lambda}(s,s') &= \mathbb{E}\left[\sum_{k=0}^{\infty}\lambda^{n_t(s',k)}\gamma^k\mathbb{1}(s_{t+k}=s')\Big|s_t=s\right]\\
&= \mathbb{E}\left[\mathbb{1}(s_t=s')+\sum_{k=1}^{\infty}\lambda^{n_t(s',k)}\gamma^k\mathbb{1}(s_{t+k}=s')\Big|s_t=s\right]\\
&\overset{(i)}{=} \mathbb{E}\left[\mathbb{1}(s_t=s')+\lambda^{n_t(s',1)}\gamma\sum_{k=1}^{\infty}\lambda^{n_{t+1}(s',k)}\gamma^{k-1}\mathbb{1}(s_{t+k}=s')\Big|s_t=s\right]\\
&\overset{(ii)}{=} \mathbb{E}\Bigg[\mathbb{1}(s_t=s')+\mathbb{1}(s_t=s')\lambda\gamma\sum_{k=1}^{\infty}\lambda^{n_{t+1}(s',k)}\gamma^{k-1}\mathbb{1}(s_{t+k}=s')\\
&\qquad\qquad +\gamma(1-\mathbb{1}(s_t=s'))\sum_{k=1}^{\infty}\lambda^{n_{t+1}(s',k)}\gamma^{k-1}\mathbb{1}(s_{t+k}=s')\Big|s_t=s\Bigg]\\
&= \mathbb{1}(s_t=s')+\mathbb{1}(s_t=s')\lambda\gamma\mathbb{E}_{s_{t+1}\sim p^{\pi}}\Phi^{\pi}_{\lambda}(s_{t+1},s')+\gamma(1-\mathbb{1}(s_t=s'))\mathbb{E}_{s_{t+1}\sim p^{\pi}}\Phi^{\pi}_{\lambda}(s_{t+1},s')\\
&= \mathbb{1}(s_t=s')(1+\gamma\lambda\mathbb{E}_{s_{t+1}\sim p^{\pi}}\Phi^{\pi}_{\lambda}(s_{t+1},s'))+\gamma(1-\mathbb{1}(s_t=s'))\mathbb{E}_{s_{t+1}\sim p^{\pi}}\Phi^{\pi}_{\lambda}(s_{t+1},s'),
\end{aligned}
\tag{A.1}
$$

where (i) is because $n_t(s',k)=n_t(s',1)+n_{t+1}(s',k)$ and (ii) is because

$$
\lambda^{n_t(s',1)}=\lambda^{\mathbb{1}(s_t=s')}=\mathbb{1}(s_t=s')\lambda+(1-\mathbb{1}(s_t=s')).
$$

B Theoretical Analysis

Here, we provide proofs for the theoretical results in the main text.

Lemma 3.1 (Bellman Impossibility)**.** *Given a reward function of the form Eq. (3.1), it is impossible to define a Bellman equation solely using the resulting value function and immediate reward.*

Proof. We have

$$
\begin{aligned}
V^{\pi}(s) &= \mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k r_{\lambda}(s_{t+k},k)\Big|s_t=s\right]\\
&= \bar{\mathbf{r}}^{\mathsf{T}}\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k\lambda^{n_t(s_{t+k},k)}\mathbf{1}(s_{t+k})\Big|s_t=s\right]\\
&= \bar{\mathbf{r}}^{\mathsf{T}}\left[\mathbf{1}(s_t)\odot(1+\gamma\lambda\mathbb{E}_{p^{\pi}}[\Phi^{\pi}_{\lambda}(s_{t+1})])+\gamma(1-\mathbf{1}(s_t))\odot\mathbb{E}_{p^{\pi}}[\Phi^{\pi}_{\lambda}(s_{t+1})]\right]\\
&= \bar{r}(s_t)+\gamma\mathbb{E}_{p^{\pi}}[V^{\pi}(s_{t+1})]+\gamma(\lambda-1)\bar{r}(s_t)\mathbb{E}_{p^{\pi}}[\Phi^{\pi}_{\lambda}(s_{t+1},s_t)]
\end{aligned}
$$

The additional term in red cannot be eliminated, and is generated by the elementwise product by the one-hot vector $\mathbf{1}(s_t)$, which prevents the associated inner products between $\bar{\mathbf{r}}$ and $\Phi^\pi_\lambda(s_{t+1})$ from producing $V^\pi(s_{t+1})$. □

The following establishes $\mathcal{G}^\pi_\lambda$ as a contraction.

Lemma B.1 (Contraction)**.** *Let $\mathcal{G}^\pi_\lambda$ be the operator as defined in Definition 4.2 for some stationary policy π. Then for any two matrices $\Phi, \Phi' \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$,*

$$|\mathcal{G}^\pi_\lambda \Phi(s,s') - \mathcal{G}^\pi_\lambda \Phi'(s,s')| \leq \gamma |\Phi(s,s') - \Phi'(s,s')|.$$

Proof. We have

$$\begin{aligned}
|(\mathcal{G}^\pi_\lambda \Phi - \mathcal{G}^\pi_\lambda \Phi')_{s,s'}| &= |(I \odot (\mathbf{1}\mathbf{1}^\mathsf{T} + \gamma\lambda P^\pi \Phi) + \gamma(\mathbf{1}\mathbf{1}^\mathsf{T} - I) \odot P^\pi \Phi \\
&\qquad - I \odot (\mathbf{1}\mathbf{1}^\mathsf{T} + \gamma\lambda P^\pi \Phi') - \gamma(\mathbf{1}\mathbf{1}^\mathsf{T} - I) \odot P^\pi \Phi')_{s,s'}| \\
&= |(I \odot \gamma\lambda P^\pi(\Phi - \Phi') + \gamma(\mathbf{1}\mathbf{1}^\mathsf{T} - I) \odot P^\pi(\Phi - \Phi'))_{s,s'}| \\
&= |((I \odot \lambda \mathbf{1}\mathbf{1}^\mathsf{T} + \mathbf{1}\mathbf{1}^\mathsf{T} - I) \odot \gamma P^\pi(\Phi - \Phi'))_{s,s'}| \\
&\overset{(i)}{\leq} |(\gamma P^\pi(\Phi - \Phi'))_{s,s'}| \\
&= \gamma |(P^\pi(\Phi - \Phi'))_{s,s'}| \\
&\leq \gamma |(\Phi - \Phi')_{s,s'}|,
\end{aligned}$$

where (i) comes from using $\lambda \leq 1$ and simplifying. □

Note that we can actually get a tighter contraction factor of $\lambda\gamma$ for $s = s'$. Given this contractive property, we can prove its convergence with the use of the following lemma.

Lemma B.2 (Max λR)**.** *The maximum possible value of $\Phi^\pi_\lambda(s,s')$ is*

$$\frac{\mathbb{1}(s=s') + (1 - \mathbb{1}(s=s'))\gamma}{1 - \lambda\gamma}.$$

Proof. For $s = s'$,

$$\Phi^\pi_\lambda(s,s) = 1 + \lambda\gamma \mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s_t)} \Phi^\pi_\lambda(s_{t+1}, s).$$

This is just the standard SR recursion with discount factor $\lambda\gamma$, so the maximum is

$$\sum_{k=0}^{\infty} (\lambda\gamma)^k = \frac{1}{1-\lambda\gamma}. \tag{B.1}$$

For $s \neq s'$, $\mathbb{1}(s_t = s') = 0$, so

$$\Phi^\pi_\lambda(s,s') = \gamma \mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s_t)} \Phi^\pi_\lambda(s_{t+1}, s').$$

Observe that $\Phi^\pi_\lambda(s,s) \geq \Phi^\pi_\lambda(s,s')$ for $s' \neq s$, so the maximum is attained for $s_{t+1} = s'$. We can then use the result for $s = s'$ to get

$$\Phi^\pi_\lambda(s,s') = \gamma \left(\frac{1}{1-\lambda\gamma}\right). \tag{B.2}$$

Combining Eq. (B.1) and Eq. (B.2) yields the desired result. □

Proposition 4.1 (Convergence)**.** *Under the conditions assumed above, set* $\Phi^{(0)} = (1-\lambda)I$*. For* $k = 1, 2, \ldots$*, suppose that* $\Phi^{(k+1)} = \mathcal{G}_\lambda^\pi \Phi^{(k)}$*. Then*

$$|(\Phi^{(k)} - \Phi_\lambda^\pi)_{s,s'}| \leq \frac{\gamma^{k+1}}{1-\lambda\gamma}.$$

Proof. Using the notation $X_{s,s'} = X(s,s')$ for a matrix X:

$$\begin{aligned} |(\Phi^{(k)} - \Phi_\lambda^\pi)_{s,s'}| &= |(\mathcal{G}_\lambda^k \Phi^{(0)} - \mathcal{G}_\lambda^k \Phi_\lambda^\pi)_{s,s'}| \\ &= |(\mathcal{G}_\lambda^k \Phi^{(0)} - \Phi_\lambda^\pi)_{s,s'}| \\ &\overset{(i)}{\leq} \gamma^k |(\Phi^{(0)} - \Phi_\lambda^\pi)_{s,s'}| \\ &\overset{(ii)}{=} \gamma^k \Phi_\lambda^\pi(s,s') \\ &\overset{(iii)}{\leq} \frac{\gamma^{k+1}}{1-\lambda\gamma} \end{aligned} \tag{B.3}$$

where (i) is due to Lemma B.1, (ii) is because $\Phi^{(0)}(s,s') = 0$ for $s \neq s'$, and (iii) is due to Lemma B.2. □

Lemma B.3 (Subadditivity)**.** *For any* $s \in \mathcal{S}$*, policy* π*,* $\lambda \in [0,1)$*, and disjoint measurable sets* $A, B \subseteq \mathcal{S}$*,*

$$\Phi_\lambda^\pi(s, A \cup B) < \Phi_\lambda^\pi(s,A) + \Phi_\lambda^\pi(s,B).$$

Proof. Note that for disjoint sets A, B, we have $n_t(A \cup B, k) = n_t(A,k) + n_t(B,k)$. Hence, conditioned on some policy π and $s_t = s$,

$$\begin{aligned} \lambda^{n_t(A\cup B,k)} \mathbb{P}(s_{t+k} \in A \cup B) &= \lambda^{n_t(A,k)} \lambda^{n_t(B,k)} \mathbb{P}(s_{t+k} \in A) + \lambda^{n_t(A,k)} \lambda^{n_t(B,k)} \mathbb{P}(s_{t+k} \in B) \\ &\leq \lambda^{n_t(A,k)} \mathbb{P}(s_{t+k} \in A) + \lambda^{n_t(B,k)} \mathbb{P}(s_{t+k} \in B), \end{aligned}$$

where the first line follows from $\mathbb{P}(s_{t+k} \in A \cup B) = \mathbb{P}(s_{t+k} \in A) + \mathbb{P}(s_{t+k} \in B)$. Equality holds over all A, B, t, k if and only if $\lambda = 1$. Summing over k yields the result. □

B.1 Proof of Theorem 5.1

We first prove two results, which rely throughout on the fact that $\Phi_\lambda(s,a,s') \leq \frac{1}{1-\lambda\gamma}$ for all s, a, s', which follows from Lemma B.2. For simplicity, we also assume throughout that all rewards are non-negative, but this assumption can easily be dropped by taking absolute values of rewards. The proofs presented here borrow ideas from those of [16].

