StaQ: a Finite Memory Approach to Discrete Action Policy Mirror Descent

Alex Davey, Alena Shilova, Brahim Driss, Riad Akrour

Keywords: Policy Mirror Descent, Entropy Regularization, Deep RL

Summary

In Reinforcement Learning (RL), regularization with a Kullback-Leibler divergence that penalizes large deviations between successive policies has emerged as a popular tool both in theory and practice. This family of algorithms, often referred to as Policy Mirror Descent (PMD), has the property of averaging out policy evaluation errors which are bound to occur when using function approximators. However, exact PMD has remained a mostly theoretical framework, as its closed-form solution involves the sum of all past Q-functions which is generally intractable. A common practical approximation of PMD is to follow the natural policy gradient or use actor-critic approaches, but this potentially introduces errors in the policy update. In this paper, we propose and analyze PMD-like algorithms for discrete action spaces that only keep the last M Q-functions in memory. We show theoretically that for a finite and large enough M, an RL algorithm can be derived that introduces no errors from the policy update, yet keeps the desirable PMD property of averaging out policy evaluation errors. Using an efficient GPU implementation, we then show empirically on medium-scale RL benchmarks such as MinAtar that increasing M improves performance up to a certain threshold after which the performance becomes close to that of exact PMD, reinforcing the theoretical findings that using an infinite sum might be unnecessary and that keeping in memory the last M Q-functions is a practical and theoretically grounded implementation of PMD.

Contribution(s)

1. We extend previous Policy Mirror Descent (PMD) analyses in two ways: i) we show that the averaging effect of policy evaluation errors can be obtained with a D_{KL} penalization of only the policy update, and ii) we extend the analysis to Policy Iteration (PI).
   Context: Kozuno et al. (2019); Vieillard et al. (2020a) analyzed a PMD Value Iteration (VI) algorithm, showing the averaging effect of the D_{KL} regularization, but needed a D_{KL} regularized policy update **and** evaluation; PI was also left as an open problem. Cen et al. (2022); Zhan et al. (2023) analyzed a PI scheme in the same setting as ours—entropy and D_{KL} regularized policy update and entropy regularized policy evaluation—but in their analysis, the effect of policy evaluation errors is minimized when the D_{KL} weight is set to zero, contradicting results of Kozuno et al. (2019); Vieillard et al. (2020a).
2. We propose and analyze StaQ, a PMD variant for discrete action spaces that keeps in memory the last M Q-functions, and show that it converges for M large enough, with no errors due to policy update, and that the averaging of policy evaluation errors is similar to that of exact PMD up to some extra terms that decay exponentially fast w.r.t. M.
   Context: Exact PMD policy is an infinitely increasing sum over previous Q-functions. Existing approaches adopt an actor-critic view to approximate this policy, introducing policy update errors. We propose an alternative update that truncates the infinite sum to M.
3. We perform an efficient batched implementation of StaQ leveraging GPU parallelization and show that increasing M has beneficial effects on performance, with diminishing returns, to the point that StaQ with a high enough M can become indistinguishable from exact PMD.
   Context: Our efficient implementation allows us to evaluate exact PMD (for up to 5 million time-steps) on medium sized MinAtar (Young & Tian, 2019) environments, providing a test bed for comparing approximate and exact PMD algorithms.

StaQ: a Finite Memory Approach to Discrete Action Policy Mirror Descent

Alex Davey[1,2], **Alena Shilova**[1], **Brahim Driss**[2], **Riad Akrour**[2]
{alex.davey,alena.shilova,brahim.driss,riad.akrour}@inria.fr

[1]**Inria TAU, LISN, Université Paris-Saclay, Orsay, France**
[2]**Univ. Lille, Inria, CNRS, Centrale Lille, UMR 9189 – CRIStAL, Lille, France**

Abstract

In Reinforcement Learning (RL), regularization with a Kullback-Leibler divergence that penalizes large deviations between successive policies has emerged as a popular tool both in theory and practice. This family of algorithms, often referred to as Policy Mirror Descent (PMD), has the property of averaging out policy evaluation errors which are bound to occur when using function approximators. However, exact PMD has remained a mostly theoretical framework, as its closed-form solution involves the sum of all past Q-functions which is generally intractable. A common practical approximation of PMD is to follow the natural policy gradient or use actor-critic approaches, but this potentially introduces errors in the policy update. In this paper, we propose and analyze PMD-like algorithms for discrete action spaces that only keep the last M Q-functions in memory. We show theoretically that for a finite and large enough M, an RL algorithm can be derived that introduces no errors from the policy update, yet keeps the desirable PMD property of averaging out policy evaluation errors. Using an efficient GPU implementation, we then show empirically on medium-scale RL benchmarks such as MinAtar that increasing M improves performance up to a certain threshold after which the performance becomes close to that of exact PMD, reinforcing the theoretical findings that using an infinite sum might be unnecessary and that keeping in memory the last M Q-functions is a practical and theoretically grounded implementation of PMD.

1 Introduction

Deep RL has seen rapid development in the past decade, achieving super-human results on several decision making tasks (Mnih et al., 2015; Silver et al., 2016; Wurman et al., 2022). However, the use of neural networks as function approximators exacerbates many challenges of RL, such as the brittleness to hyperparameters (Henderson, 2018) and the poor alignment between empirical behavior and theoretical understandings (Ilyas et al., 2020; Kumar et al., 2020; van Hasselt et al., 2018). To address these issues, many deep RL algorithms consider adding regularization terms, one of which being to penalize the Kullback-Leibler divergence (labeled D_{KL} in the following) between successive policies (Schulman et al., 2015; Wu et al., 2017; Wang et al., 2017; Vieillard et al., 2020b). This family of algorithms is often called Policy Mirror Descent (PMD, Abbasi-Yadkori et al. (2019); Lazic et al. (2021); Zhan et al. (2023)) for its connection—made more explicit in Sec. 4—to the first-order convex optimization method Mirror Descent (Nemirovsky & Yudin, 1983; Beck & Teboulle, 2003). One known property of PMD algorithms, at least in the context of value iteration, is that it averages out policy evaluation errors (Vieillard et al., 2020a). This property can be intuited from the nature of the policy π_k at iteration k, which is a Boltzmann distribution and its

(unnormalized) log-probabilities are a weighted average of past Q-function estimates $\{q_i\}_{i=0}^{k-1}$

$$\pi_k \propto \exp\left(\alpha \sum_{i=0}^{k-1} \beta^i q_{k-i}\right), \tag{1}$$

for temperature $\alpha > 0$ and weight $\beta \in (0, 1]$. In contrast, unregularized RL would pick actions according to $\arg\max_a q_{k-1}(\cdot, a)$, which would be very sensitive to the potentially large policy evaluation errors that can occur in deep RL (Ilyas et al., 2020).

While averaging Q-function estimates might cancel out their errors, implementing Eq. 1 exactly can become intractable due to the infinite nature of the sum. This is especially true when a non-linear function approximator is used for $\{q_i\}_{i=0}^{k-1}$, precluding the existence of a closed-form expression for the policy. As such, prior work considered several types of approximation to PMD, such as following the natural gradient (Kakade, 2001; Peters & Schaal, 2006; Schulman et al., 2015) or performing a few gradient steps over the regularized policy update loss (Schulman et al., 2017; Tomar et al., 2022). Instead, we study both theoretically and empirically an alternative update—that we call StaQ, that is a weighted average of at most the last M Q-function estimates

$$\pi_k \propto \exp\left(\frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i q_{k-i}\right). \tag{2}$$

From a theoretical point of view, we show that for M large enough, such a truncation does not introduce a policy update error, i.e. that in the absence of policy evaluation errors, StaQ will converge exactly to the optimal policy. Moreover, StaQ has a similar averaging of error property as PMD, up to some extra terms that decay exponentially fast w.r.t. M; suggesting that as we increase M, we can quickly recover the behavior of exact PMD. From a practical point of view, the study of such an algorithm is timely in the age of batched GPU computations: indeed we show that for medium-sized problems such as the image-based MinAtar (Young & Tian, 2019) tasks, increasing M has little to no impact on the run-time, making Eq. 2 a practical alternative policy update to actor-critic approaches. This is especially true since the policy update in the discrete action case, to which the scope of this paper is limited to, is optimization free.

2 Related Work

Regularization in RL. Regularization has seen widespread usage in RL. It was used with (natural) policy gradient (Kakade, 2001; Schulman et al., 2015; Yuan et al., 2022), policy search (Deisenroth et al., 2013), policy iteration (Abbasi-Yadkori et al., 2019; Zhan et al., 2023) and value iteration methods (Fox et al., 2016; Vieillard et al., 2020b). Common choices of regularizers include minimizing the D_{KL} between the current and previous policy (Azar et al., 2012; Schulman et al., 2015) or encouraging high Shannon entropy (Fox et al., 2016; Haarnoja et al., 2018), but other regularizers exist (Lee et al., 2019; Alfano et al., 2023). We refer the reader to Neu et al. (2017); Geist et al. (2019) for a broader categorization of entropy regularizers and their relation to existing deep RL methods. In this paper, we use both a D_{KL} penalization w.r.t. the previous policy and a Shannon entropy bonus in a policy iteration context. In Vieillard et al. (2020b), both types of regularizers were used but in a value iteration context. Abbasi-Yadkori et al. (2019); Lazic et al. (2021) are policy iteration methods but only use D_{KL} penalization.

Policy Mirror Descent. Prior works on PMD focus mostly on performing a theoretical analysis of convergence speeds or sample complexity for different choices of regularizers (Li et al., 2022; Johnson et al., 2023; Alfano et al., 2023; Zhan et al., 2023; Lan, 2022; Protopapas & Barakat, 2024). As PMD provides a general framework for many regularized RL algorithms, PMD theoretical results can be naturally extended to many policy gradient algorithms like natural policy gradient (Khodadadian et al., 2021) and TRPO (Schulman et al., 2015) as shown in Neu et al. (2017); Geist et al. (2019). However, the deep RL algorithms from the PMD family generally perform inexact policy updates, adding a source of error from the theoretical perspective. For example, TRPO and the more

recent MDPO (Tomar et al., 2022) rely on approximate policy updates using policy gradients. It was shown in Zhan et al. (2023), that an inexact policy update will add an additional error floor independent of the policy evaluation error. By proposing a finite-memory policy update, we provide new convergence results that offer a new deep RL algorithm policy update step that does not introduce any additional policy update error, in contrast to prior works.

Ensemble methods and growing neural architectures in RL. Saving past Q-functions has previously been investigated in the context of policy evaluation. In Tosatto et al. (2017), a first Q-function is learned, then frozen and a new network is added, learning the residual error. Shi et al. (2019) uses past Q-functions to apply Anderson acceleration for a value iteration type of algorithm. Anschel et al. (2017) extend DQN by saving the past 10 Q-functions, and using them to compute lower variance target values. Instead of past Q-functions, Chen et al. (2021); Lee et al. (2021); Agarwal et al. (2020); Lan et al. (2020) use an ensemble of independent Q-network functions to stabilize Q-function learning in DQN type of algorithms. The aforementioned works are orthogonal to ours, as they are concerned with learning one Q, which can all be integrated into StaQ. Conversely, both Girgin & Preux (2008) and Della Vecchia et al. (2022) use a special neural architecture called the cascade-correlation network (Fahlman & Lebiere, 1989) to grow neural policies. The former work studies such policies in combination with LSPI (Lagoudakis & Parr, 2003), without entropy regularization. The latter work is closer to ours, using a D_{KL}-regularizer but without a deletion mechanism. As such the policy grows indefinitely, limiting the scaling of the method. Finally, Abbasi-Yadkori et al. (2019) save the past 10 Q-functions to compute the policy in Eq. 1 for the specific case of $\beta = 1$, but do not study the impact of deleting older Q-functions as we do in this paper. Growing neural architectures are common in the neuroevolution community (Stanley & Miikkulainen, 2002), and have been used for sequential decision making tasks, but are beyond the scope of this paper.

Parallels with Continual Learning. Continual Learning (CL) moves from the usual i.i.d assumption of supervised learning towards a more general assumption that data distributions change through time (Parisi et al., 2019; Lesort et al., 2020; De Lange et al., 2021; Wang et al., 2024). This problem is closely related to that of incrementally updating the policy π_k, due to the changing data distributions that each Q-function is trained on, and our approach of using a growing neural architecture to implement a D_{KL}-regularized policy update can be seen as a form of parameter isolation method in the CL literature, which offer some of the best stability-performance trade-offs (see Sec. 6 in De Lange et al. (2021)). Parameter isolation methods were explored in the context of continual RL (Rusu et al., 2016), yet remain understudied in a standard *single-task* RL setting.

3 Preliminaries

Let a Markov Decision Problem (MDP) be defined by the tuple (S, A, R, P, γ), such that S and A are finite state and action spaces, R is a bounded reward function $R : S \times A \mapsto [-R_{\mathrm{x}}, R_{\mathrm{x}}]$ for some positive constant R_{x}, P defines the (Markovian) transition probabilities of the decision process and γ is a discount factor. The algorithms presented in this paper can be extended to more general state spaces. However, we limit ourselves to studying finite action spaces, to simplify the sampling from the Boltzmann distribution for the policy (Eq. 2), which would require deeper algorithmic changes for continuous actions spaces, and would remove the benefit of an optimization-free policy update.

Let $\Delta(A)$ be the space of probability distributions over A, and h be the negative entropy given by $h : \Delta(A) \mapsto \mathbb{R}$, $h(p) = p \cdot \log p$, where $\cdot$ is the dot product and the $\log$ is applied element-wise to the vector p. Let $\pi : S \mapsto \Delta(A)$ be a stationary stochastic policy mapping states to distributions over actions. We denote the entropy regularized V-function for policy π and regularization weight $\tau > 0$ as $V_\tau^\pi : S \mapsto \mathbb{R}$, which is defined by

$$V_\tau^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \{ R(s_t, a_t) - \tau h(\pi(s_t)) \} \middle| s_0 = s \right]. \tag{3}$$

In turn, the entropy regularized Q-function is given by $Q_\tau^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{s'} \left[V_\tau^\pi(s') \right]$. The V-function can be written as the expectation of the Q-function plus the current state en-

tropy, i.e. $V_\tau^\pi(s) = \mathbb{E}_a\left[Q_\tau^\pi(s,a)\right] - \tau h(\pi(s))$ which leads to the Bellman equation $Q_\tau^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{s',a'}\left[Q_\tau^\pi(s',a') - \tau h(\pi(s'))\right]$. In the following, we will write policies of the form $\pi(s) \propto \exp(Q(s,\cdot))$ for all $s \in S$ more succinctly as $\pi \propto \exp(Q)$. We define optimal V and Q functions where for all $s \in S, a \in A$, $V_\tau^\star(s) := \max_\pi V_\tau^\pi(s)$ and $Q_\tau^\star(s,a) := \max_\pi Q_\tau^\pi(s,a)$. Moreover, the policy $\pi^\star \propto \exp(Q_\tau^\star/\tau)$ satisfies $Q_\tau^{\pi^\star} = Q_\tau^\star$ and $V_\tau^{\pi^\star} = V_\tau^\star$ simultaneously for all $s \in S$ (Zhan et al., 2023). In the following, we will overload notations of real functions defined on $S \times A$ and allow them to only take a state input and return a vector in $\mathbb{R}^{|A|}$. For example, $Q_\tau^\pi(s)$ denotes a vector for which the i^{th} entry $i \in \{1, \dots, |A|\}$ is equal to $Q_\tau^\pi(s,i)$.

