EVAL: EigenVector-based Average-reward Learning

Anonymous Author(s)
Submission Id: 1171

ABSTRACT

In reinforcement learning, two objective functions have been developed extensively in the literature: discounted and averaged rewards. The generalization to an entropy-regularized setting has led to improved robustness and exploration for both of these objectives. Recently, the entropy-regularized average-reward problem was addressed using tools from large deviation theory in the tabular setting. This method has the advantage of linearity, providing access to both the optimal policy and average reward-rate through properties of a single matrix. In this paper, we extend that framework to more general settings by developing approaches based on function approximation by neural networks. This formulation reveals new theoretical insights into the relationship between different objectives used in RL. Additionally, we combine our algorithm with a posterior policy iteration scheme, showing how our approach can also solve the average-reward RL problem without entropy-regularization. Using classic control benchmarks, we experimentally find that our method compares favorably with other algorithms in terms of stability and rate of convergence.

ACM Reference Format:
Anonymous Author(s). 2025. EVAL: EigenVector-based Average-reward Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 13 pages.

1 INTRODUCTION

To solve the central problem of reinforcement learning, an agent continuously and autonomously interacts with its surroundings to maximize a long-term reward signal. The standard method of solving reinforcement learning (RL) tasks involves a discounted objective function: the agent seeks to maximize an infinite discounted sum of rewards, as expected under a chosen control policy. This geometrically discounted sum ensures a convergent objective function, making it a convenient representation of the problem to be solved. The discounted RL literature is vast in both theoretical [11, 33, 57] and algorithmic [6, 54] studies and has demonstrated great success in real-world problems of interest [28, 31, 41, 53, 55]. However, in many RL problems, this use of discounting is simply a useful proxy for the solving the true objective function: maximization of total episode reward. As a result, the actual choice of the discount factor, γ, is unphysical, not grounded in any corresponding physical timescale. As such, it is treated as a hyperparameter which is often tuned over for best performance (or set to some fixed value, e.g. $\gamma = 0.99$, for simplicity). There is a precedent in past work for rejecting the discounted framework [42], for example the Heaven and Hell MDP [56], suggesting that in many tasks the use of discounting is not only unphysical, but can lead to disastrous outcomes. Thus in long-term ("continuing") decision-making settings, another objective is needed.

An alternative approach to ensuring convergence of rewards across infinitely-long trajectories is to instead use the **average-reward** objective function [38]. Despite offering a principled alternative to the long-term optimization problem, the average reward framework has not been historically popular in the RL literature, perhaps due to the lack of successful algorithms in this setting. However, recent work has developed new algorithms and theory [8, 37, 43, 51, 62, 67] specifically for the average-reward objective. This prior work has focused on the tabular setting or on using policy-gradient techniques to develop new algorithms. Although these methods have proven useful in their respective domains, there remain gaps in the field that can be addressed with new approaches.

One such gap in the field of average-reward algorithms is the lack of entropy-regularized objectives, which have become a cornerstone of state-of-the-art algorithms in discounted RL [26, 28, 53, 55]. Formally, entropy-regularization involves including an "information" cost for deviating from a pre-specified (e.g. prior, guide, or behavioral) policy. This entropy cost is weighted with an inverse temperature parameter, β. This entropy-based regularization, in combination with treating the rewards as negative energies, reveals a close connection to statistical mechanics [8, 49]. Furthermore, including an entropy regularization term in the RL objective leads to more robust solutions (non-greedy optimal policies) and has theoretically been shown to be more adaptable to changes in reward and transition dynamics [21]. Preventing the optimal policy from collapsing to a deterministic function can ensure additional exploration occurs during training and deployment, often leading to faster learning [5].

In this paper, we introduce a novel algorithm for average-reward RL with entropy regularization: EigenVector-based Average-reward Learning (EVAL). Figure 1 illustrates a simple experiment highlighting one benefit of the EVAL approach to entropy-regularized RL compared to the discounted approach [27]. When the RL problem is posed in the discounted framework, a discount factor γ is a required input parameter. However, there is often no principled approach for choosing the value of γ corresponding to the specific problem being addressed. Thus, the experimenter must treat γ as a hyperparameter. This reduces the choice of γ to a trade-off between large values to capture long-term rewards and small values to capture computational efficiency which typically scales polynomially with the horizon, $(1-\gamma)^{-1}$ [33]. The horizon introduces a natural timescale to the problem, but this timescale may not be well-aligned with the timescale corresponding to the optimal dynamics: the mixing time of the induced Markov chain. For the discounted approach to accurately estimate the optimal policy, the discounting timescale (horizon) must be larger than the mixing time. However, estimating the mixing time for the optimal dynamics can be challenging in the general case, even when the transition dynamics are known.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), A. El Fallah Seghrouchni, Y. Vorobeychik, S. Das, A. Nowe (eds.), May 19 – 23, 2025, Detroit, Michigan, USA.

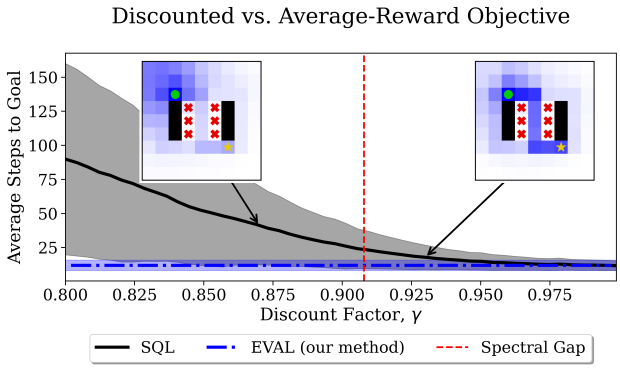

Figure 1: Performance of discounted soft Q-Learning (SQL) as a function of discount factor compared with solution using the proposed average-reward method (EVAL). Note that the average-reward solution (blue line) recovers the discounted solution as $\gamma \to 1$. For the discounted objective, computational cost grows as $(1-\gamma)^{-1}$ and choosing a low discount factor to reduce computational cost can result in lower rewards. The boundary of low reward, low complexity and high reward, high complexity is demarcated by the discount factor derived from the spectral gap of the associated tilted matrix (cf. Sec. 2). Insets: state-occupation distributions following the SQL optimal policies at $\gamma = 0.87, 0.93$. Green dot denotes initial position of agent, and star denotes the goal. The agent can move in any of the cardinal directions. Since we use entropy-regularization, the optimal policy is stochastic, thus yielding a variance in the return (plotted with a shaded interval for each method). Inverse temperature $\beta = 15$.

Therefore, an arbitrary "sufficiently large" choice of γ is often made without knowledge of the relevant problem-dependent timescale. This can be problematic from a computational standpoint as evidenced by recent work [6, 32, 55].

The approach outlined in this work for average-reward RL (EVAL) has the added benefit that it can lead to an estimation of the mixing timescale for the optimal dynamics, which can then be used to inform the choice of a discount factor, if the discounted objective is of interest. In Figure 1, the discount factor set by the spectral gap of the tilted matrix (see Section 4 for definition) is indicated by the vertical line. We empirically find that this "spectral gap discount factor" naturally separates discount factors between "small" and "large" values, seen by the distinct change in the optimal state distributions (inset). The discount factor set by the spectral gap indicates a point of diminishing return for further increasing γ in soft Q-Learning (solid black line). For comparison, we plot the return given by our average-reward algorithm (dashed blue line). Importantly, we see that the average-reward solution recovers the discounted solution in the $\gamma \to 1$ limit, as expected [12, 38].

Despite the desirable features of both the average-reward and entropy-regularized objectives, the combination of these formulations [44] is not as well-studied, and no function approximator algorithms exist for this setting. Here, we extend the ideas introduced in [9, 48] to develop a new off-policy learning-based approach to find the optimal policy of average-reward MDPs. Our main contributions are as follows:

Main Contributions

- We provide a novel off-policy solution to average-reward RL in both entropy-regularized and un-regularized settings with general function approximators.
- We provide theory for using average-rewards with reducible dynamics, a common setting across RL environments.
- We demonstrate experimentally the advantage of our approach against standard baselines in classic control environments.