Lemma B.4. *Let* $\{M_j\}_{j=1}^n \subseteq \mathcal{M}$ *and* $M \in \mathcal{M}$ *be a set of tasks in an environment* $\mathcal{M}$ *with diminishing rate* λ *and let* $Q^{\pi_j^*}$ *denote the action-value function of an optimal policy of* M_j *when executed in* M*. Given estimates* $\tilde{Q}^{\pi_j}$ *such that* $\|Q^{\pi_j^*} - \tilde{Q}^{\pi_j}\|_\infty \leq \epsilon$ *for all* j*, define*

$$\pi(s) \in \operatorname*{argmax}_a \max_j \tilde{Q}^{\pi_j}(s,a).$$

Then,

$$Q^\pi(s,a) \geq \max_j Q^{\pi_j^*}(s,a) - \frac{1}{1-\gamma}\left(2\epsilon + \frac{\gamma(1-\lambda)r(s,a)}{1-\lambda\gamma}\right),$$

where r denotes the reward function of M.

Proof. Define $\tilde{Q}_{\max}(s,a) \coloneqq \max_j \tilde{Q}^{\pi_j}(s,a)$ and $Q_{\max}(s,a) \coloneqq \max_j Q^{\pi_j^*}(s,a)$. Let T^ν denote the Bellman operator of a policy ν in task M. For all $(s,a) \in \mathcal{S} \times \mathcal{A}$ and all j,

$$
\begin{aligned}
T_i^\pi \tilde{Q}_{\max}(s,a) &= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^\pi(s', \pi(s'), s) + \tilde{Q}_{\max}(s', \pi(s')) \right) \\
&= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^\pi(s', \pi(s'), s) + \max_b \tilde{Q}_{\max}(s', b) \right) \\
&\geq r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^\pi(s', \pi(s'), s) + \max_b Q_{\max}(s', b) \right) - \gamma\epsilon \\
&\geq r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^\pi(s', \pi(s'), s) + Q_{\max}(s', \pi_j^*(s')) \right) - \gamma\epsilon \\
&\geq r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^\pi(s', \pi(s'), s) + Q_i^{\pi_j^*}(s', \pi_j^*(s')) \right) - \gamma\epsilon \\
&= r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^{\pi_j^*}(s', \pi_j^*(s'), s) + Q_i^{\pi_j^*}(s', \pi_j^*(s')) \right) - \gamma\epsilon \\
&\quad + \gamma(\lambda - 1) r(s,a) \sum_{s'} p(s'|s,a) \left(\Phi^\pi(s', \pi(s'), s) - \Phi^{\pi_j^*}(s', \pi_j^*(s'), s) \right) \\
&\geq T_i^{\pi_j^*} Q_i^{\pi_j^*}(s,a) - \gamma\epsilon - \frac{\gamma(1-\lambda) r(s,a)}{1 - \lambda\gamma} \\
&= Q_i^{\pi_j^*}(s,a) - \gamma\epsilon - \frac{\gamma(1-\lambda) r(s,a)}{1 - \lambda\gamma}.
\end{aligned}
$$

This holds for any j, so

$$
\begin{aligned}
T^\pi \tilde{Q}_{\max}(s,a) &\geq \max_j Q_i^{\pi_j^*}(s,a) - \gamma\epsilon - \frac{\gamma(1-\lambda) r(s,a)}{1 - \lambda\gamma} \\
&= Q_{\max}(s,a) - \gamma\epsilon - \frac{\gamma(1-\lambda) r(s,a)}{1 - \lambda\gamma} \\
&\geq \tilde{Q}_{\max}(s,a) - \epsilon - \gamma\epsilon - \frac{\gamma(1-\lambda) r(s,a)}{1 - \lambda\gamma}.
\end{aligned}
$$

Next, note that for any $c \in \mathbb{R}$,

$$
\begin{aligned}
T^\pi(\tilde{Q}_{\max}(s,a) + c) &= T^\pi \tilde{Q}_{\max}(s,a) + \gamma \sum_{s'} p(s'|s,a) c \\
&= T^\pi \tilde{Q}_{\max}(s,a) + \gamma c.
\end{aligned}
$$

Putting everything together, and using the fact that T^ν is monotonic and contractive,

$$\begin{aligned}
Q_i^\pi(s,a) &= \lim_{k\to\infty} (T^\pi)^k \tilde{Q}_{\max}(s,a) \\
&\geq \lim_{k\to\infty} \left[\tilde{Q}_{\max}(s,a) - \left(\epsilon(1+\gamma) - \frac{\gamma(1-\lambda)r(s,a)}{1-\lambda\gamma} \right) \sum_{j=0}^{k} \gamma^j \right] \\
&\geq \tilde{Q}_{\max}(s,a) - \frac{1}{1-\gamma} \left(\epsilon(1+\gamma) - \frac{\gamma(1-\lambda)r(s,a)}{1-\lambda\gamma} \right) \\
&\geq Q_{\max}(s,a) - \epsilon - \frac{1}{1-\gamma} \left(\epsilon(1+\gamma) - \frac{\gamma(1-\lambda)r(s,a)}{1-\lambda\gamma} \right) \\
&\geq Q^{\pi_j^*}(s,a) - \frac{1}{1-\gamma} \left(2\epsilon + \frac{\gamma(1-\lambda)r(s,a)}{1-\lambda\gamma} \right).
\end{aligned}$$

This holds for every j, hence the result. □

Lemma B.5. *Let ν be any policy, $\lambda, \hat{\lambda} \in [0,1]$, and Q_λ denote a value function with respecting to diminishing rate λ. Then,*

$$\|Q_\lambda^\nu - Q_{\hat{\lambda}}^\nu\|_\infty \leq \frac{|\lambda - \hat{\lambda}| \|r\|_\infty}{1-\gamma}.$$

Proof. The proof follows from the definition of Q: for every $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned}
|Q_\lambda^\nu(s,a) - Q_{\hat{\lambda}}^\nu(s,a)| &= \left| \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \left(\lambda^{n_t(s_{t+k},k)} - \hat{\lambda}^{n_t(s_{t+k},k)} \right) r(s_{t+k}) \middle| s_t = s, a_t = a \right] \right| \\
&\leq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \left| \lambda^{n_t(s_{t+k},k)} - \hat{\lambda}^{n_t(s_{t+k},k)} \right| r(s_{t+k}) \middle| s_t = s, a_t = a \right] \\
&= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}) \left| \lambda - \hat{\lambda} \right| \sum_{j=0}^{n_t(s_{t+k},k)-1} \lambda^{n_t(s_{t+k},k)-1-j} \hat{\lambda}^j \middle| s_t = s, a_t = a \right] \\
&\leq |\lambda - \hat{\lambda}| \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}) \middle| s_t = s, a_t = a \right] \\
&\leq \frac{|\lambda - \hat{\lambda}| \|r\|_\infty}{1-\gamma}.
\end{aligned}$$

□

Theorem 5.1 (GPI). *Let $\{M_j\}_{j=1}^n \subseteq \mathcal{M}$ and $M \in \mathcal{M}$ be a set of tasks in an environment $\mathcal{M}$ and let $Q^{\pi_j^*}$ denote the action-value function of an optimal policy of M_j when executed in M. Assume that the agent uses diminishing rate $\hat{\lambda}$ that may differ from the true environment diminishing rate λ. Given estimates $\tilde{Q}^{\pi_j}$ such that $\|Q^{\pi_j^*} - \tilde{Q}^{\pi_j}\|_\infty \leq \epsilon$ for all j, define*

$$\pi(s) \in \operatorname*{argmax}_a \max_j \tilde{Q}^{\pi_j}(s,a).$$

Then,

$$Q^\pi(s,a) \geq \max_j Q^{\pi_j^*}(s,a) - \frac{1}{1-\gamma} \left(2\epsilon + |\lambda - \hat{\lambda}| \|r\|_1 + \frac{\gamma(1-\lambda)r(s,a)}{1-\lambda\gamma} \right).$$

Proof. Let Q_λ denote a value function with respect to diminishing constant λ. We wish to bound

$$\max_j Q_{\hat{\lambda}}^{\pi_j^*}(s,a) - Q_\lambda^\pi(s,a),$$

i.e., the value of the GPI policy with respect to the true λ compared to the maximum value of the constituent policies π_j^* used for GPI, which were used assuming $\hat{\lambda}$. By the triangle inequality,

$$\begin{aligned}\max_j Q_{\hat{\lambda}}^{\pi_j^*}(s,a) - Q_\lambda^\pi(s,a) &\leq \max_j Q_\lambda^{\pi_j^*}(s,a) - Q_\lambda^\pi(s,a) + |\max_j Q_\lambda^{\pi_j^*}(s,a) - \max_j Q_{\hat{\lambda}}^{\pi_j^*}(s,a)| \\ &\leq \underbrace{\max_j Q_\lambda^{\pi_j^*}(s,a) - Q_\lambda^\pi(s,a)}_{(1)} + \max_j \underbrace{|Q_\lambda^{\pi_j^*}(s,a) - Q_{\hat{\lambda}}^{\pi_j^*}(s,a)|}_{(2)}.\end{aligned}$$

We bound (1) by Lemma B.4 and (2) by Lemma B.5 (noting that $\|r\|_\infty \leq \|r\|_1$) to get the result. □

B.2 An Extension of Theorem 5.1

Inspired by [17], we prove an extension of Theorem 5.1:

Theorem B.1. *Let $M \in \mathcal{M}$ be a task in an environment $\mathcal{M}$ with true diminishing constant λ. Suppose we perform GPI assuming a diminishing constant $\hat{\lambda}$:*

> *Let $\{M_j\}_{j=1}^n$ and M_i be tasks in $\mathcal{M}$ and let $Q_i^{\pi_j^*}$ denote the action-value function of an optimal policy of M_j when executed in M_i. Given estimates $\tilde{Q}_i^{\pi_j}$ such that $\|Q_i^{\pi_j^*} - \tilde{Q}_i^{\pi_j}\|_\infty \leq \epsilon$ for all j, define $\pi(s) \in \operatorname{argmax}_a \max_j \tilde{Q}_i^{\pi_j}(s,a)$.*

Let $Q_{\hat{\lambda}}^\pi$ and $Q_\lambda^{\pi^*}$ denote the action-value functions of π and the M-optimal policy π^* when executed in M, respectively. Then,

$$\|Q_\lambda^{\pi^*} - Q_{\hat{\lambda}}^\pi\|_\infty \leq \frac{2}{1-\gamma}\left(\frac{1}{2}|\lambda - \hat{\lambda}|\|r\|_\infty + \epsilon + \|r - r_i\|_\infty + \min_j \|r_i - r_j\|_\infty\right) + \frac{1-\lambda}{1-\lambda\gamma}C,$$

where C is a positive constant not depending on λ:

$$C = \gamma\frac{2\|r - r_i\|_\infty + 2\min_j \|r_i - r_j\|_\infty + \min\left(\|r\|_\infty, \|r_i\|_\infty\right) + \min\left(\|r_i\|_\infty, \|r_1\|_\infty, \ldots, \|r_n\|_\infty\right)}{1-\gamma}.$$

Note that when $\lambda = 1$, we recover Proposition 1 of [17] with an additional term quantifying error incurred by $\hat{\lambda} \neq \lambda$. The proof relies on two other technical lemmas, presented below.