For the convergence analysis, we will make use of a matrix representation of the MDP by overloading the notations of R and P. Let $I_s : S \mapsto \{1, \dots, |S|\}$ be an arbitrary bijective function that will provide an ordering over the state space, and we let $I_{sa} : S \times A \mapsto \{1, \dots, |S||A|\}$ be an arbitrary bijective function that orders the state-action space. Using these indexing functions, we will overload the notation of the reward function by seeing R as an $|S||A| \times 1$ matrix such that row $I_{sa}(s,a)$ and column 1 of the matrix R satisfies $R_{(I_{sa}(s,a),1)} = R(s,a)$. Similarly for the transition function P which we see as an $|S||A| \times |S|$ matrix such that $P_{(I_{sa}(s,a),I_s(s'))} = P(s'|s,a)$. A policy π will be seen as a $|S| \times |S||A|$ matrix such that $\pi_{(I_s(s'),I_{sa}(s,a))} = \pi(a|s)$ if $s' = s$ and 0 otherwise. On such matrix representation of the policy we can apply the negative entropy row-wise such that $h(\pi)$ is a $|S| \times 1$ matrix where $h(\pi)_{(I_s(s),1)} = h(\pi(s))$. Using all of the above notations, we write the Bellman equation associated to policy π in matrix notation over a Q-function represented by an $|S||A| \times 1$ matrix that satisfies $T_\tau^\pi Q_\tau^\pi = Q_\tau^\pi = R + \gamma P\left[\pi Q_\tau^\pi - \tau h(\pi)\right]$. We also write the Bellman optimality operator on matrices as $T_\tau^\star f = R + \gamma P\left[\max_p pf - \tau h(p)\right]$, where the maximization $\max_p$ is made row-wise over probability matrices of shape $|S| \times |S||A|$ encoded using the same convention as policy matrices π described above.

4 Policy Mirror Descent and averaging of error

To find $\pi^\star$, we focus on Entropy-regularized Policy Mirror Descent (EPMD) methods (Neu et al., 2017; Abbasi-Yadkori et al., 2019; Lazic et al., 2021) and notably on those that regularize the policy update with an entropy and D_{KL} term and use an entropy regularized Bellman operator (Lan, 2022; Zhan et al., 2023). The EPMD setting discussed here is also similar to the regularized natural policy gradient algorithm on softmax policies of Cen et al. (2022). We will put special emphasis in this section on policy evaluation errors and show how convergence of EPMD depends on this error.

4.1 Entropy regularized value iteration

To ease the discussion, let us first consider an approximate value iteration algorithm. Let $\xi_k : S \times A \mapsto \mathbb{R}$ be the unnormalized log-probability (which we refer to as logits for short) of π_k, i.e. $\pi_k \propto \exp(\xi_k)$. We define entropy regularized value iteration by the following two steps.

Evaluation step: let $q_k : S \times A \mapsto \mathbb{R}$ be a sequence of functions such that $q_0 = 0$ and for all $k \geq 0$, $q_{k+1} = T_\tau^{k+1} q_k + \epsilon_{k+1}$, where $T_\tau^{k+1} = T_\tau^{\pi_{k+1}}$ is the Bellman operator associated to policy π_{k+1} and ϵ_{k+1} represents the evaluation error due to, e.g., knowing only a sample estimate of the Bellman operator or knowing it only on a sub-set of the state-action space.

Policy update step: letting $\xi_k = 0$, i.e. the first policy is uniform over the action space, for each q_k we update the policy in EPMD by solving the following optimization problem

$$\forall s \in S, \quad \pi_{k+1}(s) = \operatorname*{arg\,max}_{p \in \Delta(A)} \{q_k(s) \cdot p - \tau h(p) - \eta D_{\text{KL}}(p; \pi_k(s))\} \tag{4}$$

where $D_{\text{KL}}(p;p') = p \cdot (\log p - \log p')$ and $\eta > 0$ is the D_{KL} regularization weight. This update admits the well known closed-form solution given by

$$\xi_{k+1} = \beta \xi_k + \alpha q_k, \tag{5}$$

where $\alpha = \frac{1}{\eta+\tau}$ and $\beta = \frac{\eta}{\eta+\tau}$. Let us characterize the convergence of such an algorithm. In the remainder of this paper, we will be interested in bounding the norm $\|Q^\star_\tau - \tau\xi_k\|_\infty$ which we want as small as possible since the logits of the optimal policy are $\frac{Q^\star_\tau}{\tau}$. Moreover, from a bound over $\|Q^\star_\tau - \tau\xi_k\|_\infty$ we can derive the more common bound over Q-functions since

Lemma 4.1. *Let policy $\pi_k \propto \exp(\xi_k)$ and Q^k_τ its Q-function; then $\left\|Q^\star_\tau - Q^k_\tau\right\|_\infty \leq \frac{2\|Q^\star_\tau - \tau\xi_k\|_\infty}{1-\gamma}$.*

Proofs of for all theoretical statements are given in the appendix. The convergence of entropy regularized value iteration is given by the following theorem

Theorem 4.2. *(Convergence of entropy regularized value iteration) Letting $E_j = (1-\beta)\sum_{i=1}^{j}\beta^{j-i}\epsilon_i$ and $R_m = R_{\mathrm{x}} + \gamma\tau\log|A|$, we have at iteration $k+1$ that $\|Q^\star_\tau - \tau\xi_{k+1}\|_\infty \leq \gamma^{k+1}\|Q^\star_\tau\|_\infty + R_m\sum_{i=0}^{k}\gamma^i\beta^{k-i} + \sum_{i=0}^{k}\gamma^i\|E_{k-i}\|_\infty$.*

The first term of the upper bound goes to zero as $k \to \infty$. This term is also found in unregularized value iteration (see, e.g. Theorem 1.12 of Agarwal et al. (2021)) and is due to the contraction property of the Bellman operators. The second term is a constant multiplied by $\sum_{i=0}^{k}\gamma^i\beta^{k-i}$. It can be shown that this sum satisfies $\sum_{i=0}^{k}\gamma^i\beta^{k-i} \leq \max\{\gamma,\beta\}^k(k+1)$, which goes to zero as $k \to \infty$. In the limit of $\eta \to 0$, where we would drop the D_{KL} regularization, $\beta \to 0$ and $\sum_{i=0}^{k}\gamma^i\beta^{k-i} \to \gamma^k$, yielding an error term that goes to zero at the same rate as $\gamma^{k+1}\|Q^\star_\tau\|_\infty$. However, as we increase the D_{KL} regularization, β approaches 1 and this second term satisfies whenever $\beta > \gamma$, $\sum_{i=0}^{k}\gamma^i\beta^{k-i} = \beta^k\sum_{i=0}^{k}\left(\frac{\gamma}{\beta}\right)^i \leq \frac{\beta^{k+1}}{\beta-\gamma}$. While this term still goes to zero as $k \to \infty$, by increasing the D_{KL} regularization we pay the price of a slower convergence for $\beta > \gamma$.

Finally, the term $\sum_{i=0}^{k}\gamma^i\|E_{k-i}\|_\infty$ constitutes the error floor of entropy regularized value iteration stemming from the evaluation errors ϵ_i. This error floor might remain above zero even as $k \to \infty$. In the limit of $\eta \to 0$, $\|E_j\|_\infty \to \|\epsilon_j\|_\infty$ and the error floor will tend to $\sum_{i=0}^{k}\gamma^i\|\epsilon_j\|_\infty$, i.e. a weighted sum of the norms of the evaluation errors. However, as we increase the D_{KL} penalization, there is a hope that the evaluation errors will cancel each other in $(1-\beta)\sum_{i=1}^{j}\beta^{j-i}\epsilon_i$, leading to a lower value of $\|E_j\|_\infty$ than if we would only consider the norm of the last error $\|\epsilon_j\|_\infty$. As such, by increasing η we might slow down the convergence rate but potentially lower the error floor $\sum_{i=0}^{k}\gamma^i\|E_{k-i}\|_\infty$ and return a better final policy. This result is similar to that of Vieillard et al. (2020a), except that our algorithm uses a Bellman operator that only applies entropy regularization (as used for example in the learning of Q-functions in SAC (Haarnoja et al., 2018)), whereas Vieillard et al. (2020a) considered the Bellman operator that applies both an entropy and a D_{KL} regularization. Moreover, the latter work was restricted to the analysis of value iteration, but before discussing StaQ we need first to extend the above analysis to policy iteration as StaQ is a policy iteration algorithm.

4.2 Entropy regularized policy iteration

The policy iteration version of EPMD is quite similar to value iteration except that in the evaluation step, $q_k = Q^k_\tau + \epsilon_k$, where $Q^k_\tau = Q^{\pi_k}_\tau$ is the Q-function associated to π_k. The approximation q_k of Q^k_τ can be obtained for instance by applying the Bellman operator T^k_τ (or a noisy version thereof) several times on q_{k-1}—instead of a single time in value iteration. As value and policy iteration algorithms are quite similar, the analysis of the latter follows the same general template, except that the error propagates in slightly more complex ways. In the case of policy iteration, we need a way to relate q_k and q_{k+1} as this relation is not as direct as in value iteration. To do so, we will make use of the policy improvement lemma (Section 4.2 of Sutton & Barto (2018)). In the unregularized RL case, the lemma states that a policy greedy w.r.t. a Q-function will have a greater or equal Q-function at every state-action pair. A similar property holds in the entropy regularized case. However, in the presence of policy evaluation errors, the new policy π_{k+1}—obtained by maximizing Eq. 4—might increase the probability of sub-par actions and improvement is only guaranteed up to some policy improvement error characterized by the following lemma

Lemma 4.3 (Approximate policy improvement)**.** *Let $\mu_k = (I - \gamma P \pi_k)^{-1}$ be the (unnormalized) state distribution associated to policy π_k. At any iteration $k \geq 0$ of entropy regularized policy iteration, we have that $Q_\tau^{k+1} \geq Q_\tau^k - \epsilon_{\Delta_k}$, with $\epsilon_{\Delta_k} := \gamma \mu_{k+1} P(\pi_k - \pi_{k+1})\epsilon_k$.*

The policy improvement error ϵ_{Δ_k} is invariant to a constant shift in the evaluation error ϵ_k. Indeed, we have that for any real value c, $(\pi_k - \pi_{k+1})(\epsilon_k + c\mathbf{1}) = (\pi_k - \pi_{k+1})\epsilon_k$, where $\mathbf{1}$ is a vector of ones. Additionally, if $\epsilon_k = 0$ or any other constant vector, then policy improvement is guaranteed. However, we might not improve over the previous Q-function if we overestimate a bad action or underestimate a good one. The overall convergence of entropy regularized policy iteration is characterized by the following theorem

Theorem 4.4. *(Convergence of entropy regularized policy iteration) Letting $\epsilon_{-1} = 0$ and $E_j := (1-\beta)\sum_{i=0}^{j} \beta^{j-i}(\epsilon_i - \gamma P \pi_i(\epsilon_{i-1} + \epsilon_{\Delta_i}))$, we have at iteration $k+1$ that $\|Q_\tau^\star - \tau\xi_{k+1}\|_\infty \leq \gamma^{k+1}\|Q_\tau^\star\|_\infty + \frac{2-\gamma-\beta}{1-\gamma} R_m \sum_{i=0}^{k} \gamma^i \beta^{k-i} + \sum_{i=0}^{k} \gamma^i \|E_{k-i}\|_\infty$.*

As can be seen, the upper bound of Theorem 4.4 follows a very similar structure to that of value iteration, with the main difference being in the error floor $\sum_{i=0}^{k} \gamma^i \|E_{k-i}\|_\infty$ that now notably depends on the policy improvement error discussed above. While this error floor involves more quantities, the general scheme remains the same and one hopes that there are values of η such that a cancellation of terms leads to a lower error floor compared to the unregularized case while not slowing policy iteration too much. This analysis improves over that of Zhan et al. (2023), that only considered a uniform worst case error, leading to a less interesting upper bound where the smallest error floor—and the fastest convergence rate—is always obtained by choosing $\eta = 0$, using no D_{KL} regularization.

4.3 Approximate policy update

We have analyzed so far the convergence of EPMD algorithms, considering only evaluation errors. In practice, the policy update that consists in solving Eq. 4 might prove challenging without approximations. Indeed, while this policy update leads to a closed form solution in the space of policy logits (Eq. 5), it might not be possible to implement exactly if the state-action space is too large. In this case, one would typically use a second function approximator for the policy. Such actor-critic approaches to PMD include the likes of TRPO (Schulman et al., 2015), ECPO Mei et al. (2019) and MDPO (Tomar et al., 2022) that are discussed in more details in App. 10.2.

Irrespective of the AC method used, representing the policy logits with a function approximator is likely to introduce a new type of error. In Zhan et al. (2023), the authors analyzed an approximate EPMD scheme such that the policy update objective in Eq. 4 is optimized up to some error ϵ_{opt} for all states and all iterations. They showed that the resulting algorithm would converge at the same rate as its exact counterpart but would reach an error floor that depends on ϵ_{opt}, independently of the existence of policy evaluation errors. In this paper, we investigate an alternative policy update, that truncates the infinite sum of EPMD to result in a practical algorithm that does not introduce errors from the policy update yet keeps the appealing property of averaging policy evaluation errors.

5 Finite-memory policy mirror descent

Let us now consider a PMD-like algorithm that keeps in memory at most M—with M being a finite and strictly positive integer—Q-function estimates. The policy at iteration k is now given by Eq. 2, which can be written as a recursive update in the logits space in the following way

$$\xi_{k+1} = \beta \xi_k + \alpha q_k + \frac{\alpha \beta^M}{1 - \beta^M}(q_k - q_{k-M}), \tag{6}$$

where $q_{k-M} := 0$ whenever $k - M < 0$, and $q_k = Q_\tau^k + \epsilon_k$ otherwise. In contrast to vanilla EPMD in Eq. 5, we now 'delete' at each update the oldest Q-function estimate q_{k-M} and also slightly overweight the most recent Q-function estimate to ensure that the Q-function weights sum to 1.