Notably, our implementation requires minimal changes relative to the common DQN setup, making it accessible for researchers and allowing for multiple future extensions.

2 PRELIMINARIES

Reinforcement learning treats decision-making problems under the framework of Markov Decision Processes (MDPs). In this section, we provide a brief account of the problem to be solved (the entropy-regularized average-reward) objective under this framework. As a point of notation, we will denote the set of distributions over a generic space $\mathcal{X}$ as $\Delta(\mathcal{X})$. To begin, we describe the defining quantities of an MDP: the state space $\mathcal{S}$ denotes all possible configurations of the agent within its environment; the initial state distribution $\mu \in \Delta(\mathcal{S})$ describes the initialization of the agent at each episode; the action space $\mathcal{A}$ denotes the set of allowed controls the agent may exert to affect its environment; the transition function (also "dynamics") is a function $p : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ describing the probability of transitioning into a future state given a current state and action; and the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a real-valued scalar provided to the agent at each time-step, encoding the desired behavior of the agent. In the following equations, we focus on the discrete state-action case for ease of notation. The applicability of the framework for continuous spaces is demonstrated empirically below.

Now, we state some of the usual assumptions for average-reward MDPs:

Assumption 1. The Markov chain induced by the dynamics p and any stationary policy π is irreducible and aperiodic.

Assumption 2. The reward function is upper bounded. [1]

As in the discounted case, one seeks a control policy π which maximizes the expected return. In the discounted formulation, this objective is defined as a discounted infinite sum; but in the average-reward formulation, we instead consider the limiting value of the average trajectory reward, for increasingly long trajectories:

$$\sum_{t=0}^{\infty} \gamma^t r_t \to \lim_{N\to\infty} \frac{1}{N} \sum_{t=0}^{N-1} r_t \tag{1}$$

[1] In the following we assume the reward is non-positive, which is equivalent to $r(s,a)$ having a finite upper-bound: the reward function can always be shifted to be entirely negative without affecting the optimal policy, merely shifting the reward rate accordingly.

This *reward rate*, ρ^π, becomes the new objective:

$$\rho^\pi = \lim_{N\to\infty} \frac{1}{N} \mathop{\mathbb{E}}_{\tau\sim\pi,p,\mu} \left[\sum_{t=0}^{N-1} r(s_t, a_t)\right] = \mathop{\mathbb{E}}_{\nu} [r(s,a)], \tag{2}$$

with the expectation above taken with respect to the probability of a trajectory as generated by the dynamics p, fixed policy π, and initial state distribution μ. In the rightmost expression, ν is used to denote the stationary state-action distribution induced by the transition dynamics and choice of control policy. Because of Assumption 1, this distribution is well-defined as the eigenvector of the stochastic transition matrix P^π:

$$\sum_{s,a} p(s'|s,a)\pi(a'|s')\nu(s,a) = \nu(s',a'), \tag{3}$$

or in a more compact notation, $P^\pi \nu = \nu$. Related eigenvector equations will be our primary concern in approaching the entropy-regularized average-reward objective.

With the scalar reward-rate ("bias") defined, we now turn to the corresponding "differential" value function, which plays the role of the standard Q function in discounted RL. Specifically, the (s,a)-dependent contribution to a trajectory's value is denoted as Q^π_ρ.

$$Q^\pi_\rho(s,a) = \mathop{\mathbb{E}}_{\tau\sim p,\pi} \left[\sum_{t=0}^{\infty} r(s_t,a_t) - \rho^\pi \middle| s_0 = s, a_0 = a\right] \tag{4}$$

We now consider a more general version of this MDP which includes an entropy regularization term. Note that the original objective can be recovered in the zero temperature limit ($\beta \to \infty$). For convenience we will refer to entropy-regularized average-reward MDPs as ERAR MDPs. The ERAR MDP possesses the same components as an average-reward MDP as described, in addition to a pre-specified prior policy[2] $\pi_0 : \mathcal{S} \to \Delta(\mathcal{A})$ and "inverse temperature", $\beta \in \mathbb{R}_{>0}$. The modified objective function for an ERAR MDP now includes a relative entropy based regularization term, such that the agent now aims to optimize the expected *entropy-regularized reward-rate*, denoted θ^π below:

$$\theta^\pi = \lim_{N\to\infty} \frac{1}{N} \mathop{\mathbb{E}}_{\tau\sim p,\pi,\mu} \left[\sum_{t=0}^{N-1} r(s_t,a_t) - \frac{1}{\beta}\log\frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)}\right]. \tag{5}$$

The optimal policy for an ERAR MDP is defined analogously as the solution to

$$\pi^*(a|s) = \operatorname*{argmax}_\pi \theta^\pi. \tag{6}$$

From the above, we see that the agent must balance the returns given by directly optimizing the reward with the cost of deviating from the prior policy π_0. This also emphasizes the dependence of π_0 on the resulting optimal policy. In most work on entropy-regularized RL, the uniform distribution $\pi_0(a|s) \propto 1$ is used to express the prior belief that all actions are equally likely (preferable). However, this "maximum entropy" (MaxEnt) objective has a lack of flexibility, not allowing non-trivial priors or "guide" policies to be exploited, despite its apparent theoretical and experimental advantages [3, 5, 24, 64, 65]. In the following, we show that the more general non-uniform prior π_0 plays a crucial role in solving the un-regularized problem.

[2]We assume that π_0 has full support across $\mathcal{A}$, ensuring the Kullback-Liebler divergence between any policy π and π_0 remains finite.

Our Assumption 1 guarantees the expression in Equation (5) is indeed independent of the initial state-action. If on the other hand the induced Markov chain were reducible, there may be some recurrent classes whose long-term reward rate depends on the initial state and action.

In the following, for convenience, we will simply write $\theta = \theta^{\pi^*}$ for the optimal entropy-regularized reward-rate. Now, in comparing to Equation (2), we see that the present objective, the "ERAR rate", includes an entropic contribution: the relative entropy between the control and prior policies: $\mathrm{KL}(\pi|\pi_0)$, where KL denotes the Kullback-Liebler divergence.

Corresponding to Eq. (4), the differential entropy-regularized action-value function is then given by [3]

$$Q^\pi_\theta(s,a) = \mathop{\mathbb{E}}_{\tau\sim p,\pi} \left[\sum_{t=0}^{\infty} r(s_t,a_t) - \frac{1}{\beta}\log\frac{\pi(a_t|s_t)}{\pi_0(a_t|s_t)} - \theta^\pi\right] \tag{7}$$

In this section, θ and ρ subscripts were used to distinguish the two (un-regularized and regularized) differential value functions. In the following sections, we omit the subscript as we focus solely on the entropy-regularized objective. Similarly, we use the shorthand $Q(s,a) = Q^{\pi^*}_\theta(s,a)$.

3 PRIOR WORK

Classical approaches to the average-reward problem typically involve the following logic: Solve the discounted MDP for discount factors $\gamma_1 < \gamma_2 \cdots < 1$. As the value of the discount factor approaches 1, the (appropriately re-scaled) optimal value function will approach that of the average-reward MDP [12]. Later work directly approaches the average-reward objective by attacking the associated Bellman equation head-on [38]. Indeed, [56] introduced *R-learning* (without proof of convergence), emphasizing the benefits of the average-reward algorithm: faster reward propagation, better value disambiguation, simpler learning dynamics near initialization, and potentially speedups. Schwartz [56] also mentions how the discounted framework can be subsumed in the more general average-reward problem (Sec. 6.5 therein).

More recently, a variety of papers have considered the average-reward problem in a modern light, beginning to understand the theoretical properties of this objective and associated algorithms: [1, 2, 19, 44, 62, 66]. We note this set of prior works addresses the average-reward objective in the case of tabular settings, with discrete, finite state and action spaces. To address the continuous setting, recent work has developed [37, 50, 67] a variety of policy-based methods. In contrast, we will focus on the value-based setting here, learning the Q function directly in an off-policy manner. It is worth noting that these studies have found average-reward algorithms to be superior to discounted methods in continuous control Mujoco [60] tasks. A more complete description of the history of average-reward algorithms can be found in a recent survey: [18].