Lemma B.6.

$$\|Q^{\pi^*} - Q_i^{\pi_i^*}\|_\infty \leq \frac{\|r - r_i\|_\infty}{1-\gamma} + \gamma(1-\lambda)\frac{\min\left(\|r\|_\infty, \|r_i\|_\infty\right) + \|r - r_i\|_\infty}{(1-\gamma)(1-\lambda\gamma)}.$$

Proof. Define $\Delta_i \coloneqq \|Q^{\pi^*} - Q_i^{\pi_i^*}\|_\infty$. For any $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned}
|Q^{\pi^*}(s,a) - Q_i^{\pi_i^*}(s,a)| &= \Big| r(s,a) + \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r(s,a) \Phi^{\pi^*}(s', \pi^*(s'), s) + Q^{\pi^*}(s', \pi^*(s')) \right) \\
&\quad - r_i(s,a) - \gamma \sum_{s'} p(s'|s,a) \left((\lambda - 1) r_i(s,a) \Phi^{\pi_i^*}(s', \pi_i^*(s'), s) + Q_i^{\pi_i^*}(s', \pi_i^*(s')) \right) \Big| \\
&\leq |r(s,a) - r_i(s,a)| + \gamma \sum_{s'} p(s'|s,a) |Q^{\pi^*}(s,a) - Q_i^{\pi_i^*}(s,a)| \\
&\quad + \gamma(\lambda - 1) \sum_{s'} p(s'|s,a) \left| r(s,a) \Phi^{\pi^*}(s', \pi^*(s'), s) - r_i(s,a) \Phi^{\pi_i^*}(s', \pi_i^*(s'), s) \right| \\
&\leq \|r - r_i\|_\infty + \gamma \Delta_i + \gamma(1 - \lambda) \|r\Phi^{\pi^*} - r_i \Phi^{\pi_i^*}\|_\infty.
\end{aligned}$$

The third term decomposes as

$$\begin{aligned}
\|r\Phi^{\pi^*} - r_i \Phi^{\pi_i^*}\|_\infty &\leq \|r\Phi^{\pi^*} - r\Phi^{\pi_i^*}\|_\infty + \|r\Phi^{\pi_i^*} - r_i \Phi^{\pi_i^*}\|_\infty \\
&\leq \frac{\|r\|_\infty + \|r - r_i\|_\infty}{1 - \lambda\gamma}.
\end{aligned}$$

We could equivalently use the following decomposition:

$$\begin{aligned}
\|r\Phi^{\pi^*} - r_i \Phi^{\pi_i^*}\|_\infty &\leq \|r\Phi^{\pi^*} - r_i \Phi^{\pi^*}\|_\infty + \|r_i\Phi^{\pi^*} - r_i \Phi^{\pi_i^*}\|_\infty \\
&\leq \frac{\|r_i\|_\infty + \|r - r_i\|_\infty}{1 - \lambda\gamma},
\end{aligned}$$

and so

$$\|r\Phi^{\pi^*} - r_i \Phi^{\pi_i^*}\|_\infty \leq \frac{\min\left(\|r\|_\infty, \|r_i\|_\infty\right) + \|r - r_i\|_\infty}{1 - \lambda\gamma}.$$

The inequalities above hold for all s, a and so

$$\begin{aligned}
&\Delta_i \leq \|r - r_i\|_\infty + \gamma \Delta_i + \gamma(1 - \lambda) \frac{\min\left(\|r\|_\infty, \|r_i\|_\infty\right) + \|r - r_i\|_\infty}{1 - \lambda\gamma} \\
\implies &\Delta_i \leq \frac{\|r - r_i\|_\infty}{1 - \gamma} + \gamma(1 - \lambda) \frac{\min\left(\|r\|_\infty, \|r_i\|_\infty\right) + \|r - r_i\|_\infty}{(1 - \gamma)(1 - \lambda\gamma)}.
\end{aligned}$$

Hence the result. □

Lemma B.7. *For any policy π,*

$$\|Q_i^\pi - Q^\pi\|_\infty \leq \frac{\|r - r_i\|_\infty}{1 - \gamma} + \gamma(1 - \lambda) \frac{\|r - r_i\|_\infty}{(1 - \gamma)(1 - \lambda\gamma)}.$$

Proof. Write $\Delta_i \coloneqq \|Q_i^\pi - Q^\pi\|_\infty$. Proceeding as in the previous lemma, for all $(s,a) \in \mathcal{S} \times \mathcal{A}$, we have

$$\begin{aligned}
|Q_i^\pi(s,a) - Q^\pi(s,a)| &= \Big| r_i(s,a) + \gamma \sum_{s'} p(s'|s,a)\left((\lambda-1) r_i(s,a)\Phi^\pi(s',\pi(s'),s) + Q_i^\pi(s',\pi(s'))\right) \\
&\quad - r(s,a) - \gamma \sum_{s'} p(s'|s,a)\left((\lambda-1) r(s,a)\Phi^\pi(s',\pi(s'),s) + Q^\pi(s',\pi(s'))\right) \\
&\leq |r(s,a) - r_i(s,a)| + \gamma \sum_{s'} p(s'|s,a)(1-\lambda)|r(s,a) - r_i(s,a)|\Phi^\pi(s',\pi(s'),s) \\
&\quad + \gamma \sum_{s'} p(s'|s,a)|Q_i^\pi(s',\pi(s')) - Q^\pi(s',\pi(s'))| \\
&\leq \|r - r_i\|_\infty + \gamma(1-\lambda)\|r - r_i\|_\infty \frac{1}{1-\lambda\gamma} + \gamma\Delta_i' \\
\implies \Delta_i' &\leq \|r - r_i\|_\infty + \frac{\gamma(1-\lambda)\|r - r_i\|_\infty}{1-\lambda\gamma} + \gamma\Delta_i' \\
\implies \Delta_i' &\leq \frac{\|r - r_i\|_\infty}{1-\gamma} + \frac{\gamma(1-\lambda)\|r - r_i\|_\infty}{(1-\gamma)(1-\lambda\gamma)}.
\end{aligned}$$

□

Finally, we prove Theorem B.1:

Proof of Theorem B.1. By the triangle inequality,

$$\|Q_\lambda^{\pi^*} - Q_{\hat{\lambda}}^\pi\|_\infty \leq \|Q_\lambda^{\pi^*} - Q_\lambda^\pi\|_\infty + \|Q_\lambda^\pi - Q_{\hat{\lambda}}^\pi\|_\infty.$$

By Lemma B.5, the second term is bounded above by

$$\frac{|\lambda - \hat{\lambda}|\|r\|_\infty}{1-\gamma}.$$

The first term decomposes as follows (dropping the λ subscript on all action-value functions for clarity):

$$\|Q^{\pi^*} - Q^\pi\|_\infty \leq \underbrace{\|Q^{\pi^*} - Q_i^{\pi_i^*}\|_\infty}_{(1)} + \underbrace{\|Q_i^{\pi_i^*} - Q_i^\pi\|_\infty}_{(2)} + \underbrace{\|Q_i^\pi - Q^\pi\|_\infty}_{(3)}.$$

Applying Lemma B.4 to (2) (but with respect to M_i rather than M), we have that for any j,

$$\begin{aligned}
Q_i^{\pi_i^*}(s,a) - Q_i^\pi(s,a) &\leq Q_i^{\pi_i^*}(s,a) - Q_i^{\pi_j^*}(s,a) + \frac{1}{1-\gamma}\left(2\epsilon + \frac{\gamma(1-\lambda)r_i(s,a)}{1-\lambda\gamma}\right) \\
\implies \|Q_i^{\pi_i^*} - Q_i^\pi\|_\infty &\leq \underbrace{\|Q_i^{\pi_i^*} - Q_j^{\pi_j^*}\|_\infty}_{(2.1)} + \underbrace{\|Q_j^{\pi_j^*} - Q_i^{\pi_j^*}\|_\infty}_{(2.2)} + \frac{1}{1-\gamma}\left(2\epsilon + \frac{\gamma(1-\lambda)\|r_i\|_\infty}{1-\lambda\gamma}\right).
\end{aligned}$$

We bound (2.1) using Lemma B.6 and (2.2) using Lemma B.7 (but with respect to M_j rather than M):

$$\|Q_i^{\pi_i^*} - Q_j^{\pi_j^*}\|_\infty + \|Q_j^{\pi_j^*} - Q_i^{\pi_j^*}\|_\infty \leq \frac{2\|r_i - r_j\|_\infty}{1-\gamma} + \gamma(1-\lambda)\frac{\min\left(\|r_i\|_\infty, \|r_j\|_\infty\right) + 2\|r_i - r_j\|_\infty}{(1-\gamma)(1-\lambda\gamma)}.$$

We then apply Lemma B.6 to (1) and Lemma B.7 to (3) to get the result.