This weight correction in Eq. 2—the extra multiplication by $\frac{1}{1-\beta^M}$ compared to the vanilla sum in Eq. 1—is important as otherwise, the logits might never converge to $\frac{Q^\star_\tau}{\tau}$ even when the last M Q-function estimates are all equal to $Q^\star_\tau$. Indeed, without the weight correction and since $\tau\alpha = 1-\beta$, we would have $\tau\xi_k = (1-\beta)\sum_{i=0}^{M-1}\beta^i Q^\star_\tau = (1-\beta^M)Q^\star_\tau$.

The logits update in Eq. 6 can be interpreted as the result of the following optimization problem

$$\forall s \in S, \pi_{k+1}(s) = \arg\max_{p\in\Delta(A)} p\cdot[q_k + \frac{\beta^M}{1-\beta^M}(q_k - q_{k-M})](s) - \tau h(p) - \eta D_{\mathrm{KL}}(p;\pi_k(s)) \quad (7)$$

Now instead of maximizing the latest Q-function estimate, the policy also maximizes the difference between the latest and oldest estimate out of the last M Q-functions. This introduces an additional source of policy improvement error, but in our theoretical analysis, we show that for a finite but large enough M, this error will vanish as $k\to\infty$, leaving us with an error floor that only depends on the evaluation errors. Specifically, the algorithm given by Eq. 6 has the following convergence properties

Theorem 5.1 (Convergence of finite memory EPMD)**.** *Let M such that $M > \log\frac{(1-\gamma)^3}{(1+\gamma)(1-\gamma)^2+4(\gamma+\gamma^2)}/\log\beta$, $\gamma_M = \frac{\gamma}{1-\beta^M}$, $c = \frac{\beta^M}{1-\beta^M}\left(\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2}+\gamma\right)$, let d_0 be the unique root of $d^{2M+1} - \gamma_M d^{2M} - c$ in the interval $(\gamma_M, 1)$, define the matrix $A_{k+1} = \gamma P\pi_{k+1}(I+\gamma\mu_{k+1}P(\pi_{k+1}-\pi_k))$, and error terms $E_j = \frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i(\epsilon_{k-i} - A_{k-i}\epsilon_{k-i-1})$, and $T_k = \|E_k\|_\infty + \frac{\beta^M(1-\beta)}{(1-\beta^M)^2}\left\|\sum_{i=0}^{M-1}\beta^i A_{k-i}(\epsilon_{k-i-1}-\epsilon_{k-i-1-M})\right\|_\infty$ and worst error term $\bar{T} = \max_{0\le i\le k} T_i$, then $\|Q^\star_\tau - \xi_{k+1}\|_\infty \le d_0^{k+1}\|Q^\star_\tau\|_\infty + \sum_{i=0}^{k}\gamma_M^i\left(T_{k-i} + c\frac{\bar{T}}{1-\gamma_M-c}\right)$.*

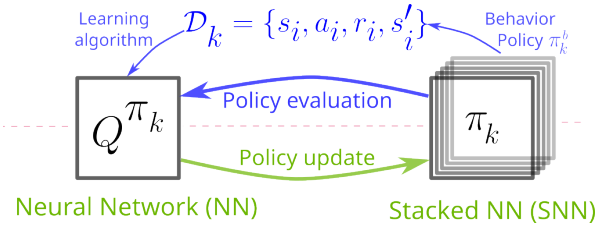
In finite memory EPMD we no longer have an explicit expression for the convergence rate, but provided M is large enough, we know that it exists as the unique root of a function in the range $(\gamma_M, 1)$. Moreover, as $M\to\infty$, the convergence rate goes to γ since in this case: $\gamma_M\to\gamma$, $c\to 0$ and γ becomes quite clearly the unique non-zero solution to $d^{2M+1} - \gamma d^{2M} = 0$ for unknown d. As in exact EPMD, we note that in the absence of evaluation errors, the algorithm converges to the optimal policy. In terms of error floor, we find a similar expression of E_j as in Sec. 4.2 up to the truncation to the latest M errors. However, the error floor does not depend only on E_j but on an additional term in T_j that depends on older evaluation error terms. This additional term is weighted by $\frac{\beta^M(1-\beta)}{(1-\beta^M)^2}$ which decreases exponentially fast towards zero as we increase M. Finally, the error floor is not just a weighted sum of average evaluation errors but also depends on the worst average evaluation error $\bar{T}$. Fortunately, this term is weighted again by a constant that decreases exponentially fast towards zero as M increases. Thus the theoretical analysis indicates in terms of averaging of policy evaluation errors, that we approach the desirable behavior of an exact EPMD policy update exponentially fast by increasing M.

The rest of the paper explores the practical implications of this result: what are the tangible benefits of EPMD, and how fast (in terms of memory size M) can a finite memory variant approach the behavior of EPMD with an exact policy update? We first need an efficient implementation of EPMD.

5.1 Practical implementation

We implement an efficient version of the policy in Eq. 2 using stacked neural networks (illustrated in Fig. 1). By using batched operations we make efficient use of GPUs and compute multiple Q-values in parallel. We call the resulting algorithm StaQ[1]. After each policy evaluation, we push the weights corresponding to this new Q-function onto the stack. If the stacked NN contains more than M NNs, the oldest NN is deleted in a "first in first out" fashion. If implementing exact EPMD, then we never delete older Q-functions.

[1]The code is available at github.com/alexdavey/StaQ.



Algorithm	Runtime (hr)
StaQ M=1	9.6
StaQ M=100	10.4
StaQ M=300	10.5
AC-NoKL (SAC)	13.9
AC-MProj (ECPO)	13.1
AC-DirectOpt (MDPO)	13.2
DQN	5.8

Figure 1: **Left:** Overview of StaQ, showing the continual training of a Q-function (left), from which we periodically "stack" frozen weight snapshots to form the policy (right). At each iteration k, we perform: i) Policy evaluation, generating a dataset $\mathcal{D}_k$ of transitions that are gathered by a behavior policy π_k^b, typically derived from π_k, and then learn Q^{π_k} from $\mathcal{D}_k$; ii) Policy update, performed by "stacking" the NN of Q^{π_k} into the current policy. The policy update is optimization-free and theoretically grounded (Sec. 5), thus only the choice of π_k^b and the policy evaluation algorithm can remain sources of instabilities in this deep RL setting. **Right:** Runtime of StaQ and Actor-Critic baselines on MinAtar/SpaceInvaders-v1 (NVIDIA Tesla V100 GPU). Increasing M has a marginal effect on run time, and StaQ is faster than AC methods due to the optimization-free policy update. DQN is $\sim$65% faster than $M = 1$ StaQ, as it only uses one target Q-function, rather than two.

To further reduce the impact of a large M, we pre-compute ξ_k for all entries in the replay buffer[2] at the start of policy evaluation. The logits ξ_k are used to sample on-policy actions when computing the targets for Q_τ^k. As a result of the pre-computation, during policy evaluation, forward and backward passes only operate on the current Q-function and hence the impact of large M is minimized. However rolling out the current behavioural policy π_k^b still requires a full forward pass. Conversely, the policy update consists only of adding the new weights to the stack, and thus, is optimization free and (almost) instantaneous. The table in Fig 1 shows that varying M has little impact on the runtime of StaQ on GPU (requiring at most 3GB of GPU memory), at least for medium-sized environments such as MinAtar (Young & Tian, 2019). StaQ is faster than the Actor-Critic baselines (described in App. 10.2) as no gradient steps are required for the policy update, and within the same order of magnitude as fast Deep RL baselines such as DQN – the speed difference being due to the use of two target Q-functions. Further implementation details are provided in App. 11.

6 Experiments

Environments. For the main experiments we consider the 5 MinAtar environments (Young & Tian, 2019). We supplement this, in the Appendix, with results for 4 Classic Control tasks from Gymnasium (Towers et al., 2023), covering the full suite of environments suggested by Ceron & Castro (2021) for comparing deep RL algorithms with finite action spaces.

Algorithms. In our main set of experiments, we compare finite memory EPMD for several values of M on up to 5 million timesteps. We notably consider the two extremes of $M = 1$, using no D_{KL} regularization and $M = 1000$ that never deletes a Q-function within the 5 million timesteps window (labeled `Exact PMD` in figures). We further add several baselines below, based on policy updates common in the literature, see App. 10 for more details. The policy evaluation strategy, and all other hyperparameters are kept the same as StaQ, such that they only differ by their policy update.

`NatGrad + LS (TRPO)` uses the conjugate gradient approach of TRPO (Schulman et al., 2015), with line search. We also add several actor-critic methods, including `AC-DirectOpt (SAC)`, a discrete-actions version of SAC (Haarnoja et al., 2018). Note that in the discrete-actions setting we can sample directly from the policy and so are not required to introduce an actor — the $M = 1$ limit of StaQ can be seen as a discrete-actions version of SAC (see App. 10.1).

[2]Since we use small replay buffer sizes of 50K transitions, we are likely to process each transition multiple times (25.6 times in expectation in our experiments) making this optimization worthwhile.

Two further actor-critic algorithms which optimize the PMD objective are: i) ECPO (`AC-MProj`, Mei et al. (2019)) which minimizes the M-Projection between the actor and the recursive PMD solution $\pi_{k+1} \propto \pi_k^\beta \exp\left(\alpha Q_\tau^k\right)$, argued to better preserve the support of the distribution and prevent the premature elimination of actions, and ii) MDPO (`AC-DirectOpt`, Tomar et al. (2022), which skips the closed form solution, and directly optimizes a state averaged version of the entropy regularized PMD update by gradient ascent.

Deep RL baselines. In the appendix, we complement our main experiment with comparisons to deep RL baselines such as the value iteration algorithm DQN (Mnih et al., 2015) and its entropy-regularized variant M-DQN (Vieillard et al., 2020b), the policy gradient algorithm TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017). These baselines differ more widely and in ways orthogonal (policy evaluation, exploration, replay buffer management) to the main focus of this paper which is the policy update of EPMD, and thus are only provided for reference.

Results. Across 20 seeds per algorithm and task, aggregate performance improves almost monotonically with M (Fig. 2). On nearly every task, sufficiently large M matches exact EPMD, with $M \geq 300$ virtually indistinguishable on all tasks (App. 9.1). This reinforces the theoretical insights that StaQ can match the behavior of exact EPMD.

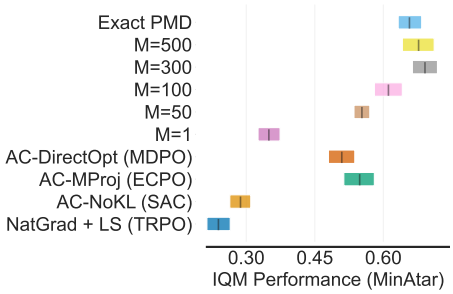


Figure 2: Normalized policy performance aggregated across MinAtar tasks, showing the interquantile mean and 95% confidence intervals. See App. 9 for further results and Deep RL baselines.

Compared to the natural policy gradient and other actor-critic PMD baselines, performance is generally improved across all tasks, indicating that StaQ is potentially a better alternative to other actor-critic based PMD schemes, at least on environments where Q-function computations can be efficiently batched, because of the improved performance and runtime. In the appendix, we provide additional experimental results, including comparisons to deep RL baselines in Fig. 4. In App. 9.4, we use the ShinRL library (Kitamura & Yonetani, 2021) to investigate the cancellation of evaluation errors on environments where Q-functions can be computed exactly, and show that it can occur even on simple environments.

7 Conclusion

In this paper, we proposed a policy update rule based on policy mirror descent, that keeps in memory at most M Q-functions. By increasing M we can quickly mimic exact EPMD both from a theoretical and empirical perspective. The resulting policy update has a solid theoretical foundation and clear empirical benefits, making it a valid alternative to existing EPMD schemes, at least for medium-sized tasks. Due to its exact policy update, and the absence of a gap between the theoretical algorithm and the practical implementation, StaQ provides a promising setting for testing other components of RL such as policy evaluation, for instance, the recent methods that use normalization techniques to reduce policy evaluation error (Gallici et al., 2025; Bhatt et al., 2024).

Surprisingly, even when M is large, the final computational burden on many common RL tasks is small, due to the stacking of the Q-functions which parallelize well on modern hardware. For large-scale tasks such as Atari (Bellemare et al., 2013), the memory usage and inference time may become significant, however future algorithmic improvements such as compressing old Q-functions could yield a computationally efficient deep RL algorithm, at the cost of an inexact policy update.

Acknowledgments

A. Davey and B. Driss were funded by the project ANR-23-CE23-0006. This work was granted access to the HPC resources of IDRIS under the allocation 2024-AD011015599 made by GENCI.

References

Yasin Abbasi-Yadkori, Peter Bartlett, Kush Bhatia, Nevena Lazic, Csaba Szepesvari, and Gellert Weisz. POLITEX: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, 2019.

Alekh Agarwal, Nan Jiang, Sham M. Kakade, and Wen Sun. *Reinforcement Learning: Theory and Algorithms*. 2021.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *International conference on machine learning*, pp. 104–114. PMLR, 2020.

Carlo Alfano, Rui Yuan, and Patrick Rebeschini. A novel framework for policy mirror descent with general parameterization and linear convergence. In *Advances in Neural Information Processing Systems*, 2023.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*, 2017.

Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J. Kappen. Dynamic policy programming. *Journal of Machine Learning Research*, 2012.

Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, May 2003. ISSN 0167-6377. DOI: 10.1016/S0167-6377(02)00231-6.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 2013.

Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. CrossQ: Batch Normalization in Deep Reinforcement Learning for Greater Sample Efficiency and Simplicity, March 2024.

Shicong Cen, Chen Cheng, Yuxin Chen, Yuting Wei, and Yuejie Chi. Fast global convergence of natural policy gradient methods with entropy regularization. *Operations Research*, 2022.

Johan Samir Obando Ceron and Pablo Samuel Castro. Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *International Conference on Machine Learning*, 2021.

Xinyue Chen, Che Wang, Zijian Zhou, and Keith Ross. Randomized ensembled double q-learning: Learning fast without a model. *arXiv preprint arXiv:2101.05982*, 2021.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

M. P. Deisenroth, G. Neumann, and J. Peters. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, 2013.

Riccardo Della Vecchia, Alena Shilova, Philippe Preux, and Riad Akrour. Entropy regularized reinforcement learning with cascading networks. *arXiv*, 2022.

Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. *Advances in neural information processing systems*, 2, 1989.

Roy Fox, Ari Pakman, and Naftali Tishby. G-learning: Taming the noise in reinforcement learning via soft updates. In *Conference on Uncertainty in Artificial Intelligence*, 2016.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying Deep Temporal Difference Learning, March 2025.

Matthieu Geist and Olivier Pietquin. Revisiting natural actor-critics with value function approximation. In *International Conference on Modeling Decisions for Artificial Intelligence*, 2010.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Sertan Girgin and Philippe Preux. Basis function construction in reinforcement learning using cascade-correlation learning architecture. In *IEEE International Conference on Machine Learning and Applications*, 2008.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. In *International Conference on Machine Learning (ICML)*, 2018.