Alongside the average-reward literature, the discounted objective has seen the introduction of many techniques to generally improve sample complexity, by tackling planning [30], estimation bias[61], and exploration [45]. Prior work has shown that the addition of an entropy-regularization term can improve robustness [21],

[3]we have suppressed the conditioning on the initial state and action for brevity, but the conditioning is identical to that of Eq. (4).

exploration [20], composability [25] and sample complexity [28]. The theory of entropy regularization also has a long history, discussed in a series of work such as [23, 28, 48, 58, 59, 68]. This innovation yields controllably stochastic optimal policies, a flexibility which has led Soft Actor-Critic [29] and its variants to become state-of-the-art solution methods for addressing the discounted objective. To date, there are no direct combinations of the average-reward and entropy-regularized objectives for deep reinforcement learning. However, recent work by [8, 48, 49] has established a framework for combining these formulations. In this work, we leverage the results derived in combination with function approximation techniques to establish value-based algorithms for the average-reward objective. In the following section we give an overview of the solution which builds on results obtained in [8] before introducing our algorithm in Section 5.

4 SOLUTION METHOD

The presented solution method is based on insights and results from recent work [8, 36, 40, 47, 49] establishing a connection between the optimization problems in RL and statistical mechanics. Specifically, there is a mapping at the trajectory level between free energy minimization in statistical mechanics and (entropy-regularized) average reward maximization[4]. This mapping is based on control-as-inference approaches to RL as outlined below.

The control-as-inference perspective of RL reviewed in [36] introduces a binary optimality variable to solve the MaxEnt RL problem for the case of deterministic dynamics. Specifically, Bayesian inference of the posterior dynamics resulting from conditioning on this optimality variable gives the solution to the optimal control problem. Furthermore, this approach also reveals that the optimal trajectory distribution follows a Boltzmann (softmax) distribution, similar to trajectory distributions arising from generalizations of the canonical ensemble in statistical mechanics [16, 17, 49].

This connection to statistical mechanics indicates that the optimal trajectory distribution can be generated by a "tilted" transition matrix, which encodes information about the RL solution, as shown in [8]. Specifically, the trajectory distribution corresponding to the optimal policy for the ERAR objective can be generated from the tilted transition matrix:

$$\widetilde{P}_{(s',a'),(s,a)} = p(s'|s,a)\pi_0(a'|s')e^{\beta r(s,a)}. \tag{8}$$

For a fixed value of β, the tilted matrix generates the dynamics for trajectories having an average return that is "tilted" or biased away from their typical value. Thus, by tuning β, the matrix $\widetilde{P}$ gives access to those trajectories with significantly larger expected returns than those obtained by running the system's unbiased dynamics: $p(s'|s,a)\pi_0(a'|s')$.

Denoting the dominant eigenvalue and corresponding eigenvector of $\widetilde{P}$ as $e^{\beta\theta}$ and $u(s,a)$, the solution to the ERAR MDP is given by:

[4]Since energies can be viewed as negative rewards, these optimization problems are formally equivalent.

Lemma 4.1 (Arriojas et al. [8]). *The entropy-regularized average reward rate is given by θ, and the optimal differential value function and optimal policy are given by:*

$$Q(s,a) = \frac{1}{\beta}\log u(s,a)\,, \tag{9}$$

$$\pi(a|s) = \frac{\pi_0(a|s)u(s,a)}{\sum_a \pi_0(a|s)u(s,a)}. \tag{10}$$

The solution method used requires the transition dynamics to be deterministic, though we discuss the stochastic generalization at the end of this section. It should also be noted that this method obtains the differential value and corresponding policy for the *optimal* entropy-regularized reward-rate, as opposed to a policy-evaluation technique, which may gather information on arbitrary suboptimal policies.

To gain further insights into the utility of the tilted matrix, consider the class of allowed trajectories (of length N) that start at a given state-action pair (s,a) and terminate at (s',a'). If we condition on optimality throughout the trajectory, then the posterior probability of observing this class of trajectories can be shown [8] to be proportional to $\left[\widetilde{P}^N\right]_{(s',a'),(s,a)}$. In the long-time limit (Eq. (5)), the probability for this class of trajectories is well-approximated by the dominant contribution in the spectral decomposition (thus corresponding to the largest eigenvalue) of $\widetilde{P}$:

$$\left[\widetilde{P}^N\right]_{(s',a'),(s,a)} \approx e^{N\beta\theta}u(s,a)v(s',a'), \tag{11}$$

where the left and right dominant eigenvectors u, v correspond to the dominant eigenvalue (Perron root) $\exp(\beta\theta)$. Now, to connect to the solution of the entropy-regularized RL problem, we recall that the soft value function is given by the log-probability of a trajectory being optimal [36], given the initial state-action (s,a). To obtain the corresponding probability, we thus need to sum over all possible final state-action pairs (s',a') in Eq. (11). Taking the logarithm then gives the soft value function as shown in Eq. (21) of [8]. This leads to the identification of θ as the ERAR rate and the derived results shown in Lemma 4.1.

In practice it may not be possible to construct this tilted matrix beyond the tabular setting without model-based techniques. Nevertheless, only a single component of its spectrum is required for an analytic solution to the ERAR MDP. As such, we will consider only the relevant component of the spectrum to devise a model-free learning algorithm, circumventing the need for learning the entire matrix.

The preceding discussion shows that the solution to the ERAR MDP is given by solving an eigenvalue equation for the tilted dynamics:

$$e^{\beta\theta}u(s,a) = \sum_{s',a'} u(s',a')\widetilde{P}_{(s',a'),(s,a)}. \tag{12}$$

Because of the assumptions imposed, the Perron-Frobenius theorem applies to $\widetilde{P}$, guaranteeing a *unique* positive left eigenvector u and well-defined ERAR rate θ. We can rewrite the above equation in the following form, reminiscent of a temporal-difference backup

equation:

$$u(s,a) = e^{\beta(r(s,a)-\theta)} \mathop{\mathbb{E}}_{s'\sim p, a'\sim\pi_0} u(s',a'), \tag{13}$$

for an eigenvector $u > 0$ and (the unique real) eigenvalue with $\theta < 0$. The next largest eigenvalue (in magnitude), denoted $e^{\beta\xi}$, determines the spectral gap of $\widetilde{P}$: $(\beta\xi - \beta\theta)^{-1}$. As discussed in the Introduction, the spectral gap determines an important timescale for the optimal dynamics. This mixing time controls the rate of convergence to the "(quasi) steady-state distribution" (v above) [8, 39] and hence controls the effective time horizon of the agent in its environment.

Given that we do not rely on policy-gradient techniques, and since the expectation in Equation (13) is over the pre-specified prior policy π_0, we have an off-policy algorithm whose trajectory data can be collected by any rollout policy. Note that in log-space, this equation resembles the TD equation for soft Q-learning, without a discount factor, and with an additional correction for the reward-rate, θ.

As discussed by [62], all average reward algorithms (except their Centered Differential TD-learning) are only able to learn an "uncentered" value function: i.e. a differential value function that may be off by some global constant. In contrast, since our solution method is based on a dominant eigenvalue and corresponding normalized eigenvector (guaranteed to exist through the Perron-Frobenius theorem, and normalized as a probability against the corresponding right eigenvector), our algorithm necessarily finds a "centered" differential value function for ERAR MDPs.

For the case of stochastic transition dynamics, the policy derived from the tilted matrix is optimal only if the agent can also change the transition dynamics correspondingly. However, controlling the dynamics is often infeasible and the transition dynamics is generally assumed to be fixed by the problem statement. Through Bayesian inference and matching the two objective functions, [7] shows how to address this issue by iteratively biasing the reward function and transition dynamics of the original MDP so that the solution for a controlled problem coincides with the uncontrolled transition dynamics. To avoid these complications, in the following we focus on the case of deterministic transition functions. We discuss possible extensions to stochastic dynamics for future work in Section 7.