□

C An nth Occupancy Representation

To generalize the first occupancy representation to account for reward functions of this type, it's natural to consider an *Nth occupancy representation*—that is, one which accumulates value only for the first N occupancies of one state s' starting from another state s:

Definition C.1 (NR). *For an MDP with finite $\mathcal{S}$, the Nth-occupancy representation (NR) for a policy π is given by $F^\pi \in [0, N]^{|\mathcal{S}|\times|\mathcal{S}|}$ such that*

$$\Phi^\pi_{(N)}(s, s') \triangleq \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^{t+k} \mathbb{1}\left(s_{t+k} = s', \#\left(\{j \mid s_{t+j} = s', j \in [0, k-1]\}\right) < N\right) \Big| s_t \right]. \tag{C.1}$$

Intuitively, such a representation sums the first N (discounted) occupancies of s' from time t to $t+k$ starting from $s_t = s$. We can also note that $\Phi^\pi_{(1)}$ is simply the FR and $\Phi_{(0)}(s, s') = 0\ \forall s, s'$. As with the FR and the SR, we can derive a recursive relationship for the NR:

$$\Phi^\pi_{(N)}(s, s') = \mathbb{1}(s_t = s')(1 + \gamma \mathbb{E}\Phi^\pi_{(N-1)}(s_{t+1}, s')) + \gamma(1 - \mathbb{1}(s_t = s'))\mathbb{E}\Phi^\pi_{(N)}(s_{t+1}, s'), \tag{C.2}$$

where the expectation is wrt $p^\pi(s_{t+1}|s_t)$. Once again, we can confirm that this is consistent with the FR by noting that for $N = 1$, the NR recursion recovers the FR recursion. Crucially, we also recover the SR recursion in the limit as $N \to \infty$:

$$\begin{aligned}\lim_{N \mapsto \infty} \Phi^\pi_{(N)}(s, s') &= \mathbb{1}(s_t = s')(1 + \gamma \mathbb{E}\Phi^\pi_{(\infty)}(s_{t+1}, s')) + \gamma(1 - \mathbb{1}(s_t = s'))\mathbb{E}\Phi^\pi_{(\infty)}(s_{t+1}, s') \\ &= \mathbb{1}(s_t = s') + \gamma \mathbb{E}\Phi^\pi_{(\infty)}(s_{t+1}, s').\end{aligned}$$

This is consistent with the intuition that the SR accumulates every (discounted) state occupancy in a potentially infinite time horizon of experience. While Definition C.1 admits a recursive form which is consistent with our intuition, Eq. (C.2) reveals an inconvenient intractability: the Bellman target for $\Phi^\pi_{(N)}$ requires the availability of $\Phi^\pi_{(N-1)}$. This is a challenge, because it means that if we'd like to learn any NR for finite $N > 1$, the agent also must learn and store $\Phi^\pi_{(1)}, \ldots \Phi^\pi_{(N-1)}$. Given these challenges, the question of how to learn a tractable general occupancy representation remains. From a neuroscientific perspective, a fixed depletion amount is also inconsistent with both behavioral observations and neural imaging [3], which indicate instead that utility disappears at a fixed rate in proportion to the *current remaining utility*, rather than in proportion to the *original utility*. We address these theoretical and practical issues in the next section.

D Further Experimental Details

D.1 Policy Evaluation

We perform policy evaluation for the policy shown in Fig. 4.1 on the 6×6 gridworld shown. The discount factor γ was set to 0.9 for all experiments, which were run for

$H = 10$ steps per episode. The error metric was the mean squared error:

$$Q_{error} \triangleq \frac{1}{|\mathcal{S}||\mathcal{A}|} \sum_{s,a} (Q^{\pi}(s,a) - \hat{Q}(s,a))^2, \tag{D.1}$$

where Q^{π} is the ground truth Q-values and $\hat{Q}$ is the estimate. Transitions are deterministic. For the dynamic programming result, we learned the λR using Eq. (4.3) for $\lambda \in \{0.5, 1.0\}$ and then measured the resulting values by multiplying the resulting λR by the associated reward vector $\mathbf{r} \in \{-1, 0, 1\}^{36}$, which was -1 in all wall states and $+1$ at the reward state g. We compared the results to the ground truth values. Dynamic programming was run until the maximum Bellman error across state-action pairs reduced below 5e-2. For the tabular TD learning result, we ran the policy for three episodes starting from every available (non-wall) state in the environment, and learned the λR for $\lambda \in \{0.5, 1.0\}$ as above, but using the online TD update:

$$\Phi_{\lambda}(s_t, a_t) \leftarrow \Phi_{\lambda}(s_t, a_t) + \alpha\delta_t,$$
$$\delta_t = \mathbf{1}(s_t) \odot (1 + \gamma\lambda\Phi_{\lambda}(s_{t+1}, a_{t+1})) + \gamma(1 - \mathbf{1}(s_t)) \odot \Phi_{\lambda}(s_{t+1}, a_{t+1}) - \Phi_{\lambda}(s_t, a_t),$$

where $a_{t+1} \sim \pi(\cdot \mid s_{t+1})$. The learned Q-values were then computed in the same way as the dynamic programming case and compared to the ground truth. For the λF result, we first learned Laplacian eigenfunction base features as described in [24] from a uniform exploration policy and normalized them to the range $[0, 1]$. We parameterized the base feature network as a 2-layer MLP with ReLU activations and 16 units in the hidden layer. We then used the base features to learn the λFs as in the tabular case, but with the λF network parameterized as a three-layer MLP with 16 units in each of the hidden layers and ReLU activations. All networks were optimized using Adam with a learning rate of 3e-4. The tabular and neural network experiments were repeated for three random seeds, the former was run for 1,500 episodes and the latter for 2,000.

D.2 Policy Learning

We ran the experiments for Fig. 5.2 in a version of the TwoRooms environment from the NeuroNav benchmark [21] with reward modified to decay with a specified $\lambda_{true} = 0.5$ and discount factor $\gamma = 0.95$. The initial rewards in the top right goal and the lower room goal locations were 5 and the top left goal had initial reward 10. The observations in the neural network experiment were one-hot state indicators. The tabular Q_{λ} experiments run the algorithm in Algorithm 1 for 500 episodes for $\lambda \in \{0.0, 0.5, 1.0\}$, with λ_{true} set to 0.5, repeated for three random seeds. Experiments used a constant step size $\alpha = 0.1$. There were five possible actions: up, right, down, left, and stay. The recurrent A2C agents were based on the implementation from the BSuite library [30] and were run for 7,500 episodes of maximum length $H = 100$ with $\gamma = 0.99$ using the Adam optimizer with learning rate 3e-4. The experiment was repeated for three random seeds. The RNN was an LSTM with 128 hidden units and three output heads: one for the policy, one for the value function, and one for the λF. The base features were one-hot representations of the current state, 121-dimensional in this case.

Algorithm 1: Online Tabular Q_λ-Learning Update

1: **Require:** Current λR-values $\Phi_\lambda^{(t)} \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|\times|\mathcal{S}|}$, current reward vector $\mathbf{r}^{(t)}$, observed (s_t, a_t, s_{t+1}) tuple

2: Compute Q_λ-values: $Q_\lambda^{(t)} \leftarrow (\Phi_\lambda^{(t)})^\intercal \mathbf{r}^{(t)}$

3: Select greedy action: $a_{t+1} \leftarrow \text{argmax}_{a\in\mathcal{A}} Q_\lambda^{(t)}(s_{t+1}, a)$

4: Update Φ_λ:

$$\Phi_\lambda^{(t+1)}(s_t, a_t) \leftarrow \Phi_\lambda^{(t)}(s_t, a_t) + \alpha\delta^{(t)}, \quad \text{where}$$
$$\delta^{(t)} = \mathbf{1}(s_t) \odot (1 + \gamma\lambda\Phi_\lambda^{(t)}(s_{t+1}, a_{t+1})) + \gamma(1 - \mathbf{1}(s_t)) \odot \Phi_\lambda^{(t)}(s_{t+1}, a_{t+1}) - \Phi_\lambda^{(t)}(s_t, a_t).$$

5: **Return** updated $\Phi_\lambda^{(t+1)}$

D.3 Tabular GPI

The agent is assumed to be given or have previously acquired four policies $\{\pi_0, \pi_1, \pi_2, \pi_3\}$ individually optimized to reach rewards located in each of the four rooms of the environment. There are three reward locations $\{g_0, g_1, g_2\}$ scattered across the rooms, each with its own initial reward $\bar{r} = [5, 10, 5]$ and all with $\lambda = 0.5$. At the beginning of each episode, an initial state s_0 is sampled uniformly from the set of available states. An episode terminates either when the maximum reward remaining in any of the goal states is less than 0.1 or when the maximum number of steps $H = 40$ is reached. Empty states carry a reward of 0, encountering a wall gives a reward of -1, and the discount factor is set to $\gamma = 0.97$.

For each of the four policies, we learn λRs with λ equal to 0, 0.5, and 1.0 using standard dynamic programming (Bellman error curves plotted in **??**), and record the returns obtained while performing GPE+GPI with each of these representations over the course of 50 episodes. Bellman error curves for the λRs are In the left panel of Fig. 5.3, we can indeed see that using the correct λ (0.5) nets the highest returns. Example trajectories for each of λR are shown in the remaining panels.

D.4 Pixel-Based GPI

In this case, the base policies Π were identical to those used in the tabular GPI experiments. First, we collected a dataset consisting of 340 observation trajectories $(o_0, o_1, \ldots, o_{H-1}) \in \mathcal{O}^H$ with $H = 19$ from each policy, totalling $6,460$ observations. Raw observations were $128 \times 128 \times 3$ and were converted to grayscale. The previous seven observations were stacked and used to train a Laplacian eigenfunction base feature network in the same way as [24]. For observations less than seven steps from the start of an episode, the remaining frames were filled in as all black observations (i.e., zeros). The network consisted of four convolutional layers with 32 3×3 filters with strides $(2, 2, 2, 1)$, each followed by a ReLU nonlinearity. This was then flattened and passed through a Layer Norm layer [31] and a $\tanh$ nonlinearity before three fully fully connected layers, the first two with 64 units each and ReLU nonlinearities and the final, output layer with 50 units. The output was L_2-normalized as in [24]. This network $\phi : \mathcal{O}^7 \mapsto \mathbb{R}^D$ (with $D = 50$) was trained on the stacked observations for 10 epochs using the Adam optimizer and learning rate

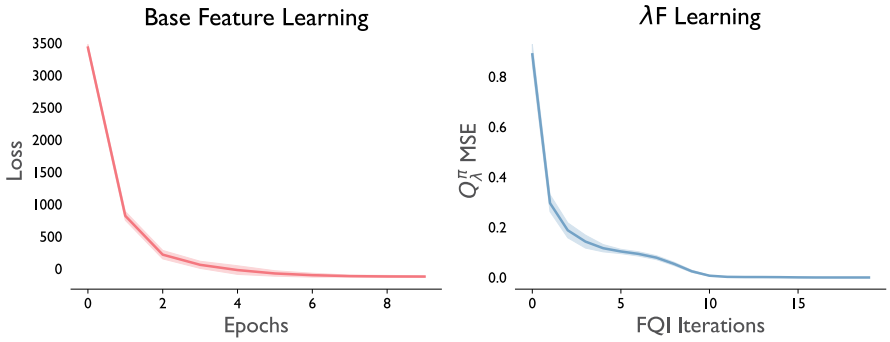

Figure D.1: **Learning curves for λF policy evaluation.** Results are averaged over three runs, with shading indicating one unit of standard error.

1e-4 with batch size $B = 64$. To perform policy evaluation, the resulting features, evaluated on the dataset of stacked observations were collected into their own dataset of $(s_t, a_{t+1}, s_{t+1}, a_{t+1})$ tuples, where $s_t \triangleq o_{t-6:t}$. The "states" were normalized to be between 0 and 1, and a vector $\mathbf{w}$ was fit to the actual associated rewards via linear regression on the complete dataset. The λF network was then trained using a form of neural fitted Q-iteration [FQI; 32] modified for policy evaluation with λFs (Algorithm 2). The architecture for the λF network was identical to the base feature network, with the exception that the hidden size of the fully connected layers was 128 and the output dimension was $D|\mathcal{A}| = 250$. FQI was run for $K = 20$ outer loop iterations, with each inner loop supervised learning setting run for $L = 100$ epochs on the current dataset. Supervised learning was done using Adam with learning rate 3e-4 and batch size $B = 64$. Given the trained networks, GPI proceeded as in the tabular case, i.e.,

$$a_t = \operatorname*{argmax}_{a \in \mathcal{A}} \max_{\pi \in \Pi} \mathbf{w}^\mathsf{T} \varphi^\pi_\theta(s_t, a). \tag{D.2}$$

50 episodes were run from random starting locations for $H = 50$ steps and the returns measured. Learning curves for the base features and for λF fitting are shown in Fig. D.1. The λF curve measures the mean squared error as in Eq. (D.1).