Peter Henderson. Reproducibility and reusability in deep reinforcement learning. Master's thesis, McGill University, 2018.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. A closer look at deep policy gradients. In *International Conference on Learning Representations*, 2020.

Emmeran Johnson, Ciara Pike-Burke, and Patrick Rebeschini. Optimal convergence rate for exact policy mirror descent in discounted markov decision processes. *Advances in Neural Information Processing Systems*, 36:76496–76524, 2023.

Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, 2001.

Sajad Khodadadian, Prakirt Raj Jhunjhunwala, Sushil Mahavir Varma, and Siva Theja Maguluri. On the linear convergence of natural policy gradient algorithm. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 3794–3799. IEEE, 2021.

Toshinori Kitamura and Ryo Yonetani. Shinrl: A library for evaluating rl algorithms from theoretical and practical perspectives. In *Proceedings of the NeurIPS Deep RL Workshop*, 2021.

Tadashi Kozuno, Eiji Uchibe, and Kenji Doya. Theoretical analysis of efficiency and robustness of softmax and gap-increasing operators in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. In *Advances in Neural Information Processing Systems*, 2020.

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 2003.

Guanghui Lan. Policy mirror descent for reinforcement learning: linear convergence, new sampling complexity, and generalized problem classes. *Mathematical Programming*, 2022.

Qingfeng Lan, Yangchen Pan, Alona Fyshe, and Martha White. Maxmin q-learning: Controlling the estimation bias of q-learning. *arXiv preprint arXiv:2002.06487*, 2020.

Nevena Lazic, Dong Yin, Yasin Abbasi-Yadkori, and Csaba Szepesvari. Improved regret bound and experience replay in regularized policy iteration. In *International Conference on Machine Learning*, 2021.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 6131–6141. PMLR, 2021.

Kyungjae Lee, Sungyub Kim, Sungbin Lim, Sungjoon Choi, and Songhwai Oh. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *arXive*, 2019.

Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2020.

Yan Li, Guanghui Lan, and Tuo Zhao. Homotopic policy mirror descent: Policy convergence, implicit regularization, and improved sample complexity. *arXiv preprint arXiv:2201.09457*, 2022.

Jincheng Mei, Chenjun Xiao, Ruitong Huang, Dale Schuurmans, and Martin Müller. On principled entropy exploration in policy optimization. International Joint Conferences on Artificial Intelligence Organization, 2019.

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. Human-level control through deep reinforcement learning. *Nature*, 2015.

A. S. Nemirovsky and D. B. Yudin. *Problem Complexity and Method Efficiency in Optimization*. Wiley-Interscience Series in Discrete Mathematics. John Wiley, 1983.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv*, 2017.

Joni Pajarinen, Hong Linh Thai, Riad Akrour, Jan Peters, and Gerhard Neumann. Compatible natural gradient policy search. *Machine Learning*, 2019.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *International Conference on Intelligent Robots and Systems*, 2006.

Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 2008.

Kimon Protopapas and Anas Barakat. Policy mirror descent with lookahead. *Advances in Neural Information Processing Systems*, 37:26443–26481, 2024.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, 2016.

John Schulman, Sergey Levine, Michael Jordan, and Pieter Abbeel. Trust Region Policy Optimization. *International Conference on Machine Learning*, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

Tim Seyde, Igor Gilitschenski, Wilko Schwarting, Bartolomeo Stellato, Martin Riedmiller, Markus Wulfmeier, and Daniela Rus. Is bang-bang control all you need? solving continuous control with bernoulli policies. In *Advances in Neural Information Processing Systems*, 2021.

Wenjie Shi, Shiji Song, Hui Wu, Ya-Chu Hsu, Cheng Wu, and Gao Huang. Regularized anderson acceleration for off-policy deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 2016.

Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 2002.

Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. The MIT Press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1999.

Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. In *AAAI Conference on Artificial Intelligence*, 2020.

Manan Tomar, Lior Shani, Yonathan Efroni, and Mohammad Ghavamzadeh. Mirror descent policy optimization. In *International Conference on Learning Representations*, 2022.

Samuele Tosatto, Matteo Pirotta, Carlo d'Eramo, and Marcello Restelli. Boosted fitted q-iteration. In *International Conference on Machine Learning*, pp. 3434–3443. PMLR, 2017.

Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Hado van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv*, 2018.

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Remi Munos, and Matthieu Geist. Leverage the average: an analysis of kl regularization in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020a.

Nino Vieillard, Olivier Pietquin, and Matthieu Geist. Munchausen reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020b.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017.

Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in Neural Information Processing Systems*, 2017.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Raj Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller,

Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion gran turismo drivers with deep reinforcement learning. *Nature*, 2022.

Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments. *arXiv preprint arXiv:1903.03176*, 2019.

Rui Yuan, Simon S Du, Robert M Gower, Alessandro Lazaric, and Lin Xiao. Linear convergence of natural policy gradient methods with log-linear policies. *arXiv preprint arXiv:2210.01400*, 2022.

Wenhao Zhan, Shicong Cen, Baihe Huang, Yuxin Chen, Jason D. Lee, and Yuejie Chi. Policy mirror descent for regularized reinforcement learning: A generalized framework with linear convergence. *SIAM Journal on Optimization*, 2023.

Supplementary Materials

The following content was not necessarily subject to peer review.

8 Proofs

This section includes proofs of the lemmas and theorems of the main paper.

8.1 Properties of entropy regularized Bellman operators

We first start with a reminder of some basic properties of the (entropy regularized) Bellman operators, as presented in (Geist et al., 2019; Zhan et al., 2023). Within the MDP setting defined in Sec. 3, let T_τ^π be the operator defined for any map $f : S \times A \mapsto \mathbb{R}$ by

$$\left(T_\tau^\pi f\right)(s,a) = R(s,a) + \gamma \mathbb{E}_{s',a'}[f(s',a') - \tau h(\pi(s'))], \tag{8}$$

This operator has the following three properties.

Proposition 8.1 (Contraction)**.** *T_τ^π is a γ-contraction w.r.t. the $\|.\|_\infty$ norm, i.e. $\|T_\tau^\pi f - T_\tau^\pi g\|_\infty \leq \gamma \|f - g\|_\infty$ for any real functions f and g of $S \times A$.*

Proposition 8.2 (Fixed point)**.** *Q_τ^π is the unique fixed point of the operator T_τ^π, i.e. $T_\tau^\pi Q_\tau^\pi = Q_\tau^\pi$.*

Let f, g be two real functions of $S \times A$. We say that $f \geq g$ iff $f(s,a) \geq g(s,a)$ for all $(s,a) \in S \times A$.

Proposition 8.3 (Monotonicity)**.** *T_τ^π is monotonous, i.e. if $f \geq g$ then $T_\tau^\pi f \geq T_\tau^\pi g$.*

Let the Bellman optimality $T_\tau^\star$ operator be defined by

$$\left(T_\tau^\star f\right)(s,a) = R(s,a) + \gamma \mathbb{E}_{s'}\left[\max_{p \in \Delta(A)} f(s') \cdot p - \tau h(p)\right]. \tag{9}$$

For the Bellman optimality operator we need the following two properties.

Proposition 8.4 (Contraction)**.** *$T_\tau^\star$ is a γ-contraction w.r.t. the $\|.\|_\infty$ norm, i.e. $\|T_\tau^\star f - T_\tau^\star g\|_\infty \leq \gamma \|f - g\|_\infty$ for any real functions f and g of $S \times A$.*

Proposition 8.5 (Optimal fixed point)**.** *$T_\tau^\star$ admits $Q_\tau^\star$ as a unique fixed point, satisfying $T_\tau^\star Q_\tau^\star = Q_\tau^\star$.*

Finally, we will make use of the well known property that the softmax distribution is entropy maximizing (Geist et al., 2019). Specifically, we know that the policy $\pi_k \propto \exp(\xi_k)$ satisfies the following property

$$\text{for all } s \in S, \quad \pi_k(s) = \arg\max_{p \in \Delta(A)} \xi_k(s) \cdot p - h(p). \tag{10}$$

8.2 Proof of Lemma 4.1

Proof. We first observe from the definition of π_k that

$$T_\tau^k \tau\xi_k = R + \gamma P(\pi_k \tau\xi_k - \tau h(\pi_k)), \tag{11}$$

$$\overset{(i)}{=} R + \gamma P(\max_p p\tau\xi_k - \tau h(p)), \tag{12}$$

$$= T_\tau^\star \tau\xi_k, \tag{13}$$

with (i) due to Eq. 10. Then

$$
\begin{aligned}
Q_\tau^k - \tau\xi_k &= (T_\tau^k Q_\tau^k - T_\tau^k \tau\xi_k) + (T_\tau^k \tau\xi_k - \tau\xi_k) && (14)\\
&= \gamma P_k (Q_\tau^k - \tau\xi_k) + (T_\tau^k \tau\xi_k - \tau\xi_k), && (15)\\
&= (I - \gamma P_k)^{-1} (T_\tau^k \tau\xi_k - \tau\xi_k), && (16)\\
&= (I - \gamma P_k)^{-1} (T_\tau^\star \tau\xi_k - \tau\xi_k), && (17)\\
\Rightarrow \left\| Q_\tau^k - \tau\xi_k \right\|_\infty &\leq \frac{1}{1-\gamma} \left\| T_\tau^\star \tau\xi_k - \tau\xi_k \right\|_\infty, && (18)\\
&= \frac{1}{1-\gamma} \left\| T_\tau^\star \tau\xi_k - Q_\tau^\star + Q_\tau^\star - \tau\xi_k \right\|_\infty, && (19)\\
&\leq \frac{1}{1-\gamma} \left(\left\| T_\tau^\star Q_\tau^\star - T_\tau^\star \tau\xi_k \right\|_\infty + \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty \right), && (20)\\
&\leq \frac{1+\gamma}{1-\gamma} \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty. && (21)
\end{aligned}
$$

Finally,

$$
\begin{aligned}
\left\| Q_\tau^\star - Q_\tau^k \right\|_\infty &\leq \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty + \left\| Q_\tau^k - \tau\xi_k \right\|_\infty, && (22)\\
&\leq \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty + \frac{1+\gamma}{1-\gamma} \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty, && (23)\\
&= \frac{2 \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty}{1-\gamma}. && (24)
\end{aligned}
$$

□

8.3 Proof of Theorem 4.2

Proof. Extracting the value function from the policy update formula, and using the fact that $\tau\alpha = 1 - \beta$, we have

$$
\begin{aligned}
\pi_{k+1} q_k - \tau h(\pi_{k+1}) &= \pi_{k+1} \left(\frac{1}{1-\beta} (\tau\xi_{k+1} - \beta\tau\xi_k) \right) - \tau h(\pi_{k+1}), && (25)\\
&= \pi_{k+1} \left(\frac{1}{1-\beta} (\tau\xi_{k+1} - \beta\tau\xi_k) \right) - \frac{1}{1-\beta} (\tau h(\pi_{k+1}) - \beta\tau h(\pi_{k+1})), && (26)\\
&= \pi_{k+1} \left(\frac{1}{1-\beta} [\tau\xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\tau\xi_k - \tau h(\pi_{k+1}))] \right), && (27)\\
&\overset{(i)}{\geq} \left(\frac{1}{1-\beta} [\pi_{k+1}\tau\xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau\xi_k - \tau h(\pi_k))] \right), && (28)
\end{aligned}
$$

with (i) due to $\pi_k \tau\xi_k - \tau h(\pi_k) = \max_p p\tau\xi_k - \tau h(p) \geq \pi_{k+1}\tau\xi_{k+1} - \tau h(\pi_{k+1})$. Using this inequality in q_{k+1} yields

$$q_{k+1} = T_\tau^{k+1} q_k + \epsilon_{k+1}, \tag{29}$$
$$= R + \gamma P(\pi_{k+1} q_k - \tau h(\pi_{k+1})) + \epsilon_{k+1}, \tag{30}$$
$$\geq R + \gamma P\Big(\frac{1}{1-\beta}[\pi_{k+1}\tau\xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau\xi_k - \tau h(\pi_k))]\Big) + \epsilon_{k+1}, \tag{31}$$
$$= \frac{1}{1-\beta}(R - \beta R) + \gamma P\Big(\frac{1}{1-\beta}[\pi_{k+1}\tau\xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau\xi_k - \tau h(\pi_k))]\Big) + \epsilon_{k+1}, \tag{32}$$
$$= \frac{1}{1-\beta}(T_\tau^{k+1}\tau\xi_{k+1} - \beta T_\tau^k \tau\xi_k) + \epsilon_{k+1}, \tag{33}$$
$$= \frac{1}{1-\beta}(T_\tau^\star\tau\xi_{k+1} - \beta T_\tau^\star \tau\xi_k) + \epsilon_{k+1}, \tag{34}$$

with , and where the last step is again due to Eq. 10. Now using this inequality in the definition of ξ_{k+1} gives

$$\tau\xi_{k+1} = (1-\beta)\sum_{i=0}^{k}\beta^{k-i} q_i, \tag{35}$$
$$\overset{(i)}{=} (1-\beta)\sum_{i=1}^{k}\beta^{k-i} q_i, \tag{36}$$
$$\geq (1-\beta)\sum_{i=1}^{k}\beta^{k-i}\left(\frac{1}{1-\beta}(T_\tau^\star\tau\xi_i - \beta T_\tau^\star\tau\xi_{i-1}) + \epsilon_i\right), \tag{37}$$
$$= T_\tau^\star\tau\xi_k - \beta^k T_\tau^\star\tau\xi_0 + (1-\beta)\sum_{i=1}^{k}\beta^{k-i}\epsilon_i, \tag{38}$$

with (i) due to $q_0 = 0$. Letting $E_j = (1-\beta)\sum_{i=1}^{j}\beta^{j-i}\epsilon_i$, , and $R_m = R_x + \gamma\tau\log|A|$ be an upper bound to $\|T_\tau^\star\tau\xi_0\|_\infty$, we finally obtain

$$Q_\tau^\star - \tau\xi_{k+1} \leq Q_\tau^\star - T_\tau^\star\tau\xi_k + \beta^k T_\tau^\star\tau\xi_0 - E_k, \tag{39}$$
$$\Rightarrow \|Q_\tau^\star - \tau\xi_{k+1}\|_\infty \leq \|Q_\tau^\star - T_\tau^\star\tau\xi_k\|_\infty + \beta^k\|T_\tau^\star\tau\xi_0\|_\infty + \|E_k\|_\infty, \tag{40}$$
$$\leq \gamma\|Q_\tau^\star - \tau\xi_k\|_\infty + \beta^k R_m + \|E_k\|_\infty, \tag{41}$$
$$\leq \gamma^{k+1}\|Q_\tau^\star\|_\infty + R_m\sum_{i=0}^{k}\gamma^i\beta^{k-i} + \sum_{i=0}^{k}\gamma^i\|E_{k-i}\|_\infty. \tag{42}$$

if $\beta > \gamma$

$$\sum_{i=0}^{k}\gamma^i\beta^{k-i} = \beta^k\sum_{i=0}^{k}\left(\frac{\gamma}{\beta}\right)^i, \tag{43}$$
$$= \frac{\beta^{k+1} - \gamma^{k+1}}{\beta - \gamma}. \tag{44}$$

if $\beta < \gamma$

$$\sum_{i=0}^{k}\gamma^i\beta^{k-i} = \sum_{i=0}^{k}\gamma^{k-i}\beta^i, \tag{45}$$
$$= \frac{\gamma^{k+1} - \beta^{k+1}}{\gamma - \beta}. \tag{46}$$

if $\beta = \gamma$

$$\sum_{i=0}^{k} \gamma^i \beta^{k-i} = \gamma^k (k+1). \tag{47}$$

In all cases

$$\sum_{i=0}^{k} \gamma^i \beta^{k-i} \leq \max\{\gamma, \beta\}^k (k+1). \tag{48}$$