In [8], the authors assume the transition dynamics are irreducible and aperiodic, allowing the Perron-Frobenius theorem to apply to the tilted matrix $\widetilde{P}$. However, many MDPs of interest are inherently reducible (they have absorbing, terminal states). This added complication can nevertheless be treated with the current theory with modest assumptions, which we expand on in the following section. In practice, we treat examples with reducible dynamics using the same approach in Section 6, but with the ERAR rate fixed to zero.

4.1 Reducible Dynamics

Here, we justify the applicability of this framework to reducible dynamics, extending the results of prior work with the following theorem:

Theorem 4.2 (Informal). *Suppose there exists a set of absorbing state-action pairs $(\hat{s},\hat{a})$. In this case, the ERAR rate is identically zero: $\theta = 0$. The optimal differential value function is still given by Eq. (9), except at the absorbing states where its value is simply the corresponding reward:*

$$Q(\hat{s},\hat{a}) = r(\hat{s},\hat{a}), \tag{14}$$

in line with the definition given in discounted approaches.

The expanded version of this result and the necessary constructions and proof can be found in Appendix 1. Essentially, we provide an extended version of the tilted matrix, showing the relevant results from linear algebra still apply. The eqauation $\theta = 0$ can be understood as trajectories will always be absorbed with probability 1, where they will accrue zero reward *ad infinitum* by definition. It is worth noting that many environments of practical importance (cf. classic control suite, Mujoco suite, Atari suite) feature absorbing states; hence breaking the irreducibility assumption common in average-reward frameworks. Our work extends the applicability to reducible settings, which we demonstrate below.

5 PROPOSED ALGORITHMS

In this section, we present pseudocode of our proposed algorithms. We would like to highlight that the core of the algorithms here are built on DQN [41]. This means that existing codebases can easily adapt DQN-style methods with significant advantage (Fig. 2) with limited additional complexity. However, there are several important distinctions specific to the ERAR objective. In Sec. 5.1 we present Algorithm 1 highlights these differences in red. In Sec. 5.2, we present an algorithm to solve the un-regularized average-reward objective. There, Algorithm 3 highlights the differences from Algorithm 1 in blue.

5.1 Solution to ERAR-MDP

Our first algorithm implements three key components present in many value-based deep RL methods: (1) an estimate of the value function (left eigenvector) parameterized by two deep neural nets (inspired by [61]), (2) stochastic gradient descent on a temporal-difference error with Adam [34] and (3) a replay buffer of stored experience. In principle, the replay buffer can be collected by any behavior policy such as an ε-greedy policy [41], but we use the learnt policy (Eq. (10)) for more effective exploration as in [28]. Since we use an average-reward objective, we must also maintain a running estimate of the entropy-regularized reward-rate, θ.

Our model-free algorithm uses the update equations prescribed by [8] (cf. Appendix 1) to learn $u(s,a)$ and θ through stochastic approximation.

For updating the u network, we use a mean squared error loss between the online network and corresponding estimate calculated through a target network:

$$\mathcal{J}(\psi) = \frac{1}{2} \mathop{\mathbb{E}}_{s,a\sim\mathcal{D}} \left(u_\psi(s,a) - \hat{u}_{\bar{\psi}}(s,a) \right)^2. \tag{15}$$

We denote the trained online network as u_ψ and its temporal difference (TD) target as $\hat{u}_{\bar{\psi}}$ (we use $\hat{u}$ to emphasize this is not a neural net itself, but a stand-in for the target value calculated by Eq. (16)).

Algorithm 1 EVAL

1: **IN:** sample budget, environment, β, hyperparameters
2: **Initialize**:
3: Online network weights: $\psi_i \sim$ init. distribution
4: Target network weights: $\bar{\psi}_i = \psi_i$
5: Entropy-regularized reward-rate: $\theta = 0$
6: Replay buffer: $\mathcal{D} = \{\}$
7: **while** $t <$ sample budget **do**
8: Collect experience:
9: Sample action $a \sim \pi_0$
10: Take step in environment $s' \sim p(\cdot|s, a)$
11: Save to replay buffer: $\mathcal{D} \leftarrow \{s, a, r, s'\}$
12: **if** train this step **then**
13: **for** each gradient step **do**
14: Sample minibatch $\mathcal{B} \subset \mathcal{D}$
15: Calculate loss via Eq. (15)
16: Do gradient descent on u network(s)
17: Calculate θ_{new} via Equation (17)
18: **end for**
19: **end if**
20: **if** update θ **then**
21: Update θ estimate: $\theta \leftarrow \theta \cdot (1 - \tau_\theta) + \theta_{\text{new}} \cdot \tau_\theta$
22: **end if**
23: **if** update target parameters **then**
24: Update target parameters (Polyak averaging with parameters τ_ψ).
25: **end if**
26: **end while**
27: **OUT:** Optimal policy for ERAR-MDP

The TD target is calculated based on the lagging network's weights, denoted $\bar{\psi}$ (cf. Appendix 2 for further implementation details). To find the TD target equation, we read off Equation (13) as:

$$\hat{u}_{\bar{\psi}}(s, a) = e^{\beta(r(s,a)-\theta)} \mathop{\mathbb{E}}_{s' \sim p, a' \sim \pi_0} u_{\bar{\psi}}(s', a'). \tag{16}$$

The value of θ must be updated online as well. To this end, we reinterpret Eq. (13) as an equation for the entropy-regularized reward-rate, θ. Since such an equation is valid for any s, a, we can use the entire batch of data $\mathcal{B}$ (sampled uniformly from the replay buffer) to obtain a more accurate estimate for θ. This (s, a)-dependent calculation of θ is based on the current estimate of the left eigenvector, u_ψ (online network as opposed to target network). To preserve the linear structure of the eigenvector equation (Eq. (12)), we propose to track θ through the eigenvalue itself (exponential of θ):

$$e^{\beta\theta} = \frac{1}{|\mathcal{B}|} \sum_{\{s,a,r,s'\} \in \mathcal{B}} \frac{e^{\beta r} \, \mathbb{E}_{a' \sim \pi_0} u_\psi(s', a')}{u_\psi(s, a)}. \tag{17}$$

The value of θ is updated after averaging its value over the "gradient steps" loop in Algorithm 1. To improve stability of the next iteration of Equation (16), we find it helpful to allow θ to slowly mix with previous estimates, and hence use a (constant) step size to update its value (Line 21 in Algorithm 1).

Inspired by [22, 61], we will train two online networks in parallel. We find this to help considerably improve the evaluation reward, as shown in Figure 4. Interestingly, we have found that the popular choice of taking a pessimistic estimate (i.e. min) is not optimal in our experiments. Instead we treat the "aggregation function" as a hyperparameter, and tune it over the choices of min, max, mean. Across all environments, we have found max to be the optimal choice for aggregation. We use two online networks and two corresponding target networks for updates, and upon calculating an aggregated estimate of $u_{\bar{\psi}}$ (shown in Eq. (16)) we use the max over the two target networks, calculated (s, a)-wise: $u_{\bar{\psi}}(s, a) = \max\left\{u^{(1)}_{\bar{\psi}}(s, a), u^{(2)}_{\bar{\psi}}(s, a)\right\}$. We train both of the online networks independently to minimize the same loss in Equation (15), measured against the aggregated target value. This is the same technique employed in [22] for discounted un-regularized RL and in [29] for discounted regularized RL. We use the same aggregation method (max) on the online networks $u^{(i)}_{\psi}$ to calculate θ across the current batch of data via Equation (17).

Algorithm 2 Posterior Policy Iteration (PPI)

Initialize: Prior policy π_0, $\beta > 0$, solve budget.
while $N <$ solve budget **do**
$\pi_0 \leftarrow \text{Solve}(\pi_0, \beta)$
end while
Output: Deterministic optimal policy $\pi^*_{\beta=\infty} = \pi_0$

5.2 Posterior Policy Iteration

Although we plot the greedy policy's reward as an evaluation metric, EVAL inherently aims to maximize the entropy-regularized reward-rate (a combination of rewards and policy relative entropy) shown in Equation (5). Thus, the ERAR MDP's corresponding optimal policy is necessarily stochastic as discussed. The greedy policy used in evaluation, $\hat{\pi}(a|s) = \text{argmax}_a \pi_0(a|s)u(s, a)$ may therefore not be the correct policy for maximizing the un-regularized average reward rate.