The feature visualizations were created by performing PCA to reduce the average λF representations for observations at each state in the environment to 2D. Each point in the scatter plot represents the reduced representation on the xy plane, and is colored according to the λ-conditioned value of the underlying state.

D.5 Continuous Control

λ-SAC See Appendix H for details.

D.6 Learning the λO with FB

Training the λO with the FB parameterization proceeds in much the same way as in [12], but adjusted for a different norm and non-Markovian environment. We summarize the learning procedure in Algorithm 3. The loss function $\mathcal{L}$ is derived in Appendix G, with the addition of the following regularizer:

$$\|\mathbb{E}_{s\sim\rho} B_\omega(s) B_\omega(s)^\top - I\|^2.$$

Algorithm 2: Fitted Q_λ-Iteration

Require: Dataset of base features $\{\phi(s) \in \mathbb{R}^D\}_{s \in \mathcal{S}}$, decay rate λ, discount factor γ, reward feature vector $\mathbf{w} \in \mathbb{R}^D$, batch size B, learning rate α
Initialize λF φ_θ parameters $\theta^{(1)}$ (we drop the subscript λ and superscript π for concision)
for $k = 1 \ldots, K$ **do**
  // Stage 1: Construct dataset
  $\mathcal{D} \leftarrow \varnothing$
  for $(s, a) \in \mathcal{S} \times \mathcal{A}$ **do**
    for $(s', a') \in \mathcal{S} \times \mathcal{A}$ **do**

$$\mathcal{D} \leftarrow \mathcal{D} \cup \left\{ \left((s,a), \underbrace{\mathbf{w}^\mathsf{T} \left[\phi(s) \odot (1 + \lambda\gamma \bar{\varphi}_{\theta^{(k)}}(s', a')) + \gamma(1 - \phi(s)) \odot \bar{\varphi}_{\theta^{(k)}}(s', a') \right]}_{\triangleq y(s,a)} \right) \right\}$$

    end for
  end for
  // Stage 2: Supervised learning
  Randomly initialize θ_0
  for $\ell = 1, \ldots, L$ **do**
    Randomly shuffle $\mathcal{D}$
    for $\{((s, a), y)\}_{b=1}^B \in \mathcal{D}$ **do**
      $\theta_\ell \leftarrow \theta_{\ell-1} - \alpha \nabla_\theta \frac{1}{2B} \sum_{b=1}^B \left(y_b - \mathbf{w}^\mathsf{T} \varphi_{\theta_{\ell-1}}(s_b, a_b) \right)^2$
    end for
  end for
  $\theta^{(k+1)} \leftarrow \theta_L$
end for

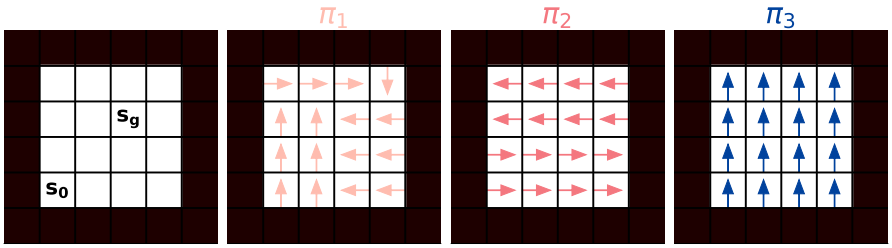

Figure E.1: **A simple grid and several policies.**

This regularizer encourages B to be approximately orthonormal, which promotes identifiability of F_θ and B_ω [12].

E Additional Results

See surrounding sections.

F Advantage of the Correct λ

Importantly, for GPE using the λR to work in this setting, the agent must either learn or be provided with the updated reward vector $\mathbf{r}_\lambda$ after each step/encounter with a rewarded state. This is because the λR is forward-looking in that it measures the (diminished) expected occupancies of states in the future without an explicit mechanism for remembering previous visits. For simplicity in this case, we provide this vector to the agent at each step—though if we view such a multitask agent as

Algorithm 3: λO FB Learning

Require: Probability distribution ν over $\mathbb{R}^d$, randomly initialized networks F_θ, B_ω, learning rate η, mini-batch size B, number of episodes E, number of epochs M, number of time steps per episode T, number of gradient steps N, regularization coefficient β, Polyak coefficient α, initial diminishing constant λ, discount factor γ, exploratory policy greediness ϵ, temperature τ
// Stage 1: Unsupervised learning phase
$\mathcal{D} \leftarrow \varnothing$
for epoch $m = 1, \ldots, M$ **do**
  for episode $i = 1 \ldots, E$ **do**
    Sample $z \sim \nu$
    Observe initial state s_1
    for $t = 1, \ldots, T$ **do**
      Select a_t $\epsilon-$greedy with respect to $F_\theta(s_t, a, z)^\top z$
      Observe reward $r_t(s_t)$ and next state s_{t+1}
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t(s_t), s_{t+1})\}$
    end for
  end for
  for $n = 1, \ldots, N$ **do**
    Sample a minibatch $\{(s_j, a_j, r_j(s_j), s_{j+1})\}_{j \in J} \subset \mathcal{D}$ of size $|J| = B$
    Sample a minibatch $\{\tilde{s}_j\}_{j \in J} \subset \mathcal{D}$ of size $|J| = B$
    Sample a minibatch $\{s'_j\}_{j \in J} \overset{\text{iid}}{\sim} \mu$ of size $|J| = B$
    Sample a minibatch $\{z_j\}_{j \in J} \overset{\text{iid}}{\sim} \nu$ of size $|J| = B$
    For every $j \in J$, set $\pi_{z_j}(\cdot|s_{j+1}) = \texttt{softmax}\left(F_{\theta^-}(s_{j+1}, \cdot, z_j)^\top z_j / \tau\right)$

$$\begin{aligned}\mathcal{L}(\theta, \omega) \leftarrow \frac{1}{2B^2} \sum_{j,k \in J^2} \Bigg(& F_\theta(s_j, a_j, z_j)^\top B_\omega(s'_k) - \gamma \sum_{a \in \mathcal{A}} \pi_{z_j}(a|s_{j+1}) F_{\theta^-}(s_{j+1}, a, z_j)^\top B_{\omega^-}(s'_k) \Bigg)^2 \\ & - \frac{1}{B} \sum_{j \in J} F_\theta(s_j, a_j, z_j)^\top B_\omega(s_j) \\ & + \frac{\gamma(1-\lambda)}{B} \sum_{j \in J} \mu(s_j) F_\theta(s_j, a_j, z_j)^\top B_\omega(s_j) \sum_{a \in \mathcal{A}} \pi_{z_j}(a|s_{j+1}) F_{\theta^-}(s_{j+1}, a, z_j)^\top B_{\omega^-}(s_j) \\ & + \beta \left(\frac{1}{B^2} \sum_{j,k \in J^2} B_\omega(s_j)^\top \bar{B}_\omega(\tilde{s}_k) \bar{B}_\omega(s_j)^\top \bar{B}_\omega(\tilde{s}_k) - \frac{1}{B} \sum_{j \in J} B_w(s_j)^\top \bar{B}_\omega(s_j) \right)\end{aligned}$$

    Update θ and ω via one step of `Adam` on $\mathcal{L}$
    Sample a minibatch $\{(s_j, r_j(s_j), s_{j+1}, r_{j+1}(s_{j+1}))\}_{j \in J}$ of size $|J| = B$ from $\mathcal{D}$
    $\mathcal{L}_\lambda(\lambda) \leftarrow \frac{1}{2B} \sum_{j \in J} \mathbb{1}(s_{j+1} = s_j) \left(r_{j+1}(s_{j+1}) - \lambda r_j(s_j)\right)^2$
    Update λ via one step of `Adam` on $\mathcal{L}_\lambda$
  end for
  $\theta^- \leftarrow \alpha\theta^- + (1-\alpha)\theta$
  $\omega^- \leftarrow \alpha\omega^- + (1-\alpha)\omega$
end for
// Stage 2: Exploitation phase for a single episode with initial reward $r_0(s)$
$z_R \leftarrow \sum_{s \in \mathcal{S}} \mu(s) r_0(s) B_\omega(s)$
Observe initial state s_1
for $t = 1, \ldots, T$ **do**
  $a_t \leftarrow \text{argmax}_{a \in \mathcal{A}} F(s_t, a, z_R)^\top z_R$
  Observe reward $r_t(s)$ and next state s_{t+1}
  $z_R \leftarrow \sum_{s \in \mathcal{S}} \mu(s) r_t(s) B_\omega(s)$
end for

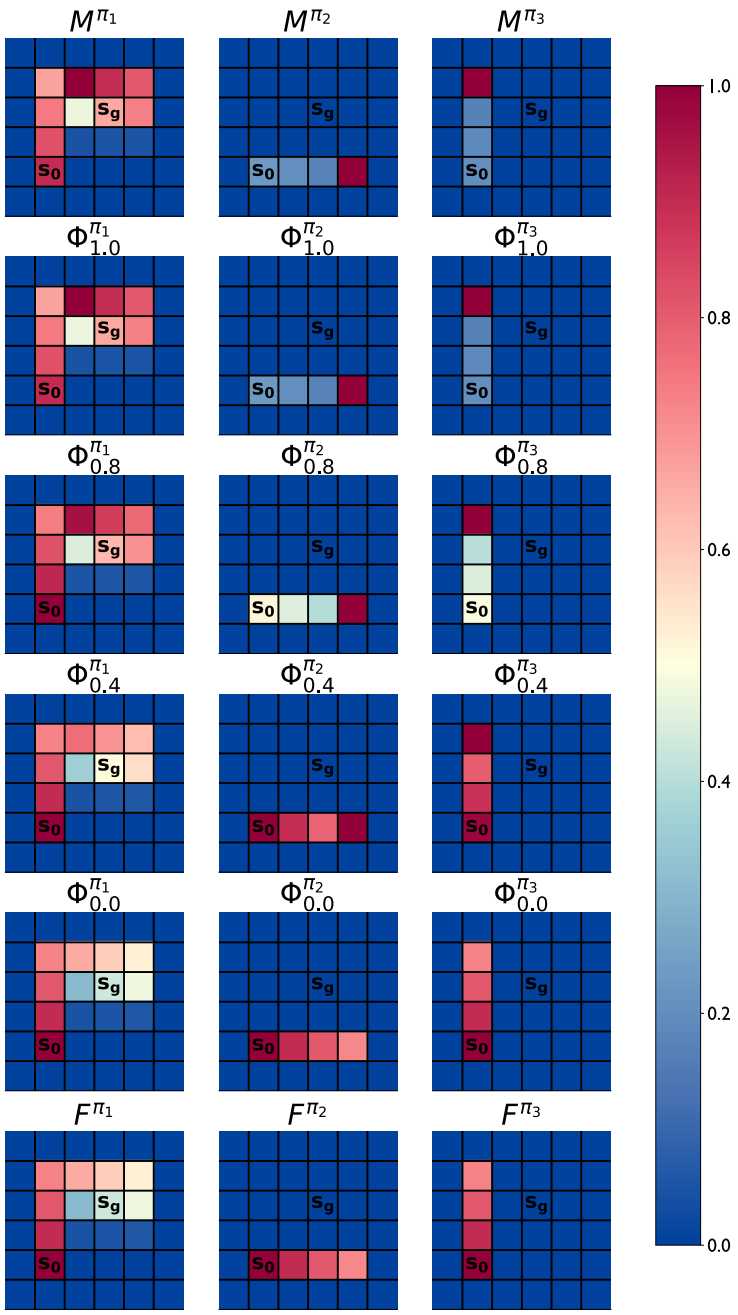

Figure E.2: **Visualizing the SR, the λR and the FR**. We can see that the Φ^{π}_{1} is equivalent to the SR and Φ^{π}_{0} is equivalent to the FR, with intermediate values of λ providing a smooth transition between the two.