□

8.4 Proof of Lemma 4.3

Proof. As π_{k+1} maximizes the policy update Eq. 4, and from the non-negativity of the D_{KL} and the fact that $D_{\mathrm{KL}}(\pi_k; \pi_k) = 0$ we have

$$\begin{aligned}
\pi_k q_k - \tau h(\pi_k) &\leq \pi_{k+1} q_k - \tau h(\pi_{k+1}) - \eta D_{\mathrm{KL}}(\pi_{k+1}; \pi_k), && (49)\\
&\leq \pi_{k+1} q_k - \tau h(\pi_{k+1}), && (50)\\
\Leftrightarrow \pi_k Q_\tau^k - \tau h(\pi_k) &\leq \pi_{k+1} Q_\tau^k - \tau h(\pi_{k+1}) + (\pi_{k+1} - \pi_k)\epsilon_k. && (51)
\end{aligned}$$

$$\begin{aligned}
Q_\tau^{k+1} - Q_\tau^k &= \gamma P(\pi_{k+1} Q_\tau^{k+1} - \tau h(\pi_{k+1})) - \gamma P(\pi_k Q_\tau^k - \tau h(\pi_k)), && (52)\\
&\overset{(i)}{\geq} \gamma P(\pi_{k+1} Q_\tau^{k+1} - \tau h(\pi_{k+1})) - \\
&\qquad \gamma P(\pi_{k+1} Q_\tau^k - \tau h(\pi_{k+1}) + (\pi_{k+1} - \pi_k)\epsilon_k), && (53)\\
&= \gamma P \pi_{k+1}(Q_\tau^{k+1} - Q_\tau^k) + \gamma P(\pi_k - \pi_{k+1})\epsilon_k, && (54)\\
\Leftrightarrow (I - \gamma P \pi_{k+1})(Q_\tau^{k+1} - Q_\tau^k) &\geq \gamma P(\pi_k - \pi_{k+1})\epsilon_k, && (55)\\
\Leftrightarrow Q_\tau^{k+1} - Q_\tau^k &\overset{(ii)}{\geq} \gamma (I - \gamma P \pi_{k+1})^{-1} P(\pi_k - \pi_{k+1})\epsilon_k. && (56)
\end{aligned}$$

where in (i) we have used the fact that P is a probability matrix with only positive entries, and similarly in (ii) for the matrix $(I - \gamma P \pi_{k+1})^{-1} = \sum_{i=0}^{\infty} (\gamma P \pi_{k+1})^i$. □

8.5 Proof of Theorem 4.4

Proof. The beginning of the proof is the same as in value iteration and we can show using the same arguments that

$$\pi_{k+1} q_k - \tau h(\pi_{k+1}) \geq \left(\frac{1}{1-\beta} [\pi_{k+1} \tau \xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau \xi_k - \tau h(\pi_k))] \right). \tag{57}$$

Using this inequality in q_{k+1} yields

$$\begin{aligned}
q_{k+1} &= Q_\tau^{k+1} + \epsilon_{k+1}, && (58)\\
&= R + \gamma P(\pi_{k+1} Q_\tau^{k+1} - \tau h(\pi_{k+1})) + \epsilon_{k+1}, && (59)\\
&\overset{(i)}{\geq} R + \gamma P(\pi_{k+1}(q_k - \epsilon_k - \epsilon_{\Delta_k}) - \tau h(\pi_{k+1})) + \epsilon_{k+1}, && (60)\\
&\geq R + \gamma P \left(\frac{1}{1-\beta} [\pi_{k+1} \tau \xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau \xi_k - \tau h(\pi_k))] \right) + \epsilon_{k+1} - \gamma P_{k+1}(\epsilon_k + \epsilon_{\Delta_k}), && (61)\\
&= \frac{1}{1-\beta}(T_\tau^{k+1} \tau \xi_{k+1} - \beta T_\tau^k \tau \xi_k) + \epsilon_{k+1} - \gamma P_{k+1}(\epsilon_k + \epsilon_{\Delta_k}), && (62)\\
&= \frac{1}{1-\beta}(T_\tau^\star \tau \xi_{k+1} - \beta T_\tau^\star \tau \xi_k) + \epsilon_{k+1} - \gamma P_{k+1}(\epsilon_k + \epsilon_{\Delta_k}), && (63)
\end{aligned}$$

where for (i) we used Lemma 4.3 and the definition of q_k. Now using this inequality in the definition of ξ_{k+1} gives

$$\tau\xi_{k+1} = (1-\beta)\sum_{i=0}^{k}\beta^{k-i}q_i, \tag{64}$$

$$\geq (1-\beta)\beta^k q_0 + (1-\beta)\sum_{i=1}^{k}\beta^{k-i}\left(\frac{1}{1-\beta}(T_\tau^\star\tau\xi_i - \beta T_\tau^\star\tau\xi_{i-1}) + \epsilon_i - \gamma P_i(\epsilon_{i-1} + \epsilon_{\Delta_{i-1}})\right), \tag{65}$$

$$= T_\tau^\star\tau\xi_k - \beta^k T_\tau^\star\xi_0 + (1-\beta)\sum_{i=0}^{k}\beta^{k-i}(\epsilon_i - \epsilon_i') + (1-\beta)\beta^k Q_\tau^0, \tag{66}$$

with $\epsilon_0' = 0$ and $\forall i > 0 : \epsilon_i' = \gamma P_i(I + \gamma P(I-\gamma P_i)^{-1}(\pi_{i-1} - \pi_i))\epsilon_{i-1}$. Letting $E_j := (1-\beta)\sum_{i=0}^{j}\beta^{j-i}(\epsilon_i - \epsilon_i')$, $R_m := R_x + \gamma\tau\log|A|$ be an upper bound to $\|T_\tau^\star\xi_0\|_\infty$ and $\bar{R} = \frac{R_m}{1-\gamma}$ be an upper bound to $\left\|Q_\tau^0\right\|_\infty$, we finally obtain

$$Q_\tau^\star - \tau\xi_{k+1} \leq Q_\tau^\star - T_\tau^\star\tau\xi_k + \beta^k T_\tau^\star\xi_0 - E_k + (1-\beta)\beta^k Q_\tau^0, \tag{67}$$

$$\Rightarrow \|Q_\tau^\star - \tau\xi_{k+1}\|_\infty \leq \|Q_\tau^\star - T_\tau^\star\tau\xi_k\|_\infty + \beta^k\|T_\tau^\star\xi_0\|_\infty + \|E_k\|_\infty + (1-\beta)\beta^k\left\|Q_\tau^0\right\|_\infty, \tag{68}$$

$$\leq \gamma\|Q_\tau^\star - \tau\xi_k\|_\infty + \beta^k R_m + (1-\beta)\beta^k\bar{R} + \|E_k\|_\infty, \tag{69}$$

$$\leq \gamma^{k+1}\|Q_\tau^\star\|_\infty + (R_m + (1-\beta)\bar{R})\sum_{i=0}^{k}\gamma^i\beta^{k-i} + \sum_{i=0}^{k}\gamma^i\|E_{k-i}\|_\infty, \tag{70}$$

$$= \gamma^{k+1}\|Q_\tau^\star\|_\infty + \frac{2-\gamma-\beta}{1-\gamma}R_m\sum_{i=0}^{k}\gamma^i\beta^{k-i} + \sum_{i=0}^{k}\gamma^i\|E_{k-i}\|_\infty \tag{71}$$

□

8.6 Proof of Eq. (6)

Proof. For $k = 0$,

$$\xi_1 = \beta\times 0 + \alpha q_0 + \frac{\alpha\beta^M}{1-\beta^M}(q_0 - 0), \tag{72}$$

$$= \alpha\left(1 + \frac{\beta^M}{1-\beta^M}\right)q_0, \tag{73}$$

$$= \frac{\alpha}{1-\beta^M}q_0. \tag{74}$$

If it is true for k, then

$$\xi_{k+1} = \beta\frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i q_{k-1-i} + \alpha q_k + \frac{\alpha\beta^M}{1-\beta^M}(q_k - q_{k-M}), \tag{75}$$

$$= \frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-2}\beta^{i+1}q_{k-1-i} + \frac{\alpha\beta^M}{1-\beta^M}(q_{k-M} - q_{k-M}) + \frac{\alpha}{1-\beta^M}q_k, \tag{76}$$

$$= \frac{\alpha}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i q_{k-i} \tag{77}$$

□

8.7 Proof of Theorem 5.1

As with policy iteration, we first need a policy improvement lemma

Lemma 8.1 (Approximate policy improvement of finite memory EPMD)**.** *For any* $k \geq 0$, $Q_\tau^{k+1} \geq Q_\tau^k - \gamma(I - \gamma P \pi_{k+1})^{-1} P[(\pi_{k+1} - \pi_k)(\epsilon_k + \Delta_k)]$, *with* $\Delta_k := \frac{\beta^M}{1-\beta^M}(q_k - q_{k-M})$.

Proof. We can see the policy π_{k+1} as the maximizer of Eq. (4) if we would replace q_k with $q_k + \frac{\beta^M}{1-\beta^M}(q_k - q_{k-M})$. From the non-negativity of the D_{KL} and the fact that $D_{\mathrm{KL}}(\pi_k; \pi_k) = 0$ we have

$$\pi_k q_k - \tau h(\pi_k) \leq \pi_{k+1} q_k - \tau h(\pi_{k+1}) + \frac{\beta^M}{1-\beta^M}(\pi_{k+1} - \pi_k)(q_k - q_{k-M}), \tag{78}$$

$$\Rightarrow \pi_k Q_\tau^k - \tau h(\pi_k) \leq \pi_{k+1} Q_\tau^k - \tau h(\pi_{k+1}) + \frac{\beta^M}{1-\beta^M}(\pi_{k+1} - \pi_k)(q_k - q_{k-M}) + (\pi_{k+1} - \pi_k)\epsilon_k. \tag{79}$$

Let $\Delta_k := \frac{\beta^M}{1-\beta^M}(q_k - q_{k-M})$. Writing down the (matrix) definition of Q_τ^{k+1} and Q_τ^k gives

$$Q_\tau^{k+1} - Q_\tau^k = \gamma P(\pi_{k+1} Q_\tau^{k+1} - \tau h(\pi_{k+1})) - \gamma P(\pi_k Q_\tau^k - \tau h(\pi_k)), \tag{80}$$

$$\overset{(i)}{\geq} \gamma P(\pi_{k+1} Q_\tau^{k+1} - \tau h(\pi_{k+1})) - \gamma P(\pi_{k+1} Q_\tau^k - \tau h(\pi_{k+1}) + (\pi_{k+1} - \pi_k)(\epsilon_k + \Delta_k)), \tag{81}$$

$$= \gamma P \pi_{k+1}(Q_\tau^{k+1} - Q_\tau^k) - \gamma P[(\pi_{k+1} - \pi_k)(\epsilon_k + \Delta_k)], \tag{82}$$

$$\Leftrightarrow Q_\tau^{k+1} - Q_\tau^k \overset{(ii)}{\geq} -\gamma(I - \gamma P \pi_{k+1})^{-1} P[(\pi_{k+1} - \pi_k)(\epsilon_k + \Delta_k)]. \tag{83}$$

where in (i) we have used the fact that P is a probability matrix with only positive entries, and similarly in (ii) for the matrix $(I - \gamma P \pi_{k+1})^{-1} = \sum_{i=0}^{\infty} (\gamma P \pi_{k+1})^i$. □

We are now ready to prove the main theorem

Proof. The beginning of the proof is similar to vanilla entropy regularized policy/value iteration

$$\pi_{k+1} q_k - \tau h(\pi_{k+1}) = \pi_{k+1} \left(\frac{1}{1-\beta}(\tau \xi_{k+1} - \beta \tau \xi_k) \right) - \tau h(\pi_{k+1}) - \pi_{k+1} \Delta_k, \tag{84}$$

$$= \pi_{k+1} \left(\frac{1}{1-\beta}[\tau \xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\tau \xi_k - \tau h(\pi_{k+1}))] \right) - \pi_{k+1} \Delta_k, \tag{85}$$

$$\overset{(i)}{\geq} \frac{1}{1-\beta}[\pi_{k+1} \tau \xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau \xi_k - \tau h(\pi_k))] - \pi_{k+1} \Delta_k. \tag{86}$$

Using this inequality in q_{k+1} yields

$$q_{k+1} = Q_\tau^{k+1} + \epsilon_{k+1}, \tag{87}$$

$$= R + \gamma P(\pi_{k+1} Q_\tau^{k+1} - \tau h(\pi_{k+1})) + \epsilon_{k+1}, \tag{88}$$

$$\overset{(i)}{\geq} R + \gamma P(\pi_{k+1}(q_k - \epsilon_k - \gamma \mu_{k+1} P(\pi_{k+1} - \pi_k)(\epsilon_k + \Delta_k)) - \tau h(\pi_{k+1})) + \epsilon_{k+1}, \tag{89}$$

$$\tag{90}$$

where for (i) we used Lemma 4.3 and the definition of q_k. Using Eq. (86) on the following terms gives

$$R + \gamma P(\pi_{k+1} q_k - \tau h(\pi_{k+1})) \geq R + \gamma P\Big(\frac{1}{1-\beta}[\pi_{k+1}\tau\xi_{k+1} - \tau h(\pi_{k+1}) - \beta(\pi_k \tau\xi_k - \tau h(\pi_k))] - \pi_{k+1}\Delta_k\Big), \tag{91}$$

$$= \frac{1}{1-\beta}(T_\tau^{k+1}\tau\xi_{k+1} - \beta T_\tau^k \tau\xi_k) - \gamma P\pi_{k+1}\Delta_k, \tag{92}$$

$$= \frac{1}{1-\beta}(T_\tau^\star\tau\xi_{k+1} - \beta T_\tau^\star \tau\xi_k) - \gamma P\pi_{k+1}\Delta_k. \tag{93}$$