Instead, it may be of interest to directly obtain the true greedy solution corresponding to the un-regularized average-reward RL objective ($\beta \to \infty$) shown in Equation (2). A simple way to find such a solution is to slowly increase the value of β throughout training, as done in [28]. However, this can lead to numerical instabilities because of the exponential term $\exp \beta \, (r(s, a) - \theta)$ in Equation 13. One workaround was proposed by an updated version of SAC [29], where the authors implement a learning-based method of temperature annealing based on entropy constraints in the dual space, facilitated by Lagrange multipliers.

A different method of recovering the un-regularized solution ($\beta \to \infty$) is based on an observation by [48], which we outline below. Consider for notational convenience the function Solve : $(\mathbb{R}^{|\mathcal{S}||\mathcal{A}|}, \Delta(\mathcal{A}), \mathbb{R}_+) \to \Delta(\mathcal{A})$, which solves an ERAR MDP for a given choice of reward function, prior policy $\pi_0 \in \Delta(\mathcal{A})$, and inverse temperature β by outputting the corresponding optimal policy, $\pi^* \in \Delta(\mathcal{A})$ (which maximizes the objective in Equation (5)). Then, Theorem 4 in [47] indicates that the iteration shown in Algorithm 2 will converge to the greedy optimal policy: the solution of the un-regularized MDP. Thus, we implement this "Posterior Policy Iteration" (PPI) technique using a function approximator (MLP) to parameterize the prior policy. Additionally, we note that

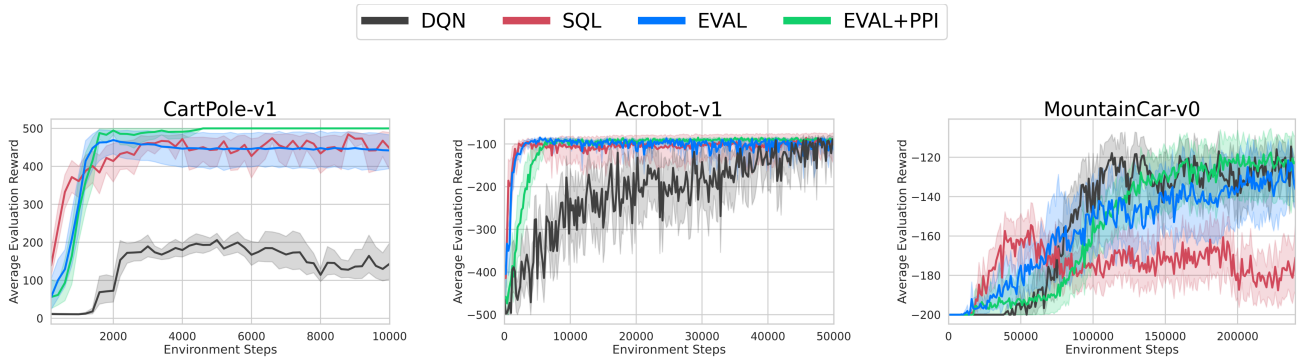

Figure 2: Classic control benchmark comparing soft Q-learning (SQL), deep Q network (DQN) and our two proposed methods (EVAL, EVAL+PPI). We find EVAL and EVAL+PPI to generally obtain higher reward with less variance than SQL or DQN.

this method has the benefits of not requiring a set temperature, an "annealing schedule" $\beta_t \to \infty$ or requiring a minimum entropy with additional computational complexity for a learning scheme [29]. These benefits come at the expense of some additional overhead (the memory and training of another network) and two new hyperparameters: the prior network's update frequency and update weight (Polyak parameter, τ_ϕ, ω_ϕ in Algorithm 3).

To update the prior as prescribed in Algorithm 2, we periodically replace the policy π_0 with the current estimate of the optimal policy $\pi^*(a|s) \propto \pi_0(a|s)u(s,a)$ at a fixed frequency throughout training. Given sufficient time for convergence at each iteration, this process is guaranteed to converge to the solution for $\beta \to \infty$, thereby solving the average-reward MDP without entropy-regularization. In practice, we do not perform hard updates of the prior network's weights, and instead employ Polyak averaging between the parameters of an online prior and a target prior. The online prior is trained to minimize the Kullback-Leibler (KL) divergence to the (current estimate of the) optimal policy via Equation (18):

$$\mathcal{L}_\phi = \frac{1}{|\mathcal{B}|} \sum_{\{s,a,r,s'\} \in \mathcal{B}} \mathrm{KL}\left(\pi^{\bar{\phi}\psi}(\cdot|s) \mid \pi_0^{\phi}(\cdot|s)\right) \tag{18}$$

where $\mathcal{B}$ is a mini-batch randomly sampled from the replay buffer and π is the current estimate for the optimal policy calculated from a combination of the online u network and target π_0 network according to Equation (10),

$$\pi^{\bar{\phi}\psi}(a|s) = \frac{\pi_0^{\bar{\phi}}(\cdot|s)u^{\psi}(s,\cdot)}{\sum_a \pi_0^{\bar{\phi}}(a|s)u^{\psi}(s,a)}. \tag{19}$$

Notice that since the current estimate of the policy is dependent upon the estimate of π_0, we use the online version of both to select actions. Further implementation details and pseudocode of this extension are provided in Algorithm 3 below. Experimentally, we find that even for short timescales of iteration (compared to the timescale for a full solution), the PPI method is still able to recover the $\beta \to \infty$ solution, in agreement with Section 4.1.2 of [48].

5.3 Choice of Architecture

We note that since $u(s,a) > 0$ (by the Perron-Frobenius theorem), we require the u network outputs to be strictly positive and thus use a soft-plus activation function at the output layer. Although a ReLU activation is possible, we found it to give much worse performance across all tasks, despite additional hyperparameter tuning.

We similarly parameterize the prior net as an MLP (same hidden dimension as u-network), but with a softmax output. We have experimented with sharing weights between u and π_0 networks without finding significant performance gains, but this may be an interesting avenue for future exploration.

6 EXPERIMENTS

We focus on the classic control environments from OpenAI's Gymnasium [13] with varying levels of complexity: CartPole-v1, Acrobot-v1, MountainCar-v0. First we will provide benchmark experiments, comparing EVAL against DQN [41] (as it shares many implementation details) and SQL [26] (for its entropy-regularized objective). Although we recognize that these algorithms are not designed to optimize the same objective function, we do not have other benchmarks with which to compare in the discrete-action setting. On the other hand, we emphasize that the metric of interest in discounted RL is often the average evalulation reward. Against this metric, EVAL and EVAL-PPI have stronger performance in terms of sample complexity compared to DQN and SQL.

We compare EVAL with fixed (tuned) β and with PPI (denoted EVAL and EVAL-PPI, respectively) to DQN [46] and SQL [26] on classic control environments in Figure 2. Hyperparameter tuning has been performed for each algorithm on each environment, with the associated values shown in Appendix 2.3.

For proper comparison with other un-regularized methods, the evaluation phase for EVAL uses the greedy policy derived from the learned stochastic policy. Every one thousand environment steps, we pause the training to perform a greedy evaluation, averaged over ten episodes. The resulting reward from each algorithm is averaged over twenty random initializations. All code for reproducing the experiments is available at https://anonymous.4open.science/r/EVAL-D25E.