simply as a module carrying out the directives of a higher-level module or policy within a hierarchical framework as in, e.g., Feudal RL [33], the explicit provision of reward information is not unrealistic. Regardless, a natural question in this case is whether there is actually any value in using the λR with the corret value of λ in this setting: If the agent is provided with the correct reward vector, then wouldn't policy evaluation work with any λR?

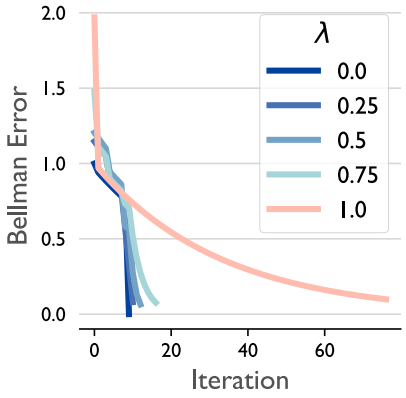

Figure E.3: **Dynamic programming converges more quickly for lower λ.**

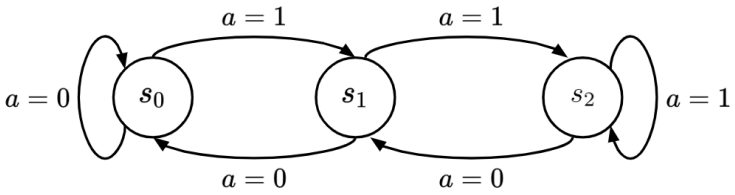

Figure F.1: **A 3-state toy environment**.

To see that this is not the case, consider the three-state toy MDP shown in Figure Fig. F.1, where $\bar{r}(s_1) = 10$, $\bar{r}(s_2) = 6$, $\bar{r}(s_0) = 0$, $\lambda(s_1) = 0$, $\lambda(s_2) = 1.0$, and $\gamma = 0.99$. At time $t = 0$, the agent starts in s_0. Performing policy evaluation with $\lambda(s_1) = \lambda(s_2) = 1$ (i.e., with the SR) would lead the agent to go left to s_1. However, the reward would then disappear, and policy evaluation on the second step would lead it to then move right to s_0 and then s_2, where it would stay for the remainder of the episode. In contrast, performing PI with the correct values of λ would lead the agent to go right to s_2 and stay there. In the first two timesteps, the first policy nets a total reward of $10 + 0 = 10$, while the second policy nets $6 + 5.94 = 11.94$. (The remaining decisions are identical between the two policies.) This is a clear example of the benefit of having the correct λ, as incorrect value estimation leads to suboptimal decisions even when the correct reward vector/function is provided at each step.

G The λ Operator

To learn the λO, we would like to define $\Phi^\pi_\lambda(s_t, ds') \triangleq \varphi^\pi_\lambda(s_t, s')\mu(ds')$ for some base policy μ. However, this would lead to a contradiction:

$$\Phi^\pi_\lambda(s, A \cup B) = \int_A \varphi^\pi_\lambda(s, ds')\mu(ds') + \int_B \varphi^\pi_\lambda(s, ds')\mu(ds') = \Phi^\pi_\lambda(s, A) + \Phi^\pi_\lambda(s, B)$$

for all disjoint A, B, contradicting Lemma B.3.

For now, we describe how to learn the λO for discrete $\mathcal{S}$, in which case we have $\Phi^\pi_\lambda(s, s') = \varphi^\pi_\lambda(s, s')\mu(s')$, i.e., by learning φ we learn a weighted version of Φ. We

define the following norm, inspired by Touati et al. [24]:

$$\|\Phi^\pi_\lambda\|^2_\rho \triangleq \mathbb{E}_{\substack{s\sim\rho\\ s'\sim\mu}}\left[\left(\frac{\Phi^\pi_\lambda(s,s')}{\mu(s')}\right)^2\right],$$

where μ is any density on $\mathcal{S}$. In the case of finite $\mathcal{S}$, we let μ be the uniform density. We then minimize the Bellman error for Φ^π_λ with respect to $\|\cdot\|^2_{\rho,\mu}$ (dropping the sub/superscripts on Φ and φ for clarity):

$$\begin{aligned}
\mathcal{L}(\Phi) &= \|\varphi\mu - (I \odot (\mathbf{1}\mathbf{1}^\mathsf{T} + \lambda\gamma P^\pi \varphi\mu) + \gamma(\mathbf{1}\mathbf{1}^\mathsf{T} + I) \odot P^\pi\varphi\mu)\|^2_{\rho,\mu} \\
&= \mathbb{E}_{s_t\sim\rho,s'\sim\mu}\Big[\Big(\varphi(s_t,s') - \frac{\mathbb{1}(s_t=s')}{\mu(s')} \\
&\qquad + \gamma(1-\lambda)\frac{\mathbb{1}(s_t=s')}{\mu(s')}\mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s_t)}\bar\Phi(s_{t+1},s') - \gamma\mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s_t)}\bar\varphi(s_{t+1},s')\Big)^2\Big] \\
&\overset{+c}{=} \mathbb{E}_{s_t,s_{t+1}\sim\rho,s'\sim\mu}\left[\left(\varphi(s_t,s') - \gamma\bar\varphi(s_{t+1},s')\right)^2\right] \\
&\qquad - 2\mathbb{E}_{s_t,s_{t+1}\sim\rho}\left[\sum_{s'}\mu(s')\varphi(s_t,s')\frac{\mathbb{1}(s_t=s')}{\mu(s')}\right] \\
&\qquad + 2\gamma(1-\lambda)\mathbb{E}_{s_t,s_{t+1}\sim\rho}\left[\sum_{s'}\mu(s')\varphi(s_t,s')\bar\varphi(s_{t+1},s')\mu(s')\frac{\mathbb{1}(s_t=s')}{\mu(s')}\right] \\
&\overset{+c}{=} \mathbb{E}_{s_t,s_{t+1}\sim\rho,s'\sim\mu}\left[\left(\varphi(s_t,s') - \gamma\bar\varphi(s_{t+1},s')\right)^2\right] - 2\mathbb{E}_{s_t\sim\rho}[\varphi(s_t,s_t)] \\
&\qquad + 2\gamma(1-\lambda)\mathbb{E}_{s_t,s_{t+1}\sim\rho}[\mu(s_t)\varphi(s_t,s_t)\bar\varphi(s_{t+1},s_t)],
\end{aligned}$$

Note that we recover the SM loss when $\lambda = 1$. Also, an interesting interpretation is that when the agent can never return to its previous state (i.e., $\varphi(s_{t+1},s_t) = 0$), then we also recover the SM loss, regardless of λ. In this way, the above loss appears to "correct" for repeated state visits so that the measure only reflects the first visit.

$$\begin{aligned}
\mathcal{L}(\Phi) = \mathbb{E}_{s_t,a_t,s_{t+1}\sim\rho,s'\sim\mu}&\left[\left(F(s_t,a_t,z)^\top B(s') - \gamma\bar F(s_{t+1},\pi_z(s_{t+1}),z)^\top\bar B(s')\right)^2\right] \\
&- 2\mathbb{E}_{s_t,a_t\sim\rho}\left[F(s_t,a_t,z)^\top B(s_t)\right] \\
&+ 2\gamma(1-\lambda)\mathbb{E}_{s_t,a_t,s_{t+1}\sim\rho}\left[\mu(s_t)F(s_t,a_t,z)^\top B(s_t)\bar F(s_{t+1},\pi_z(s_{t+1}),z)^\top\bar B(s_t)\right]
\end{aligned} \tag{G.1}$$

Even though the λO is not a measure, we can use the above loss to the continuous case, pretending as though we could take the Radon-Nikodym derivative $\frac{\Phi(s,ds')}{\mu(ds')}$.

G.1 Experimental Results with the FB Parameterization

To show that knowing the correct value of λ leads to improved performance, we trained λO with the FB parameterization on the FourRooms task of Fig. 5.3, but with each episode initialized at a random start state and with two random goal states. Average per-epoch reward is shown in Fig. G.2. We tested performance

Hyperparameter	Value
M	100
E	100
N	25
B	128
T	50
γ	0.99
α	0.95
η	0.001
τ	200
ϵ	1

Table 1: **λO-FB hyperparameters.**

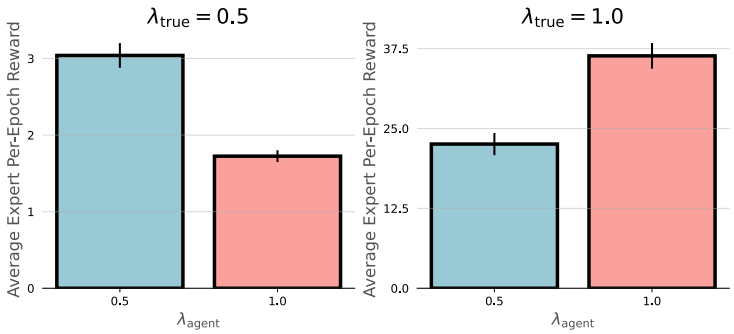

Figure G.1: **Performance of the λO-FB with two values of λ.** Results averaged over six seeds and 10 episodes per seed. Error bars indicate standard error.

with $\lambda_{\text{true}}, \lambda_{\text{agent}} \in \{0.5, 1.0\}$, where λ_{true} denotes the true environment diminishing rate and λ_{agent} denotes the diminishing rate that the agent uses. For the purpose of illustration, we do not allow the agent to learn λ. We see in Fig. G.2 that using the correct λ leads to significantly increased performance. In particular, the left plot shows that assuming $\lambda = 1$, i.e., using the SR, in a diminishing environment can lead to highly suboptimal performance.