Completing with the rest of the terms finally gives

$$q_{k+1} \geq \frac{1}{1-\beta}(T_\tau^\star\tau\xi_{k+1} - \beta T_\tau^\star\tau\xi_k) - \gamma P\pi_{k+1}(I + \gamma\mu_{k+1}P(\pi_{k+1} - \pi_k))(\epsilon_k + \Delta_k) + \epsilon_{k+1}. \tag{94}$$

$$\tag{95}$$

Let $A_{k+1} = \gamma P\pi_{k+1}(I + \gamma\mu_{k+1}P(\pi_{k+1} - \pi_k))$ using this inequality in the definition of ξ_{k+1} gives

$$\tau\xi_{k+1} = \frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i q_{k-i}, \tag{96}$$

$$\geq \frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i\left(\frac{1}{1-\beta}(T_\tau^\star\tau\xi_{k-i} - \beta T_\tau^\star\tau\xi_{k-i-1}) - A_{k-i}(\epsilon_{k-i-1} + \Delta_{k-i-1}) + \epsilon_{k-i}\right), \tag{97}$$

$$= \frac{1}{1-\beta^M}T_\tau^\star\tau\xi_k - \frac{\beta^M}{1-\beta^M}T_\tau^\star\xi_{k-M} + \frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i(\epsilon_{k-i} - A_{k-i}(\epsilon_{k-i-1} + \Delta_{k-i-1})). \tag{98}$$

Letting $E_j := \frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i(\epsilon_{k-i} - A_{k-i}\epsilon_{k-i-1})$, $\Delta_q^k = \frac{1-\beta}{1-\beta^M}\sum_{i=0}^{M-1}\beta^i A_{k-i}\Delta_{k-i-1}$, $R_m := R_x + \gamma\tau\log|A|$ be an upper bound to $\|T_\tau^\star\xi_0\|_\infty$ and $\bar{R} = \frac{R_m}{1-\gamma}$ be an upper bound to $\left\|Q_\tau^0\right\|_\infty$, we finally obtain

$$Q_\tau^\star - \tau\xi_{k+1} \leq Q_\tau^\star - \frac{1}{1-\beta^M}T_\tau^\star\tau\xi_k + \frac{\beta^M}{1-\beta^M}T_\tau^\star\xi_{k-M} - E_k - \Delta_q^k, \tag{99}$$

$$\Rightarrow \|Q_\tau^\star - \tau\xi_{k+1}\|_\infty \leq \frac{\|Q_\tau^\star - T_\tau^\star\tau\xi_k\|_\infty + \beta^M\|Q_\tau^\star - T_\tau^\star\tau\xi_{k-M}\|_\infty}{1-\beta^M} + \left\|\Delta_q^k\right\|_\infty + \|E_k\|_\infty, \tag{100}$$

$$\leq \gamma\frac{\|Q_\tau^\star - \tau\xi_k\|_\infty + \beta^M\|Q_\tau^\star - \tau\xi_{k-M}\|_\infty}{1-\beta^M} + \left\|\Delta_q^k\right\|_\infty + \|E_k\|_\infty \tag{101}$$

Let us now look into the term $\left\|\Delta_q^k\right\|_\infty$, and split it into policy evaluation error and distance to $Q_\tau^\star$

$$\left\|\Delta_q^k\right\|_\infty = \left\| \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i A_{k-i} \Delta_{k-i-1} \right\|_\infty , \tag{102}$$

$$= \left\| \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i A_{k-i} \frac{\beta^M}{1-\beta^M} (q_{k-i-1} - q_{k-i-1-M}) \right\|_\infty , \tag{103}$$

$$= \left\| \frac{\beta^M(1-\beta)}{(1-\beta^M)^2} \sum_{i=0}^{M-1} \beta^i A_{k-i} (Q_{k-i-1} - Q_{k-i-1-M} + \epsilon_{k-i-1} - \epsilon_{k-i-1-M}) \right\|_\infty , \tag{104}$$

$$\begin{aligned} &\leq \left\| \frac{\beta^M(1-\beta)}{(1-\beta^M)^2} \sum_{i=0}^{M-1} \beta^i A_{k-i} (\epsilon_{k-i-1} - \epsilon_{k-i-1-M}) \right\|_\infty \\ &\quad + \left\| \frac{\beta^M(1-\beta)}{(1-\beta^M)^2} \sum_{i=0}^{M-1} \beta^i A_{k-i} (Q_{k-i-1} - Q_{k-i-1-M}) \right\|_\infty . \end{aligned} \tag{105}$$

To bound the infinite norm of A_k, we note that $(1-\gamma)\mu_k$ is a probability matrix (the state distribution induced by policy π_k). Using the sub-additivity of norms and the fact that the multiplication of probability matrices is a probability matrix with infinite norm equal to 1, we have

$$\|A_k\|_\infty = \left\| \gamma P \pi_{k+1} \left(I + \frac{\gamma}{1-\gamma} ((1-\gamma)\mu_{k+1}) P(\pi_{k+1} - \pi_k) \right) \right\|_\infty , \tag{106}$$

$$\leq \gamma + \frac{2\gamma^2}{1-\gamma}. \tag{107}$$

Looking now at the rightmost inner sum in Eq. (105) gives

$$\left\| \sum_{i=0}^{M-1} \beta^i A_{k-i} (Q_{k-i-1} - Q_{k-i-1-M}) \right\|_\infty \tag{108}$$

$$\leq \sum_{i=0}^{M-1} \beta^i \left\| A_{k-i} (Q_{k-i-1} - Q_{k-i-1-M}) \right\|_\infty , \tag{109}$$

$$\leq \left(\gamma + \frac{2\gamma^2}{1-\gamma} \right) \sum_{i=0}^{M-1} \beta^i \left\| Q_{k-i-1} - Q_{k-i-1-M} \right\|_\infty , \tag{110}$$

$$\leq \left(\gamma + \frac{2\gamma^2}{1-\gamma} \right) \sum_{i=0}^{M-1} \beta^i \left(\left\| Q_\tau^\star - Q_{k-i-1} \right\|_\infty + \left\| Q_\tau^\star - Q_{k-i-1-M} \right\|_\infty \right) , \tag{111}$$

$$\overset{(i)}{\leq} \frac{2}{1-\gamma} \left(\gamma + \frac{2\gamma^2}{1-\gamma} \right) \sum_{i=0}^{M-1} \beta^i \left(\left\| Q_\tau^\star - \tau\xi_{k-i-1} \right\|_\infty + \left\| Q_\tau^\star - \tau\xi_{k-i-1-M} \right\|_\infty \right) , \tag{112}$$

where (i) is due to Lemma 4.1. Let $z_k = \left\| Q_\tau^\star - \tau\xi_k \right\|_\infty$ and T_k grouping all the error terms

$$T_k = \|E_k\|_\infty + \frac{\beta^M(1-\beta)}{(1-\beta^M)^2} \left\| \sum_{i=0}^{M-1} \beta^i A_{k-i} (\epsilon_{k-i-1} - \epsilon_{k-i-1-M}) \right\|_\infty . \tag{113}$$

Putting everything together we have

$$z_{k+1} \leq \gamma \frac{z_k + \beta^M z_{k-M}}{1-\beta^M} + \frac{\beta^M(1-\beta)}{(1-\beta^M)^2} \frac{2(\gamma+\gamma^2)}{(1-\gamma)^2} \sum_{i=0}^{M-1} \beta^i \left(z_{k-i-1} + z_{k-i-1-M} \right) + T_k. \tag{114}$$

Let us first study the sequence $\{z_k\}$ without policy evaluation errors and try to upper bound it with a simpler sequence. We define the sequence $\{x_k\}$ for all integers k by

$$x_k = \|Q_\tau^\star\|_\infty\,, \text{ for all } k \leq 0, \tag{115}$$

and for $k \geq 0$ we let

$$x_{k+1} = \gamma\frac{x_k + \beta^M x_{k-M}}{1-\beta^M} + \frac{\beta^M(1-\beta)}{(1-\beta^M)^2}\frac{2(\gamma+\gamma^2)}{(1-\gamma)^2}\sum_{i=0}^{M-1}\beta^i\left(x_{k-i-1} + x_{k-i-1-M}\right). \tag{116}$$

We first find a condition for which the sequence is strictly decreasing starting from $k \geq 0$. For $k = 0$ we have that

$$x_1 = \left(\gamma\frac{1+\beta^M}{1-\beta^M} + \frac{\beta^M}{1-\beta^M}\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2}\right)x_0. \tag{117}$$

This will be strictly decreasing if

$$\gamma\frac{1+\beta^M}{1-\beta^M} + \frac{\beta^M}{1-\beta^M}\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2} < 1, \tag{118}$$

$$\Leftrightarrow \gamma(1+\beta^M) + \beta^M\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2} < (1-\beta^M), \tag{119}$$

$$\Leftrightarrow \left(1+\gamma+\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2}\right)\beta^M < 1-\gamma, \tag{120}$$

$$\Leftrightarrow \log\left(1+\gamma+\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2}\right) + M\log\beta < \log(1-\gamma), \tag{121}$$

$$\Leftrightarrow M > \log\frac{(1-\gamma)^3}{(1+\gamma)(1-\gamma)^2+4(\gamma+\gamma^2)}/\log\beta. \tag{122}$$

As it is true for $k = 0$ and the sequence is constant for $k < 0$, assume now that the sequence is strictly decreasing from there on, up to some positive index k. Then since all the weights of past terms are positive, we can replace all terms by their predecessors and we immediately have that

$$x_{k+1} < \gamma\frac{x_{k-1} + \beta^M x_{k-M-1}}{1-\beta^M} + \frac{\beta^M(1-\beta)}{(1-\beta^M)^2}\frac{2(\gamma+\gamma^2)}{(1-\gamma)^2}\sum_{i=0}^{M-1}\beta^i\left(x_{k-i-2} + x_{k-i-2-M}\right), \tag{123}$$

$$= x_k. \tag{124}$$

Thus the sequence $\{x_k\}$ is non-increasing for all k if M satisfies the inequality in (122). For such values of M we will now find an upper bounding sequence that has a simpler geometric form. Letting $c = \frac{\beta^M}{1-\beta^M}\left(\frac{4(\gamma+\gamma^2)}{(1-\gamma)^2} + \gamma\right)$, and since the sequence is non-decreasing, we have for all $k \geq 0$ that

$$x_{k+1} \leq \frac{\gamma}{1-\beta^M}x_k + cx_{k-2M}. \tag{125}$$

Let us now try to find a rate $d \in (0,1)$ such that for all k we have

$$x_k \leq d^k x_0. \tag{126}$$

For all $k \leq 0$, the above inequality holds for any $d \in (0,1)$. Now, if the upper bounding is true up to some index k then using Eq. (125) we have

$$x_{k+1} \leq \left(\frac{\gamma}{1-\beta^M}d^k + cd^{k-2M}\right)x_0, \tag{127}$$

$$= d^k\left(\frac{\gamma}{1-\beta^M} + cd^{-2M}\right)x_0. \tag{128}$$

The smallest acceptable d would be one such that

$$\frac{\gamma}{1-\beta^M} + cd^{-2M} = d, \tag{129}$$

$$\Leftrightarrow d^{2M+1} - \frac{\gamma}{1-\beta^M} d^{2M} - c = 0. \tag{130}$$

Let $f(d) = d^{2M+1} - \frac{\gamma}{1-\beta^M} d^{2M} - c$, we have that $f(\frac{\gamma}{1-\beta^M}) = -c < 0$ and that $f(1) = 1 - \frac{\gamma}{1-\beta^M} - c > 0$ from the above condition on M. Since f is continuous and increasing between these two values it accepts a unique root $d_0 \in (\frac{\gamma}{1-\beta^M}, 1)$ which would satisfy the sought geometric sequence upper bound for all k.

We now turn to the part of the sequence of z_k that depends on the error terms T_k. Define for $k \leq 0$

$$y_k = 0, \tag{131}$$

and for $k \geq 0$

$$y_{k+1} = \gamma \frac{y_k + \beta^M y_{k-M}}{1-\beta^M} + \frac{\beta^M(1-\beta)}{(1-\beta^M)^2} \frac{2(\gamma+\gamma^2)}{(1-\gamma)^2} \sum_{i=0}^{M-1} \beta^i \left(y_{k-i-1} + y_{k-i-1-M}\right) + T_k. \tag{132}$$

Let $\bar{T} = \max_{0 \leq i \leq k} T_i$, and $\gamma_M = \frac{\gamma}{1-\beta^M}$. Then we have that

$$y_k \leq \frac{\bar{T}}{1-(\gamma_M + c)}. \tag{133}$$

Indeed, it is true for $k \leq 0$, and if it is true up to k then

$$y_{k+1} \leq (\gamma_M + c) \frac{\bar{T}}{1-(\gamma_M + c)} + T_k, \tag{134}$$

$$\leq (\gamma_M + c) \frac{\bar{T}}{1-(\gamma_M + c)} + \bar{T}, \tag{135}$$

$$= \frac{\bar{T}}{1-(\gamma_M + c)}. \tag{136}$$

Replacing in Eq. (132), we have

$$y_{k+1} \leq \gamma_M y_k + c \frac{\bar{T}}{1-(\gamma_M + c)} + T_k, \tag{137}$$

$$\leq \gamma_M^{k+1} y_0 + \sum_{i=0}^{k} \gamma_M^i \left(T_{k-i} + c \frac{\bar{T}}{1-\gamma_M - c}\right), \tag{138}$$

$$= \sum_{i=0}^{k} \gamma_M^i \left(T_{k-i} + c \frac{\bar{T}}{1-\gamma_M - c}\right). \tag{139}$$

Finally, because the upper bound z_{k+1} in Eq. (114) has linear dependencies on previous z_i terms ($i \leq k$), we immediately have that $z_k \leq x_k + y_k$. Indeed, it is true for $k = 0$, since $z_0 = \|Q_\tau^\star\|_\infty = x_0 + y_0$. And if we assume that it is true for k, using Eq. (114), we immediately have that it is true for $k+1$. Thus

$$z_{k+1} \leq x_{k+1} + y_{k+1}, \tag{140}$$

$$\leq d_0^{k+1} \|Q_\tau^\star\|_\infty + \sum_{i=0}^{k} \gamma_M^i \left(T_{k-i} + c \frac{\bar{T}}{1-\gamma_M - c}\right). \tag{141}$$

□

9 Additional Experiments

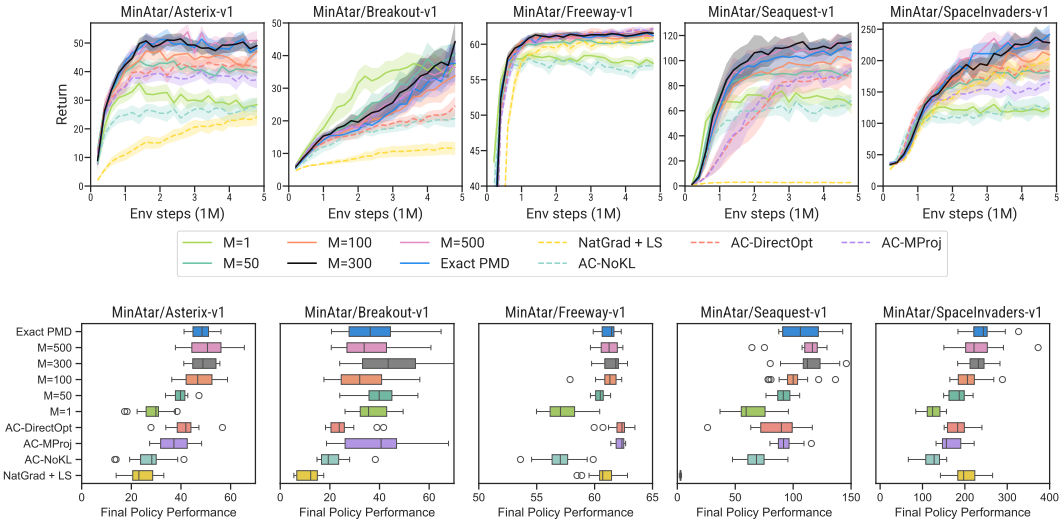


Figure 3: Evaluation of StaQ for different memory sizes M, as well as AC PMD baselines, on MinAtar environments. Learning curves (**top**) show mean and 95% confidence interval for 20 seeds, and box plots (**bottom**) of final policy performance.