Algorithm 3 EVAL with PPI

IN: sample budget, environment, β, hyperparameters
Initialize:
Online network weights: $\psi_i, \phi_i \sim W(\cdot)$
Target network weights: $\bar{\psi}_i = \psi_i, \bar{\phi}_i = \phi_i$
Reward-rate: $\theta = 0$
Replay buffer: $\mathcal{D} = \{\}$
while $t <$ sample budget **do**
  Collect experience:
  Sample action $a \sim \pi_0^{\phi} \propto u^{\psi} \cdot \pi_0^{\widehat{\phi}}$
  Take step in environment $s' \sim p(\cdot|s, a)$
  Save to replay buffer: $\mathcal{D} \leftarrow \{s, a, r, s'\}$
  for each gradient step **do**
    Sample minibatch $\mathcal{B} \subset \mathcal{D}$
    Calculate loss via Eq. (15) and Eq. (18)
    Do gradient descent using Adam on u networks and π_0 network
    Calculate θ_{new} via Equation (17)
    Update θ estimate: $\theta \leftarrow \theta \cdot (1 - \tau_\theta) + \theta_{\text{new}} \cdot \tau_\theta$
  end for
  if update target params **then**
    Update target parameters (Polyak averaging with parameters τ_ψ, τ_ϕ).
  end if
end while
OUT: Optimal policy for AR-MDP

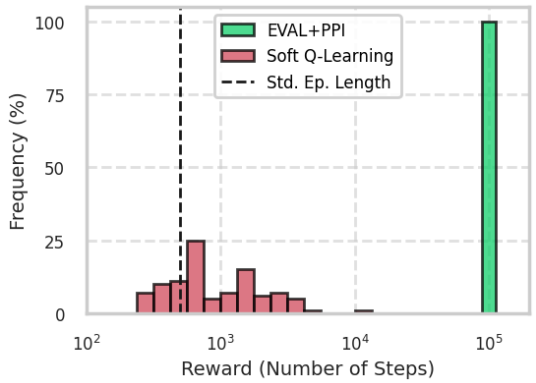

Figure 3: As a demonstration of the usefulness of EVAL+PPI, we consider a modified version of CartPole which represents a continuing task. After training for 5000 steps (in the standard CartPole-v1 environment with a maximum episode length of 500), we compare the evaluation performance of SQL with EVAL+PPI. Specifically, we set the time-limit of the environment much higher: to 100,000 steps. We find that EVAL+PPI consistently reaches the maximum number of steps while SQL only rarely achieves similarly high reward. We find that EVAL+PPI can continue episodes for at least 50 million steps (lower bound at the time of submission).

Since the average-reward objective is aimed at solving continuing tasks, we devise a paradigmatic experiment (inspired by Fig. 2 in [67]) based on CartPole. showcasing the ability of EVAL in the long-time setting. Although both agents are trained in environments with a default (500) timestep limit, the EVAL agent is able to generalize further. The results of this experiment, shown in Figure 3, indicate that EVAL+PPI is far superior to SQL at solving the (effectively) continuing task of balancing the pole forever.

7 LIMITATIONS AND FUTURE WORK

The current work focuses on the case of deterministic transition dynamics. [7] has developed a method for the more general case of stochastic transition dynamics, by iteratively learning biases for the dynamics and rewards. With a model-based algorithm, this seems to be a promising avenue for future exploration for the general case of stochastic transition dynamics. While we have focused on the left eigenvector of $\widetilde{P}$, the right eigenvector $v(s, a)$ can also be learned online. Learning v gives access to the optimal steady-state distribution (uv) a quantity of interest as mentioned in recent work [50]. Although we have tested our method on classic control environments, other more challenging benchmarks such as the ALE [10] can be used as well. Preliminary experiments show that EVAL is able to find the optimal policy in some Atari environments, but further hyperparameter tuning is needed to improve its stability and sample efficiency.

Our current method features discrete action spaces, but adapting the present algorithm to actor-based approaches could allow for a straightforward extension to continuous actions, analogous to the soft actor-critic approach [28]. Since we use an off-policy method, it would be interesting to apply specific exploration policies for rollout collection. Since PPI [47] does not require a particular initial policy, this method can similarly benefit from an improved exploration policy at initialization. As a value-based technique, other ideas from the literature such as TD(n), REDQ [15], PER [52], or dueling architectures [63] may be included. An important contribution for future work is studying the sample complexity and convergence properties of the proposed algorithm. Further work may also provide a connection between PPI and temperature annealing procedures, and port this idea to the discounted framework as well.

8 CONCLUSION

By leveraging the connection of the ERAR objective to the solution of a particular eigenvalue problem, we have presented the first solution to deterministic ERAR MDPs in continuous state spaces by use of function approximation. Our experiments suggest that EVAL compares favorably in several respects to DQN and Soft Q-Learning. Our algorithm leverages the existing DQN framework allowing for a straightforward and easily extendable model applicable to the ERAR objective. We have also provided a solution to the un-regularized, greedy average-reward objective through an iterative procedure in the prior policy space. Our algorithms show an advantage in the classic control suite, especially given that there are no other algorithms for this setting. We believe that the average-reward objective with entropy regularization is a fruitful direction for further research and real-world application, with this work addressing an important gap in the existing literature.

REFERENCES

[1] Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvári, and Gellért Weisz. 2019. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*. PMLR, 3692–3702.

[2] J. Abounadi, D. Bertsekas, and V. S. Borkar. 2001. Learning Algorithms for Markov Decision Processes with Average Cost. *SIAM Journal on Control and Optimization* 40, 3 (2001), 681–698.

[3] Jacob Adamczyk, Argenis Arriojas, Stas Tiomkin, and Rahul V. Kulkarni. 2023. Utilizing Prior Solutions for Reward Shaping and Composition in Entropy-Regularized Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 6 (Jun. 2023), 6658–6665.

[4] Jacob Adamczyk, Volodymyr Makarenko, Stas Tiomkin, and Rahul V Kulkarni. 2024. Boosting Soft Q-Learning by Bounding. *Reinforcement Learning Journal* 5 (2024), 2373–2399.

[5] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. 2019. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*. PMLR, 151–160.

[6] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphaël Marinier, Leonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. 2021. What matters for on-policy deep actor-critic methods? a large-scale study. In *International conference on learning representations*.

[7] Argenis Arriojas, Jacob Adamczyk, Stas Tiomkin, and Rahul V. Kulkarni. 2023. Bayesian inference approach for entropy regularized reinforcement learning with stochastic dynamics. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, Robin J. Evans and Ilya Shpitser (Eds.), Vol. 216. PMLR, 99–109.

[8] Argenis Arriojas, Jacob Adamczyk, Stas Tiomkin, and Rahul V. Kulkarni. 2023. Entropy regularized reinforcement learning using large deviation theory. *Phys. Rev. Res.* 5 (May 2023), 023085. Issue 2.

[9] Argenis A. Arriojas. 2022. *Analytical Framework for Entropy Regularized Reinforcement Learning Using Probabilistic Inference*. Ph.D. Dissertation. University of Massachusetts Boston.

[10] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47 (2013), 253–279.

[11] Dimitri Bertsekas. 2012. *Dynamic programming and optimal control: Volume I*. Vol. 4. Athena scientific.

[12] David Blackwell. 1962. Discrete dynamic programming. *The Annals of Mathematical Statistics* (1962), 719–726.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI gym. *arXiv preprint arXiv:1606.01540* (2016).

[14] Johan Samir Obando Ceron and Pablo Samuel Castro. 2021. Revisiting Rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *Proceedings of the 38th International Conference on Machine Learning*, Marina Meila and Tong Zhang (Eds.), Vol. 139. PMLR, 1373–1383.

[15] Xinyue Chen, Che Wang, Zijian Zhou, and Keith W. Ross. 2021. Randomized Ensembled Double Q-Learning: Learning Fast Without a Model. In *International Conference on Learning Representations*.

[16] Raphaël Chetrite and Hugo Touchette. 2015. Nonequilibrium Markov Processes Conditioned on Large Deviations. *Annales Henri Poincaré* 16, 9 (Sep 2015), 2005–2057.

[17] Raphaël Chetrite and Hugo Touchette. 2015. Variational and optimal control representations of conditioned and driven processes. *Journal of Statistical Mechanics: Theory and Experiment* 2015, 12 (2015), P12001.

[18] Vektor Dewanto, George Dunn, Ali Eshragh, Marcus Gallagher, and Fred Roosta. 2020. Average-reward model-free reinforcement learning: a systematic review and literature mapping. *arXiv preprint arXiv:2010.08920* (2020).

[19] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. 2009. Online Markov decision processes. *Mathematics of Operations Research* 34, 3 (2009), 726–736.