Hyperparameters used are given in Table 1 (notation as in Algorithm 3).

G.2 λO and the Marginal Value Theorem

To study whether the agent's behavior is similar to behavior predicted by the MVT, we use a very simple task with constant starting state and vary the distance between rewards (see Fig. G.1(a)). When an agent is in a reward state, we define an MVT-optimal leaving time as follows (similar to that of [8] but accounting for the non-stationarity of the reward).

Let R denote the average per-episode reward received by a trained agent, $r(s_t)$ denote the reward received at time t in a given episode, $R_t = \sum_{u=0}^{t} r(s_u)$ denote the total reward received until time t in the episode, and let T be episode length. Then, on average, the agent should leave its current reward state at time t if the next reward that it would receive by staying in s_t, i.e., $\lambda r(s_t)$, is less than

$$\frac{R - R_t}{T}.$$

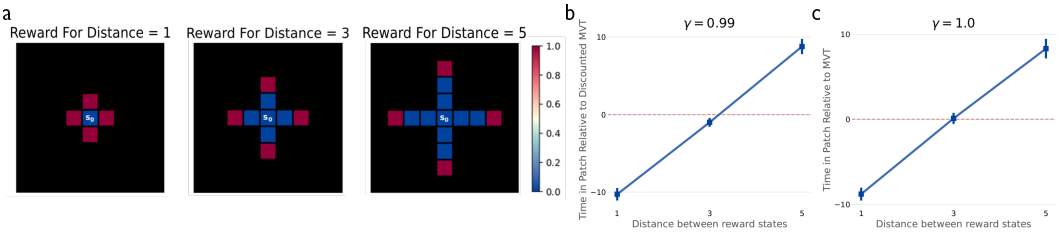

Figure G.2: **Analysis of MVT-like behavior of λO-FB. a)** Three environments with equal start state and structure but different distances between reward states. **b)** Difference between the agent's leave times and MVT-predicted leave times for $\gamma = 0.99$, with discounting taken into account. The agent on average behaves similar to the discounted MVT. **c)** Difference between the agent's leave times and MVT-predicted leave times for $\gamma = 1.0$, i.e., with no discounting taken into account. The agent on average behaves similar to the MVT.

In other words, the agent should leave a reward state when its incoming reward falls below the diminished average per-step reward of the environment. We compute R by averaging reward received by a trained agent over many episodes.

Previous studies have trained agents that assume stationary reward to perform foraging tasks, even though the reward in these tasks is non-stationary. These agents can still achieve good performance and MVT-like behavior [8]. However, because they target the standard RL objective

$$\mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}) \middle| s_t = s \right],$$

which requires $\gamma < 1$ for convergence, optimal behavior is recovered only with respect to the *discounted MVT*, in which R (and in our case, R_t) weights rewards by powers of γ [8].

In Fig. G.1(b-c) we perform a similar analysis to that of [8] and show that, on average over multiple distances between rewards, λO-FB performs similarly to the discounted MVT for $\gamma = 0.99$ and the standard MVT for $\gamma = 1.0$. An advantage of the λO is that it is finite for $\gamma = 1.0$ provided that $\lambda < 1$. Hence, as opposed to previous work, we can recover the standard MVT without the need to adjust for discounting.

Hyperparameters used are given in Table 1 (notation as in Algorithm 3).

H SAC

Mitigating Value Overestimation One well-known challenge in deep RL is that the use of function approximation to compute values is prone to overestimation. Standard approaches to mitigate this issue typically do so by using *two* value functions and either taking the minimum $\min_{i\in\{1,2\}} Q_i^\pi(s,a)$ to form the Bellman target for a given (s,a) pair [34] or combining them in other ways [35]. However, creating multiple networks is expensive in both computation and memory. Instead, we hypothesized that it might be possible to address this issue by using λ-based values. To test this idea, we modified the Soft Actor-Critic [SAC; 36] algorithm to compute λFs-based values by augmenting the soft value target $\mathcal{T}_{soft}Q = r_t + \gamma \mathbb{E} V_{soft}(s_{t+1})$, where

$V_{soft}(s_{t+1})$ is given by the expression

$$\mathbb{E}_{a_{t+1}\sim\pi(\cdot|s_{t+1})}\Big[\bar{Q}(s_{t+1},a_{t+1}) + (\lambda-1)\mathbf{w}^\mathsf{T}(\phi(s_t,a_t)\odot\varphi_\lambda(s_{t+1},a_{t+1})) \\ - \alpha\log\pi(a_{t+1}\mid s_{t+1})\Big]$$

A derivation as well as pseudocode for the modified loss is provided in Appendix D.5. Observe that for $\lambda = 1$, we recover the standard SAC value target, corresponding to an assumed stationary reward. We apply this modified SAC algorithm, which we term λ-SAC to feature-based Mujoco continuous control tasks within OpenAI Gym [37]. We found that concatenating the raw state and action observations $\tilde{\phi}_t = [s_t, a_t]$ and normalizing them to $[0, 1]$ make effective regressors to the reward. That is, we compute base features as

$$\phi_t^b = \frac{\tilde{\phi}_t^b - \min_b \tilde{\phi}_t^b}{\max_b \tilde{\phi}_t^b - \min_b \tilde{\phi}_t^b},$$

where b indexes (s_t, a_t) within a batch. Let $X \in [0,1]^{B\times D}$ be the concatenated matrix of features for a batch, where $D = \dim(\mathcal{S}) + \dim(\mathcal{A})$. Then,

$$\mathbf{w}_t = \left(X^\mathsf{T}X\right)^{-1}X^\mathsf{T}\mathbf{r},$$

where here $\mathbf{r}$ denotes the vector of rewards from the batch. In addition to using a fixed λ value, ideally we'd like an agent to adaptively update λ to achieve the best balance of optimism and pessimism in its value estimates. Following [38], we frame this decision as a multi-armed bandit problem, discretizing λ into three possible values $\{0, 0.5, 1.0\}$ representing the arms of the bandit. At the start of each episode, a random value of λ is sampled from these arms and used in the value function update. The probability of each arm is updated using the Exponentially Weighted Average Forecasting algorithm [39], which modulates the probabilities in proportion to a feedback score. As in [38], we use the difference in cumulative (undiscounted) reward between the current episode ℓ and the previous one $\ell - 1$ as this feedback signal: $R_\ell - R_{\ell-1}$. That is, the probability of selecting a given value of λ increases if performance is improving and decreases if it's decreasing. We use identical settings for the bandit algorithm as in [38]. We call this variant λ-SAC. We plot the results for SAC with two critics (as is standard), SAC with one critic, SAC with a single critic trained with λF-based values ("x-SAC" denotes SAC trained with a fixed $\lambda = x$), and λ-SAC trained on the HalfCheetah-v2 Mujoco environment. This task was found by [38] to support "optimistic" value estimates in that even without pessimism to reduce overestimation it was possible to perform well. Consistent with this, we found that single-critic SAC matched the performance of standard SAC, as did 1-SAC (which amounts to training a standard value function with the auxiliary task of SF prediction). Fixing lower values of λ performed poorly, indicating that over-pessimism is harmful in this environment. However, λ-SAC eventually manages to learn to set $\lambda = 1$ and matches the final performance of the best fixed algorithms. We consider these results to be very preliminary, and hope to perform more experiments on other environments. We also believe λ-SAC could be improved by using the difference between the current episode's total reward and the *average* of the total rewards from previous episodes $R_\ell - (\ell-1)^{-1}\sum_{i=1}^{\ell-1} R_i$ as a

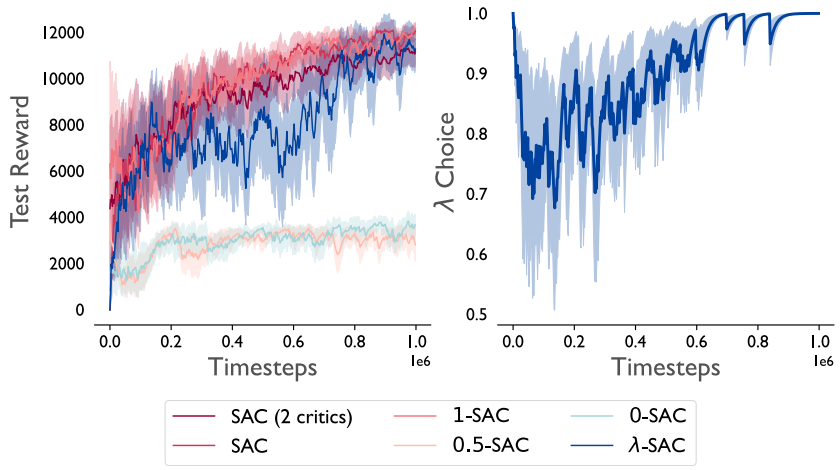

Figure H.1: **λ-SAC adaptively identifies the optimal λ.** Rewards are measure on the HalfCheetah task averaged over three random seeds, with shading indicating one unit of standard error.

more stable feedback signal for the bandit. There is also non-stationarity in the base features due to the per-batch normalization, which could also likely be improved. Hyperparameters are described in Table 2.

Hyperparameter	**Value**
Collection Steps	1000
Random Action Steps	10000
Network Hidden Layers	256:256
Learning Rate	3×10^{-4}
Optimizer	Adam
Replay Buffer Size	1×10^{6}
Action Limit	$[-1, 1]$
Exponential Moving Avg. Parameters	5×10^{-3}
(Critic Update:Environment Step) Ratio	1
(Policy Update:Environment Step) Ratio	1
Has Target Policy?	No
Expected Entropy Target	$-\dim(\mathcal{A})$
Policy Log-Variance Limits	$[-20, 2]$
Bandit Learning Rate*	0.1
λ Options*	$\{0, 0.5, 1.0\}$

Table 2: Hyperparameters for SAC Mujoco experiments. *Only applicable to λ-SAC

I Replenishing Rewards

We list below a few candidate reward replenishment schemes, which are visualized in Fig. I.1.

Time elapsed rewards

$$r(s,t) = \lambda^{n(s,t)/m(s,t)} \bar{r}(s),$$

where $m(s,t)$ is the time elapsed since the last visit to state s:

$$m(s,t) \triangleq t - \max\{j | s_{t+j} = s,\ 0 \leq j \leq t-1\}.$$

Due to the $\max$ term in $m(s,t)$, the corresponding representation

$$\mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k\lambda^{n(s,t)/m(s,t)}\mathbb{1}(s_{t+k}=s')\middle|s_t=s\right]$$

does not admit a closed-form recursion. However, we empirically tested a version of this type of reward with Q_λ-learning in the TwoRooms environment, modified so that the exponent on λ is $n(s,t)/(0.1m(s,t))$. This was done so that reward replenishes at a slow rate, reducing the deviation from the standard diminishing setting. Episode length was capped at $H=100$. All other settings are identical to the Q_λ experiment described in Appendix D. Results are presented in Fig. I.2 and a GIF is included in the supplementary material.