9.1 Learning curves and final performance box plots

Figure 3 shows the mean and 95% confidence interval of the performance of StaQ for different choices of M. Setting $M = 1$ corresponds to no KL-regularization as discussed in App. 10.1 and can be seen as an adaptation of SAC to discrete action spaces. $M = 1$ can be unstable and lead to performance collapse. Adding KL-regularization and increasing M helps to improve performance, but in general we note diminishing returns in increasing M, with $M = 100$ matching the performance of exact EPMD in most environments and $M = 300$ being virtually indistinguishable from exact EPMD. Compared to natural policy gradient and other AC methods, performance is generally improved on most tasks, making StaQ a potentially better alternative to other PMD implementations, at least on environments where Q-function computations can be efficiently batched.

9.2 Comparison with deep RL baselines

We summarize all performance comparisons with the deep RL baselines in Fig. 4 and Table 1. We provide a discussion of the MountainCar environment and some of the challenges of exploration in an entropy-regularized setting in App. 9.3.

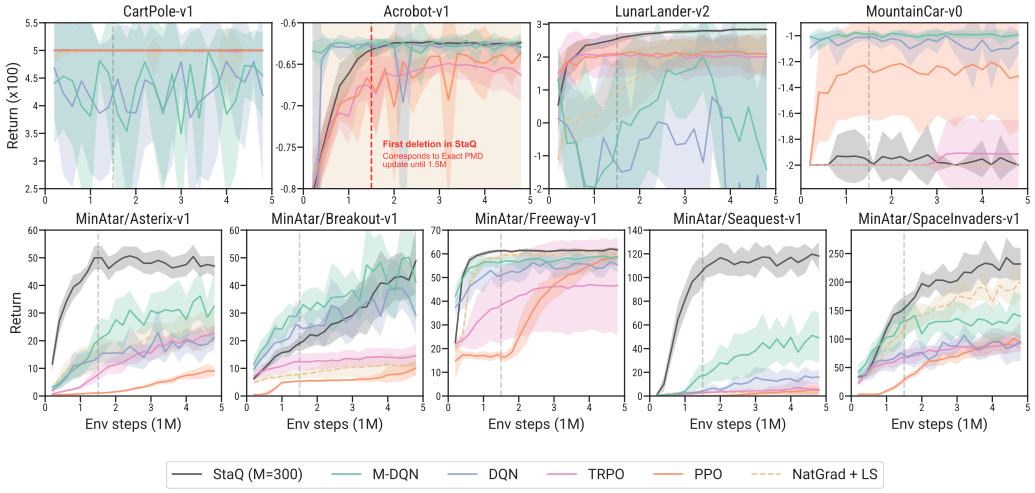


Figure 4: Policy performance of StaQ ($M = 300$) vs deep RL baselines across all environments. Results showing the mean and standard deviation across 10 seeds. Note that for $M = 300$ the first deletion occurs at 1.5M timesteps – **this deletion has no visible impact on policy performance**.

	StaQ (M=300)	DQN	M-DQN	PPO	TRPO	NatGrad + LS
CartPole-v1	**500**	418	454	**500**	**500**	497
Acrobot-v1	**-63**	**-63**	**-63**	-64	-66	-97
LunarLander-v2	**283**	-343	-142	209	201	240
MountainCar-v0	-199	-105	**-99**	-131	-191	-200
MinAtar/Asterix-v1	**49**	21	32	9	23	24
MinAtar/Breakout-v1	**44**	29	41	10	15	11
MinAtar/Freeway-v1	**62**	55	59	59	47	61
MinAtar/Seaquest-v1	**115**	16	49	5	5	3
MinAtar/SpaceInvaders-v1	**229**	96	140	94	93	203

Table 1: Final performance on all environments comparing StaQ ($M = 300$) and Deep RL baselines, as well as `NatGrad + LS` for comparison with TRPO.

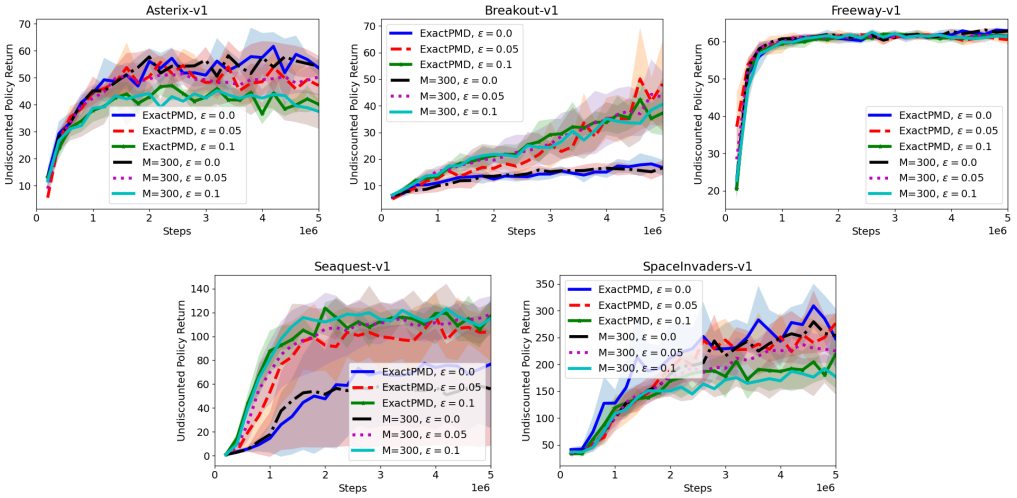


Figure 5: Impact of the ϵ exploration parameter of StaQ on performance in MinAtar tasks. Plots showing mean and std. deviation over 5 seeds.

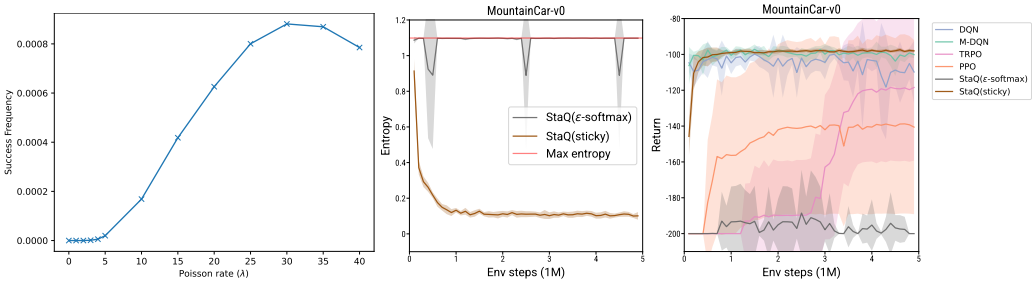


Figure 6: **Left:** Frequency of non-zero rewards of a uniform policy with sticky actions for different choice of Poisson rate λ on MountainCar over 5M timesteps. **Middle:** Entropy of learned policies under different behavior policies. Entropy of the uniform (Max entropy) policy plotted for reference. **Right:** Policy returns for StaQ with different behavior policies and deep RL baselines on MountainCar. Adding sticky actions to StaQ's behavior policy fixes its performance on this task.

9.3 Entropy regularization does not solve exploration

The behavioural policy π_k^b used for exploration is based on sampling from the softmax policy with logits ξ_k. However, we observe that some environments benefit from the addition of a constant probability $\epsilon = 0.05$ of sampling a uniformly-random action throughout the learning phase. This observation that uniform sampling can give rise to better exploration that softmax policies has been made previously, especially in sparse-reward settings such as Atari. See e.g. the discussion in App. B2 in the M-DQN paper Vieillard et al. (2020b). In this appendix we ablate the choice of this ϵ-softmax policy and discuss modified behavioural policies.

In a set of additional experiments we evaluate the impact ϵ has on performance. To this end, we launch StaQ with $M = 300$ and Exact PMD on MinAtar. We see that ϵ exploration is important for both Breakout and Seaquest, but has less of an impact on other tasks. In fact, on Asterix and SpaceInvaders, increasing the epsilon decreases the performance slightly, perhaps because of the increase to the off-policiness of the data which might deteriorate policy evaluation. As such, our choice of $\epsilon = 0.05$ in the main experiment appears to provide a good trade-off between improving exploration and limiting the off-policiness of data on MinAtar tasks. In all cases, independent of the choice of ϵ, StaQ with $M = 300$ remains close to Exact PMD in performance.

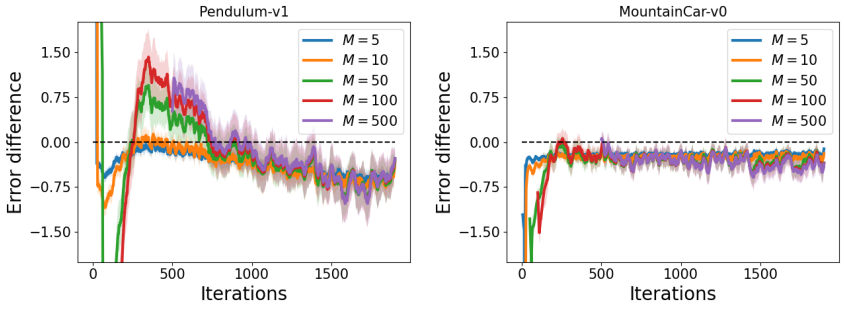


Figure 7: Policy evaluation difference $\left\| \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \epsilon_{k-i} \right\|_\infty - \|\epsilon_k\|_\infty$ between the weighted average of the last M evaluation errors versus the last evaluation error ϵ_k, for different values of M, for a PMD value iteration scheme. As the number of iterations increases, we observe an averaging of policy evaluation errors effect due to the D_{KL} regularization even on simple classic control tasks. Results averaged over 50 seeds, showing mean and 95% confidence interval. For readability, the results are smoothed with a moving average of window size 20.

StaQ and exact EPMD achieve strong performance on all 9 environments except on MountainCar where they fail to learn. In this section, we perform additional experiments to understand the failure of StaQ on MountainCar. In short, it appears that the initial uniform policy—which has maximum entropy—acts as a strong (local) attractor for this task: StaQ starts close to the uniform policy, and exploration with this policy does not generate a reward signal in MountainCar. As StaQ does not observe a reward signal in early training, it quickly converges to the uniform policy which has maximum entropy, but also has extremely small probability of generating a reward signal. Indeed, we unrolled a pure uniform policy on MountainCar for 5M steps, and never observed a reward.

However, StaQ is not limited to a specific choice of behavior policy, and choosing a policy that introduces more correlation between adjacent actions, like a simple "sticky" policy allows StaQ to solve MountainCar. This policy samples an action from π_k and applies it for a few consecutive steps, where a number of steps is drawn randomly from Poisson(λ) distribution (in our experiments with StaQ we fix the rate of Poisson distribution at $\lambda = 10$). In Fig. 6, we can see that StaQ with the same hyperparameters for classical environments (see Table 2) and a "sticky" behavior policy manages to find a good policy for MountainCar matching the best baseline. The final policy demonstrates much lower entropy compared to the default policy that fails at learning for this environment. We focused on this paper on the benefits of entropy regularization in averaging evaluation error. While it is believed that entropy might help with exploration, these observations are a good reminder that entropy regularization remains a heuristic exploration strategy that does not replace a theoretically grounded strategy, which is beyond the scope of this paper that only focuses on using entropy regularization to reduce the error floor of approximate policy iteration when using function approximators.

9.4 Averaging of evaluation error

We explore in this section whether the averaging effect of the evaluation errors due to the D_{KL} regularization does occur in practice. While we would want the per iteration evaluation errors $\{\epsilon_i\}_{i=1}^k$ to behave like uncorrelated random noise, in practice in a deep RL setting, the neural network at the current iteration is often reused to estimate the Q-function at the next iteration. Datasets also often overlap in replay buffers for the estimation of successive Q-function, creating correlations between successive Q-functions and their successive errors. Taking all this into account, it becomes unclear whether a cancellation of evaluation errors does indeed occur in practice. To evaluate this effect, we make use of the ShinRL library (Kitamura & Yonetani, 2021) that allows the computation of the exact Q-function for small scale environments such as Pendulum and MountainCar. The library operates by internally building a discrete MDP, and associating to each discrete state an auxiliary vectorial observation that can be processed by a function approximator.

Using these environments, we implement a PMD value iteration algorithm following exactly the definition of Sec. 4.1, using a neural network q_k for approximating the current soft Q-function Q_τ^k, but using exact policy update on the discrete states. The learning of q_k follows a standard procedure where a target is computed using q_{k-1} and regressed for 5000 gradient steps. The policy is then updated using $\eta = 1$ and $\tau = 0.03$. At each iteration, we compute the exact Q-function Q_τ^k and the evaluation error ϵ_k using the internally discrete MDP.

To evaluate the potential cancellation of errors, we then compute the difference between $\left\| \frac{1-\beta}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i \epsilon_{k-i} \right\|_\infty - \|\epsilon_k\|_\infty$, i.e. between the weighted average of the last M evaluation errors versus the last evaluation error ϵ_k, for different values of M. Fig. 7 shows that as the number of iterations increases, the averaging of policy evaluation error does drop below the last error ϵ_k, even for these simple environments.

10 Connection to Actor-Critic and PMD baselines

In this section, we describe how the baselines SAC (Haarnoja et al., 2018), TRPO (Schulman et al., 2015), MDPO (Tomar et al., 2022) and ECPO (Mei et al., 2019) connect to StaQ, and more generally to the Entropy-regularised Policy Mirror Descent Objective (4).