[20] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. 2018. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070* (2018).

[21] Benjamin Eysenbach and Sergey Levine. 2022. Maximum Entropy RL (Provably) Solves Some Robust RL Problems. In *International Conference on Learning Representations*.

[22] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[23] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. 2019. A theory of regularized Markov decision processes. In *International Conference on Machine Learning*. PMLR, 2160–2169.

[24] Jordi Grau-Moya, Felix Leibfried, and Peter Vrancx. 2018. Soft q-learning with mutual-information regularization. In *International conference on learning representations*.

[25] Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. 2018. Composable deep reinforcement learning for robotic manipulation. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 6244–6251.

[26] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, 1352–1361.

[27] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*. PMLR, 1352–1361.

[28] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, 1861–1870.

[29] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. 2018. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905* (2018).

[30] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. 2023. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104* (2023).

[31] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[32] Nan Jiang, Alex Kulesza, Satinder Singh, and Richard Lewis. 2015. The dependence of effective planning horizon on model accuracy. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. 1181–1189.

[33] Sham Machandranath Kakade. 2003. *On the sample complexity of reinforcement learning*. Ph.D. Dissertation. University College London.

[34] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

[35] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2021. SUNRISE: A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, Marina Meila and Tong Zhang (Eds.), Vol. 139. PMLR, 6131–6141.

[36] Sergey Levine. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909* (2018).

[37] Xiaoteng Ma, Xiaohang Tang, Li Xia, Jun Yang, and Qianchuan Zhao. 2021. Average-Reward Reinforcement Learning with Trust Region Methods. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 2797–2803. Main Track.

[38] Sridhar Mahadevan. 1996. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine learning* 22 (1996), 159–195.

[39] Sylvie Méléard and Denis Villemonais. 2012. Quasi-stationary distributions and population processes. *Probability Surveys* 9 (2012), 340–410.

[40] Sanjoy K Mitter and NJ Newton. 2000. The duality between estimation and control. *Published in Festschrift for A. Bennoussan* (2000).

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[42] Abhishek Naik, Roshan Shariff, Niko Yasui, Hengshuai Yao, and Richard S Sutton. 2019. Discounted reinforcement learning is not an optimization problem. *arXiv preprint arXiv:1910.02140* (2019).

[43] Abhishek Naik, Yi Wan, Manan Tomar, and Richard S Sutton. 2024. Reward Centering. *arXiv preprint arXiv:2405.09999* (2024).

[44] Gergely Neu, Anders Jonsson, and Vicenç Gómez. 2017. A unified view of entropy-regularized Markov decision processes. *arXiv preprint arXiv:1705.07798* (2017).

[45] Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. 2023. Controllability-Aware Unsupervised Skill Discovery. In *Proceedings of the 40th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 202)*, Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (Eds.). PMLR, 27225–27245.

[46] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22, 268 (2021), 1–8.

[47] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. 2012. On stochastic optimal control and reinforcement learning by approximate inference. *Proceedings of Robotics: Science and Systems VIII* (2012).

[48] Konrad Cyrus Rawlik. 2013. *On probabilistic inference approaches to stochastic optimal control*. Ph.D. Dissertation. The University of Edinburgh.

[49] Dominic C Rose, Jamie F Mair, and Juan P Garrahan. 2021. A reinforcement learning approach to rare trajectory sampling. *New Journal of Physics* 23, 1 (2021), 013013.

[50] Naman Saxena, Subhojyoti Khastagir, Shishir Kolathaya, and Shalabh Bhatnagar. 2023. Off-policy average reward actor-critic with deterministic policy search. In *International Conference on Machine Learning*. PMLR, 30130–30203.

[51] Naman Saxena, Subhojyoti Khastagir, NY Shishir, and Shalabh Bhatnagar. 2023. Off-policy average reward actor-critic with deterministic policy search. In *International Conference on Machine Learning*. PMLR, 30130–30203.

[52] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2015. Prioritized experience replay. *arXiv preprint arXiv:1511.05952* (2015).

[53] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

[54] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[55] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[56] Anton Schwartz. 1993. A reinforcement learning method for maximizing undiscounted rewards. In *Proceedings of the tenth international conference on machine learning*, Vol. 298. 298–305.

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

[58] Emanuel Todorov. 2006. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, B. Schölkopf, J. Platt, and T. Hoffman (Eds.), Vol. 19. MIT Press.

[59] Emanuel Todorov. 2009. Efficient computation of optimal actions. *Proceedings of the national academy of sciences* 106, 28 (2009), 11478–11483.

[60] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 5026–5033.

[61] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double Q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[62] Yi Wan, Abhishek Naik, and Richard S Sutton. 2021. Learning and planning in average-reward Markov decision processes. In *International Conference on Machine Learning*. PMLR, 10653–10662.

[63] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1995–2003.

[64] Yifan Wu, George Tucker, and Ofir Nachum. 2019. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361* (2019).

[65] Xue Yan, Yan Song, Xidong Feng, Mengyue Yang, Haifeng Zhang, Haitham Bou Ammar, and Jun Wang. 2024. Efficient Reinforcement Learning with Large Language Model Priors. arXiv:2410.07927 [cs.LG] https://arxiv.org/abs/2410.07927

[66] Shangtong Zhang, Yi Wan, Richard S Sutton, and Shimon Whiteson. 2021. Average-reward off-policy policy evaluation with function approximation. In *international conference on machine learning*. PMLR, 12578–12588.

[67] Yiming Zhang and Keith W Ross. 2021. On-policy deep reinforcement learning for the average-reward criterion. In *International Conference on Machine Learning*. PMLR, 12535–12545.

[68] Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.

1 THEORY

For convenience, we provide the update equations from [8] (Eq. 26, 27) which will be learned with EVAL:

$$u(s,a) \to (1-\alpha)u(s,a) + \alpha e^{\beta(r(s,a)-\theta)}u(s',a') \tag{20}$$

$$e^{\beta\theta} \to (1-\alpha_\theta)e^{\beta\theta} + \alpha_\theta e^{\beta r(s,a)}\frac{u(s',a')}{u(s,a)} \tag{21}$$

which resembles our Eq. (16) and (17), respectively.

An important distinction between Eq. 26 in [8] and Eq. (20) is that we have changed the sign of θ in their definition. This allows us to write the original objective in Eq. (5) in a way consistent with the average-reward literature (e.g. Eq. (2)).

1.1 Proof of Theorem 4.1

Theorem 1.1. *Suppose there exists a set of absorbing state-action pairs $(\hat{s},\hat{a})$ with reward $r(\hat{s},\hat{a}) \le 0$. Defining a new tilted transition matrix based on the following quantities:*

$$\widetilde{\Delta}_{(\hat{s},\hat{a}),(s,a)} = p(\hat{s}|s,a)\pi_0(\hat{a}|\hat{s})e^{\beta r(s,a)} \tag{22}$$

$$\widehat{\mathbf{r}} = e^{\beta r(\hat{s},\hat{a})}, \tag{23}$$

the tilted matrix $\widetilde{P}$ can be extended as

$$\widehat{P} = \begin{pmatrix} \widetilde{P} & 0 & 0 \\ \widetilde{\Delta} & 0 & 0 \\ 0 & \widehat{\mathbf{r}} & 1 \end{pmatrix} \tag{24}$$

Then, the dominant eigenvalue of $\widehat{P}$ is 1 so that $\theta = 0$ and the conditional trajectory distribution is given by (for $N \gg 1$):

$$\left[\widehat{P}^N\right]_{(s',a'),(s,a)} \approx u(s,a)v(s',a'), \tag{25}$$

where u, v are the left and right eigenvector of $\widehat{P}$ corresponding with the dominant eigenvalue. Although $\widehat{P}$ is not irreducible and the Perron-Frobenius theorem can not be applied, positivity $u(s,a) > 0$ still holds. Moreover, the differential value function at the absorbing states is

$$Q(\hat{s},\hat{a}) = r(\hat{s},\hat{a}), \tag{26}$$

in line with the definition given in discounted approaches.