Eligibility trace rewards

$$r(s,t)=\left(1-(1-\lambda_d)\sum_{j=0}^{t-1}\lambda_r^{t-j}\mathbb{1}(s_{t+j}=s)\right)\bar{r}(s),$$

where $\lambda_d,\lambda_r\in[0,1]$ are diminishment and replenishment constants, respectively. Denoting the corresponding representation by Ω^π, i.e.,

$$\Omega^\pi(s,s')=\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k\left(1-(1-\lambda_d)\sum_{j=0}^{k}\lambda_r^{k-j}\mathbb{1}(s_{t+j}=s')\right)\mathbb{1}(s_{t+k}=s')\middle|s_t=s\right],$$

we obtain the following recursion:

$$\begin{aligned}\Omega^\pi(s,s')=\mathbb{1}(s=s')\,(\lambda_d-\gamma\lambda_r(1-\lambda_d)\,&\mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s)}M^\pi_{\gamma\lambda_r}(s_{t+1},s')\\&+\gamma\mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s)}\Omega^\pi(s_{t+1},s'),\end{aligned}$$

where $M^\pi_{\gamma\lambda_r}$ denotes the successor representation of π with discount factor $\gamma\lambda_r$. This representation could be learned by alternating TD learning between Ω^π and $M^\pi_{\gamma\lambda_r}$. We leave this to future work.

Total time rewards

$$r(s,t)=\lambda_d^{n(s,t)}\lambda_r^{n(s,t)-t}\bar{r}(s),$$

where $\lambda_d,\lambda_r\in[0,1]$ are diminishment and replenishment constants, respectively. The corresponding representation is

$$P^\pi(s,s')=\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^k\lambda_d^{n_t(s',k)}\lambda_r^{k-n_t(s',k)}\mathbb{1}(s_{t+k}=s')\middle|s_t=s\right],$$

which satisfies the following recursion:

$$\begin{aligned}P^\pi(s,s')=\mathbb{1}(s=s')+\gamma(\lambda_d\mathbb{1}(s=s')+1\\-\mathbb{1}(s=s'))(\lambda_r(1-\mathbb{1}(s=s'))+\mathbb{1}(s=s'))\mathbb{E}_{s_{t+1}\sim p^\pi(\cdot|s)}P^\pi(s_{t+1},s').\end{aligned}$$

While neither the reward nor the representation are guaranteed to be finite, P^π could be learned via TD updates capped at a suitable upper bound.

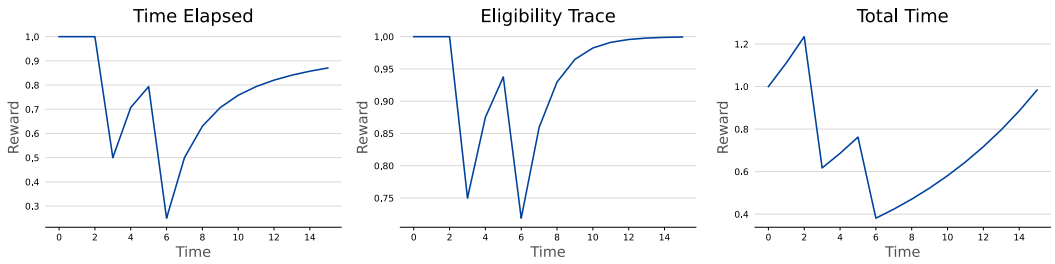

Figure I.1: **Visualizing three different replenishment schemes.** For all schemes, $\bar{r}(s) = 1$ and visits to s are at $t = 2, 5$. (Left) The time elapsed reward with $\lambda = 0.5$; (Middle) The eligibility trace reward with $\lambda_r = \lambda_d = 0.5$; (Right) The total time reward with $\lambda_d = 0.5, \lambda_r = 0.9$.

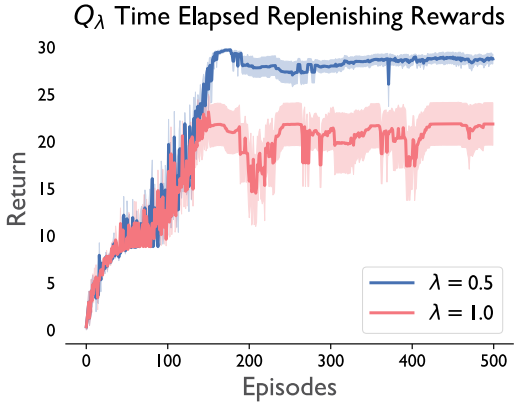

Figure I.2: **Performance on TwoRooms with replenishing rewards.** Return is averaged over five runs, with shading indicating one unit of standard error.

J λ vs. γ

We now briefly discuss the interaction between the temporal discount factor γ commonly used in RL and the diminishing utility rate λ. The key distinction between the two is that all rewards decay in value every time step with respect to γ, regardless of whether a state is visited or not. With λ, however, decay is specific to each state (or (s, a) pair) and only occurs when the agent visits that state. Thus, γ decays reward in a global manner which is independent of the agent's behavior, and λ decays reward in a local manner which dependent on the agent's behavior. In combination, they have the beneficial effect of accelerating convergence in dynamic programming (**??**). This indicates the potential for the use of higher discount factors in practice, as paired with a decay factor λ, similar (or faster) convergence rates could be observed even as agents are able to act with a longer effective temporal horizon.

K Compute Resources

The λF-based experiments shown were run on a single NVIDIA GeForce GTX 1080 GPU. The recurrent A2C experiments took roughly 30 minutes, base feature learning for policy composition took approximately 45 minutes, λF learning for policy composition took approximately 10 hours, and the SAC experiments took

approximately 8 hours per run. The λF training required roughly 30GB of memory due to the size of the dataset. All experiments in Section 6 and Appendix G were run on a single RTX5000 GPU and each training and evaluation run took about 30 minutes. All other experiments were run on a 2020 MacBook Air laptop 1.1 GHz Quad-Core Intel Core i5 CPU and took less than one hour to train.

L Broader Impact Statement

We consider this work to be primarily of a theoretical nature pertaining to sequential decision-making primarily in the context of natural intelligence. While it may have applications for more efficient training of artificial RL agents, it is hard to predict long-term societal impacts.

References

[1] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. Econometrica, 47(2):263–292, 1979.

[2] Matthew Rabin. Risk aversion and expected-utility theory: a calibration theorem. Econometrica, 68(5):1281–1292, 2000. doi: 10.2307/2999450.

[3] Alex Pine, Ben Seymour, Jonathan P Roiser, Peter Bossaerts, Karl J Friston, H Valerie Curran, and Raymond J Dolan. Encoding of marginal utility across time in the human brain. Journal of Neuroscience, 29(30):9575–9581, 2009.

[4] Martin L. Puterman. Markov decision processes: discrete stochastic dynamic programming. John Wiley and Sons, 2010.

[5] Hermann Heinrich Gossen and Rudolph C Blitz. The laws of human relations and the rules of human action derived therefrom. Mit Press Cambridge, MA, 1983.

[6] Kenneth J Arrow. The theory of risk-bearing: small and great risks. Journal of risk and uncertainty, 12:103–111, 1996.

[7] John W Pratt. Risk aversion in the small and in the large. In Uncertainty in economics, pages 59–79. Elsevier, 1978.

[8] Nathan Wispinski, Andrew Butcher, Kory Wallace Mathewson, Craig S Chapman, Matthew Botvinick, and Patrick M. Pilarski. Adaptive patch foraging in deep reinforcement learning agents. Transactions on Machine Learning Research, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=a0T3nOP9sB.

[9] Sergey Shuvaev, Sarah Starosta, Duda Kvitsiani, Adam Kepecs, and Alexei Koulakov. R-learning in actor-critic model offers a biologically relevant mechanism for sequential decision-making. Advances in neural information processing systems, 33:18872–18882, 2020.

[10] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. Neural Computation, 5(4):613–624, 1993. doi: 10.1162/neco.1993.5.4.613.

[11] Ted Moskovitz, Spencer R Wilson, and Maneesh Sahani. A first-occupancy representation for reinforcement learning. In International Conference on Learning Representations, 2022. URL https://openreview.net/forum?id=JBAZe2yN6Ub.

[12] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. Advances in Neural Information Processing Systems, 34:13–23, 2021.

[13] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[14] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. Nature neuroscience, 20(11):1643–1653, 2017.

[15] Andre Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. Proceedings of the National Academy of Sciences, 117(48):30079–30087, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907370117. URL https://www.pnas.org/content/117/48/30079.

[16] 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. Advances in neural information processing systems, 30, 2017.

[17] André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Žídek, and Rémi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement, 2019.

[18] Léonard Blier and Yann Ollivier. The description length of deep learning models. Advances in Neural Information Processing Systems, 31, 2018.

[19] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8:279–292, 1992.

[20] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pages 1928–1937. PMLR, 2016.

[21] Arthur Juliani, Samuel Barnett, Brandon Davis, Margaret Sereno, and Ida Momennejad. Neuronav: a library for neurally-plausible reinforcement learning. arXiv preprint arXiv:2206.03312, 2022.

[22] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural Computation, 9 (8):1735–1780, 1997.

[23] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence, 112 (1):181–211, 1999. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

[24] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In The Eleventh International Conference on Learning Representations, 2023. URL https://openreview.net/forum?id=MYEap_OcQI.

[25] Benjamin Y Hayden, John M Pearson, and Michael L Platt. Neuronal basis of sequential foraging decisions in a patchy environment. Nature neuroscience, 14(7):933–939, 2011.

[26] Tehrim Yoon, Robert B Geary, Alaa A Ahmed, and Reza Shadmehr. Control of movement vigor and decision making during foraging. Proceedings of the National Academy of Sciences, 115 (44):E10476–E10485, 2018.

[27] Anthony S Gabay and Matthew AJ Apps. Foraging optimally in social neuroscience: computations and methodological considerations. Social Cognitive and Affective Neuroscience, 16(8): 782–794, 2021.

[28] Eric L. Charnov. Optimal foraging, the marginal value theorem. Theoretical Population Biology, 9(2):129–136, 1976. ISSN 0040-5809.

[29] Peter Nonacs. State dependent behavior and the marginal value theorem. Behavioral Ecology, 12(1):71–83, 2001.

[30] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvari, Satinder Singh, et al. Behaviour suite for reinforcement learning. arXiv preprint arXiv:1908.03568, 2019.

[31] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

[32] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16, pages 317–328. Springer, 2005.

[33] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. Advances in neural information processing systems, 5, 1992.

[34] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018, 2018.

[35] Ted Moskovitz, Michael Arbel, Ferenc Huszar, and Arthur Gretton. Efficient wasserstein natural gradients for reinforcement learning, 2020.

[36] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[37] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[38] Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical optimism and pessimism for deep reinforcement learning. Advances in Neural Information Processing Systems, 34:12849–12863, 2021.

[39] Nicolo Cesa-Bianchi and Gábor Lugosi. Prediction, learning, and games. Cambridge university press, 2006.