Proof. First, note that the newly constructed matrix $\widehat{P}$ is stochastic, implying a dominant eigenvalue of 1 (and a uniform right eigenvector). Denote the left eigenvector of $\widehat{P}$ as $u = (u_1, u_2, 1)$. (The third entry corresponds to the absorbed states and is a free variable we set to unity without loss of generality.) Then the corresponding eigenvector equation can be written as the following linear system

$$u_1\widetilde{P} + u_2\widetilde{\Delta} = u_1 \tag{27}$$

$$1\widehat{\mathbf{r}} = u_2 \tag{28}$$

so that

$$u_2 = e^{\beta r(\hat{s},\hat{a})} \tag{29}$$

$$u_1 = u_2\widetilde{\Delta}(I-\widetilde{P})^{-1} \tag{30}$$

The elements of u_2 are strictly positive vector due to our choice of rewards. Also, since $\widetilde{P}$ is a non-negative irreducible matrix with a dominant eigenvalue less than 1, the matrix $I - \widetilde{P}$ can be classified as an "M-Matrix" where the inverse exists, all the elements of the inverse are non-negative, and

$$(I-\widetilde{P})^{-1} = \sum_{N=0}^{\infty}\widetilde{P}^N. \tag{31}$$

Since $\widetilde{P}$ is irreducible, there exists an $n > 0$ such that for all $N > n$, the summand $\widetilde{P}^N_{ij}$ is strictly positive. Again, all the elements of $\widetilde{\Delta}$ are greater than zero by construction, and hence the components of u_1 (and thus u in its entirety) are strictly positive. As such, a corresponding differential value $Q(s,a) = \log u(s,a)$ can be defined for the case of reducible dynamics. □

2 EXPERIMENTAL DETAILS

2.1 Implementation

We implement our algorithm in PyTorch, and follow the style of Stable Baselines3 [46]. The function approximators for $u(s,a)$ (and similarly for π_0, discussed below) are MLPs with state as input, and $|\mathcal{A}|$ output heads. Below, we show the pseudocode for EVAL with the PPI method

(a combination of Algorithm 1 and 2 in the main text), allowing one to solve the un-regularized RL objective. We highlight the differences from Algorithm 1 (main text) in blue. As mentioned in the main text, we require a parameterization for the (online and target) prior policy as well as an update frequency and rolling average weight for the Polyak update.

We highlight the use of Eq. (17) in this algorithm, as it involves an expectation operation taken over the prior policy. We take the target prior for this calculation, as opposed to the online prior network for the calculation of the optimal policy, as seen in the loss function of Eq. (18).

2.2 Soft Q-Learning

We implement our own soft Q-learning algorithm, in a similar style as EVAL (i.e. with a minimization over two networks, and use of a separate lagging target network). Similar to EVAL's evaluation, we take the greedy policy $\pi(s) = \text{argmax}_a Q(s, a)$ for evaluating the agent.

2.3 Hyperparameters

Table 1: Hyperparameter Ranges Used for Finetuning EVAL & PPI

Hyperparameter	Value Range
Learning Rate, η	$(10^{-4}, 10^{-1})$
Inverse Temperature, β	0.01, 0.05, 0.1, 0.5, 1, 2, 5, 10
Batch Size, b	16, 32, 64, 128, 256
Target Update Interval, ω	10, 100, 500, 1000, 2000, 5000
Gradient Steps, UTD	1,5,10,50

For each run, we choose the buffer size to be the same number of steps in the sample budget (except for DQN, which has a tuned value of buffer size B, given by [46]). We also tune the hidden dimension "hid. dim." of the MLP over values in $\{32, 64, 128\}$. In the following Table 2, ω_π refers to the prior policy's target network update interval (in terms of environment steps).

For simplicity, we have found the algorithms to generally work well for some fixed hyperparameters, reducing the search space and potential sensitivity: The Polyak update ratio (for u and π_0 in PPI) is fixed to 1.0 ("hard updates"); The ERAR rate, θ is kept frozen at zero (except for irreducible tabular dynamics); the number of gradient steps per environment step is fixed to 5 throughout; The replay buffer size set to the maximum value (total number of environment steps during training); and the agent is trained after every environment step (cf. "training frequency" in [46]). The finetuned hyperparameters for each environment are listed below. Each is the result of a sweep of roughly 200 runs (in random search), each:

Table 2: Finetuned Hyperparameter Values for EVAL(-PPI)

Environment	hid. dim.	η	b	β	ω	learn starts	ω_π
CartPole-v1	16	$1 \cdot 10^{-3}$	64	2	10	0	500
Acrobot-v1	64	$5 \cdot 10^{-4}$	64	0.01	10	0	500
MountainCar-v0	32	$6 \cdot 10^{-4}$	128	20	100	5000	2000

Table 3: The final two columns show the additionally tuned (with all others held fixed) hyperparameters specific to PPI.

Table 4: Finetuned Hyperparameter Values for DQN

Environment	η	b	B	γ	ω	$\varepsilon_{\text{final}}$	$\varepsilon_{\text{frac}}$	grad step	train freq	learn starts
CartPole-v1	$2.3 \cdot 10^{-3}$	64	100,000	0.99	10	0.04	0.16	128	256	1000
Acrobot-v1	$6.3 \cdot 10^{-4}$	128	50,000	0.99	250	0.1	0.12	-1	4	0
MountainCar-v0	$4 \cdot 10^{-3}$	128	10,000	0.98	600	0.07	0.2	8	16	1000

Table 5: $\varepsilon_{\text{frac}}$ denotes the exploration fraction over which to decay $\varepsilon = 1.0$ to $\varepsilon = \varepsilon_{\text{final}}$. A training frequency of -1 indicates that training of the Q networks occurs only after the end of each rollout episode. The provided optimal values for $\tau = 1.0$ and hidden dimension of 256 throughout all environments.

Table 6: Finetuned Hyperparameter Values for SQL

Environment	hid. dim	η	b	β	γ	τ	ω	grad steps	learn starts
CartPole-v1	64	$2 \cdot 10^{-2}$	64	0.1	0.98	0.95	100	9	1,000
Acrobot-v1	32	$6.6 \cdot 10^{-3}$	128	2.6	0.999	0.92	100	9	2,000
MountainCar-v0	64	$2 \cdot 10^{-3}$	128	0.7	0.99	0.97	100	2	9,000

Table 7: We use the finetuned hyperparameters given in [4] for soft Q-learning in the classic control benchmark.

We use implementations of DQN from stable-baselines3, with the finetuned hyperparameters for each environment given by [46].

3 ADDITIONAL RESULTS

3.1 Number of networks

In light of several recent works maintaining multiple estimates or an entire ensemble of the agent's Q function [15, 35], we consider the effect of learning more than two (online and target) networks on training performance. The results shown in Figure 4 indicate that having greater than one or two networks can drastically improve performance in some environments, with diminishing returns as seen in prior work. Our experiments indicate that a minimum of two networks is required for learning the optimal policy, so we use two networks throughout for simplicity, despite this generally being a tunable parameter. For convenience, we use two independent target networks as well, to separately perform rolling averages of the target parameters before aggregating.

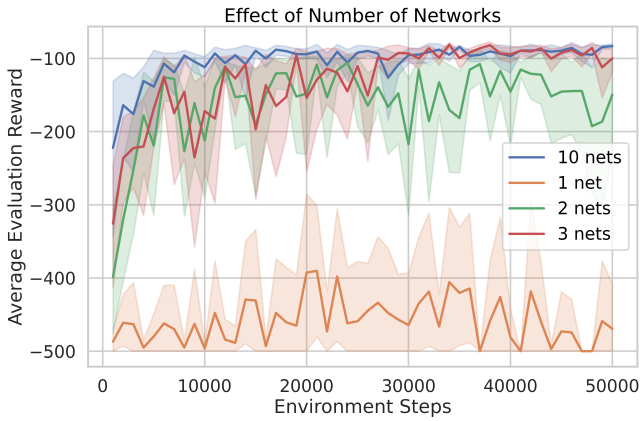

Figure 4: For EVAL (without PPI) we compare the performance for training multiple networks in parallel. All networks are aggregated with the max **function as discussed in Sec. 5.**