10.1 Soft Actor Critic

By setting $M = 1$ in StaQ, we remove the KL-divergence regularization and only keep the entropy bonus. This baseline can also be seen as an adaptation of SAC to discrete action spaces: indeed, if we set $M = 1$ in Eq. (2) we recover the policy logits

$$\xi_{k+1} = \frac{\alpha}{1-\beta^M} \sum_{i=0}^{M-1} \beta^i Q_\tau^{k-i} \tag{142}$$

$$= \frac{\alpha}{1-\beta} Q_\tau^k \tag{143}$$

$$= \frac{Q_\tau^k}{\tau}, \tag{144}$$

where the last line is due to $\alpha\tau = 1 - \beta$. This results in a policy of the form $\pi_{k+1} \propto \exp\left(\frac{Q_\tau^k}{\tau}\right)$. Meanwhile, for SAC, the actor network is obtained by minimizing the following problem (Eq. 14 in Haarnoja et al. (2018)) for states sampled from some replay buffer $\mathcal{D}$

$$\pi_{k+1} = \arg\min_\pi \mathbb{E}_{s\sim\mathcal{D}} \left[\mathrm{KL}\left(\pi(s) \middle| \frac{\exp\left(\frac{Q_\tau^k(s)}{\tau}\right)}{Z_{\text{norm.}}} \right) \right] \tag{145}$$

However, in the discrete action setting, we can sample directly from $\exp\left(\frac{Q_\tau^k}{\tau}\right)$—which is the minimizer of the above KL-divergence term—and we do not need an explicit actor network. As such StaQ with $M = 1$ could be seen as an adaptation of SAC to discrete action spaces. Nevertheless, we also consider as a baseline an actor-critic version that updates the actor by attempting to solve Eq. 145 via gradient descent, resulting in an even closer baseline to SAC. We label this baseline `AC-NoKL (SAC)`.

10.2 Approximate Policy Mirror Descent

In the presence of function approximation, simple updates in logit space of a Boltzmann policy do not in general translate to simple updates in the Q-function parameter space. Let Θ_Q and Θ_π respectively be the parameter spaces of Q-functions and policy logits; these parameter spaces can for instance be subsets of $\mathbb{R}^d$ for some integer d. Let $\xi_\theta : S \times A \mapsto \mathbb{R}$, with $\theta \in \Theta_\pi$ be a function

that provides the logits of a policy $\pi_\theta \propto \exp(\xi_\theta)$ and let $q_{\theta'} : S \times A \mapsto \mathbb{R}$, with $\theta' \in \Theta_\pi$, be the (approximate) Q-function associated to π_θ. We want to find $\xi_{\theta''}$ as the solution to the entropy regularized policy update in Eq. 4. We know that ideally we would have $\xi_{\theta''} = \beta\xi_\theta + \alpha q_{\theta'}$, but this does not give an expression for θ'' since in general the policy maximizing Eq. 4 is not necessarily parameterized by $\beta\theta + \alpha\theta'$.

One exception to the above claim is when the Q-function and the logits function are linear w.r.t. some predefined feature function, in which case $\xi_{\beta\theta+\alpha\theta'} = \beta\xi_\theta + \alpha q_{\theta'}$. This is the so-called compatible feature setting of policy gradient (Sutton et al., 1999; Geist & Pietquin, 2010; Pajarinen et al., 2019) where the Q-function and the logits share the same linear-in-feature function approximation class and in which case, policy gradient and natural policy gradient become equivalent (Peters & Schaal, 2008). However, beyond the linear-in-feature case, the closed form solution of the entropy regularized policy update in the space of logits does not yield a trivial update in parameter space.

One approach to policy update in parameter space is to solve the following optimization problem

$$\arg\max_{\theta'' \in \Theta_\pi} \theta'' \cdot \nabla_\theta J_\tau(\pi_\theta) - \eta \mathbb{E}_{s\sim\mu_{\pi_\theta}} D_{\mathrm{KL}}(\pi_{\theta''}(s); \pi_\theta(s)), \tag{146}$$

where $J_\tau(\pi_\theta)$ is the policy return π_θ, i.e. the expectation of the V^{π_θ} over states sampled from a predefined initial state distribution. Interestingly, Cen et al. (2022) showed that in the tabular case (i.e. when optimizing directly over the logit space), the above problem is equivalent to the maximization in Eq. 4 over each state independently. In the general case however, because of the approximation in the optimization of Eq. 146 or because of the use of a restricted policy class, we are likely to obtain a new policy $\pi_{\theta''}$ that is worse than π_{k+1} in the sense of the policy update objective defined in Eq. 4.

10.3 Approximate PMD Algorithms

We now describe several approximate algorithms for solving either the original PMD objective (4) or a state-averaged formulation thereof.

NatGrad + LS (TRPO). We implement `NatGrad + LS (TRPO)` that updates the policy by approximately solving Eq. 146 using TRPO's conjugate gradient approach. We use a post-update line search step that constrains the D_{KL} between successive policies to be under an ϵ_{KL} threshold. This baseline has subtle differences with TRPO: for instance even when an entropy bonus is used, TRPO does not learn soft value functions and only regularizes the policy update with an additional entropy term. We found that by matching StaQ's setting, "NatGrad + LS" is comparable in performance to `stable-baselines3`'s TRPO on most of the tasks but can outperform it on others, especially on MinAtar tasks, as seen from the results in App. 9.2.

AC-MProj (ECPO). The entropy regularized PMD solution is $\pi_{k+1} \propto \pi_k^\beta \exp\left(\alpha Q_\tau^k\right)$. Tracking this recursive policy definition exactly results in an infinite sum, but we can instead approximate $\pi_k^\beta \exp\left(\alpha Q_\tau^k\right)$ with an actor network. Specifically, we consider a second baseline inspired by ECPO (Mei et al., 2019) that advocates for minimizing the M-Projection between the actor and the PMD solution

$$\pi_{k+1} = \arg\min_\pi \mathbb{E}_{s\sim\mathcal{D}} \left[\mathrm{KL}\left(\frac{\pi_k^\beta \exp\left(\alpha Q_\tau^k\right)}{Z_{\text{norm.}}} \middle| \pi(s) \right) \right], \tag{147}$$

where states are again sampled from a replay buffer $\mathcal{D}$. This differs from the forward-KL objective in e.g. SAC; compare with (145). The authors of ECPO argue that the M-Projection will better preserve the support of the distribution and prevent the premature elimination of actions during exploration. We will label this baseline `AC-MProj (ECPO)`.

AC-DirectOpt (MDPO) Finally, we consider an actor-critic baseline inspired by MDPO (Tomar et al., 2022) that optimizes a state averaged version of the entropy regularized PMD update, i.e.

$$\pi_{k+1} = \arg\max_\pi \mathbb{E}_{s\sim\mathcal{D}} \left[Q_\tau^k(s) \cdot \pi(s) - \tau h(\pi(s)) - \eta D_{\mathrm{KL}}(\pi(s); \pi_k(s)) \right], \tag{148}$$

by performing a few gradient steps over the above objective. Unlike ECPO, MDPO skips the closed form solution and optimizes directly the entropy and D_{KL} regularized objective by gradient ascent. We will call this baseline `AC-DirectOpt (MDPO)`.

Experimental comparisons with Exact PMD. As can be seen from the main experimental results in Fig. 2 (and also the further results in App. 9), AC-DirectOpt and AC-MProj being approximations of Exact PMD, end-up having worse performance on several tasks. Comparing the AC methods between themselves, AC-DirectOpt and AC-MProj have close performance on the aggregate metrics, while AC-NoKL lags behind the other two and shows again the benefits of D_{KL} regularization on policy update. Compared to StaQ, the performance of the D_{KL} regularized AC methods (i.e. AC-DirectOpt and AC-MProj) is between that of StaQ with $M = 1$ and $M = 50$, while AC-NoKL performs worse than $M = 1$. The improved performance of StaQ with higher M comes at little performance cost as can be seen in the table in Fig 1, where the runtime remains close to that of $M = 1$, whereas all three AC methods have to perform double the amount of gradient steps to update the actor in addition to the critic, resulting in a 35 to 45% longer run time than StaQ with $M = 1$.

11 Hyperparameters and Implementation Details

Policy Evaluation. In all our experiments, we use an ensemble of two neural networks, similarly to e.g. SAC (Haarnoja et al., 2018), to evaluate a Q-function and therefore two stacked neural networks for ξ-logits. In particular, we optimize the current Q-function weights θ to minimize the loss $\mathcal{L}(\theta)$,

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta\left(s,a\right) - \hat{Q}\left(s,a\right)\right)^2\right] \tag{149}$$

$$\hat{Q}(s,a) := R(s,a) + \gamma\mathbb{E}_{s'\sim\mathcal{D},a\sim\pi(s')}\left[\frac{1}{2}\sum_{i=1}^{2} Q_{\hat{\theta}_i}(s',a') - \tau h(\pi(s'))\right] \tag{150}$$

computing the mean over the target Q-functions with weights $\hat{\theta}_1, \hat{\theta}_2$.

Hyperparameters. Below, we provide the full list of hyperparameters used in our experiments[3]. StaQ's hyperparameters are listed in Table 2, while the hyperparameters for our baselines are provided in Tables 3-5. For TRPO and PPO, we use the implementation provided in `stable-baselines4` (Raffin et al., 2021), while we used our in-house PyTorch implementation of (M)-DQN[4].

To account for the different scales of the reward between environments, we apply a different reward scaling to the Classic environments and MinAtar. Note that this is equivalent to inverse-scaling the entropy weight τ and KL weight η, ensuring that ξ_k is of the same order of magnitude for all environments. To account for the varying action dimension $|A|$ of the environments, we set the *scaled entropy coefficient* $\bar{\tau}$ as a hyperparameter, defined by $\bar{\tau} = \tau \log|A|$, rather than directly setting τ. Furthermore, the entropy weight is linearly annealed from its minimum and maximum values.

[3] Code will be released to an open-source repository upon publication.
[4] We will add the link to the in-house library upon publication.

Hyperparameter	Classic	MinAtar
Discount (γ)	0.99	0.99
Memory size (M)	300	300
Policy update interval	5000	5000
Target type	hard	hard
Target update interval	200	200
Epsilon	0.05	0.05
Reward scale	**10**	**100**
KL weight (η)	20	20
Initial scaled ent. weight	2.0	2.0
Final scaled ent. weight	0.4	0.4
Ent. weight decay steps	**500K**	**1M**
Architecture	$\mathbf{256 \times 2}$	**Conv(16, 3, 3) + 128 MLP**
Activation function	ReLU	ReLU
Learning rate	0.0001	0.0001
Optimizer	Adam	Adam
Replay capacity	50K	50K
Batch size	256	256

Table 2: StaQ hyperparameters, with parameters which vary across environment types in bold.

Hyperparameter	Classic	MinAtar
Discount factor (γ)	0.99	0.99
Horizon	2048	1024
Num. epochs	10	3
Learning starts	5000	20000
GAE parameter	0.95	0.95
VF coefficient	0.5	1
Entropy coefficient	0	0.01
Clipping parameter	0.2	$0.1 \times \alpha$
Optimizer	Adam	Adam
Architecture	256×2	Conv(16, 3, 3) + 128 MLP
Activation function	Tanh	Tanh
Learning rate	3×10^{-4}	$2.5 \times 10^{-4} \times \alpha$
Batch size	64	256

Table 3: PPO hyperparameters, based on (Schulman et al., 2017). In the MinAtar environments α is linearly annealed from 1 to 0 over the course of learning.

Hyperparameter	Classic	MinAtar
Discount factor (γ)	0.99	0.99
Horizon	2048	2048
Learning starts	5000	20000
GAE parameter	0.95	0.95
Stepsize	0.01	0.01
Optimizer	Adam	Adam
Architecture	256×2	Conv(16, 3, 3) + 128 MLP
Activation function	Tanh	Tanh
Learning rate	3×10^{-4}	2.5×10^{-4}
Batch size	64	256

Table 4: TRPO hyperparameters, based on (Schulman et al., 2015).

Hyperparameter	Classic	MinAtar
Discount factor (γ)	0.99	0.99
Target update interval	100	8000
Update-to-data ratio	1	1
Epsilon	0.1	0.1
Decay steps	20K	20K
M-DQN temperature	0.03	0.03
M-DQN scaling term	1.0	0.9
M-DQN clipping value	-1	-1
Architecture	512×2	Conv(16, 3, 3) + 128 MLP
Activation function	ReLU	ReLU
Learning rate	1×10^{-3}	2.5×10^{-4}
Optimizer	Adam	Adam
Replay capacity	50K	1M
Batch size	128	32

Table 5: MDQN and DQN hyperparameters, based on (Vieillard et al., 2020b; Ceron & Castro, 2021)

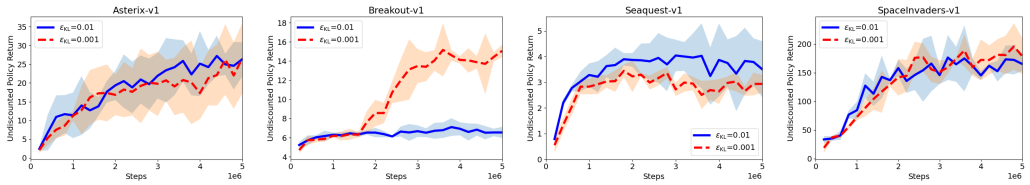


Figure 8: Mean and standard deviation of policy performance of "NatGrad + LS (TRPO)" baseline for two values of ϵ_{KL} averaged over 3 seeds.

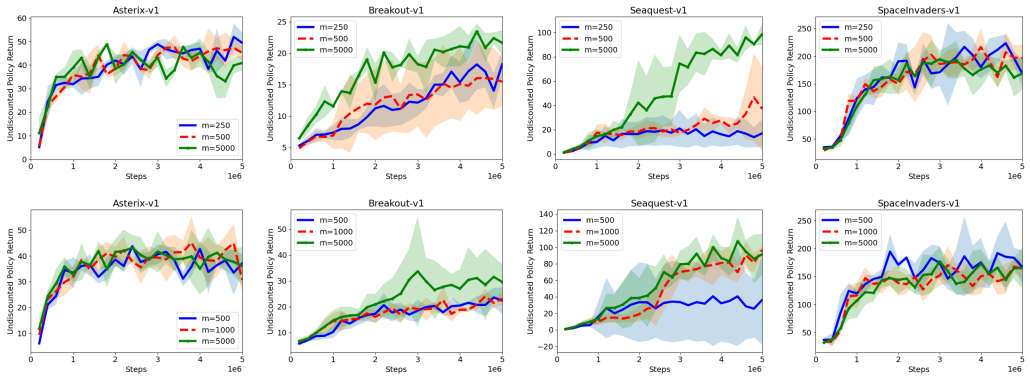


Figure 9: Sensitivity of AC-DirectOpt (MDPO) (**top**) and AC-MProj (ECPO) (**bottom**) to the number of gradient steps performed to update the actors. Plots show mean and std. deviation over 3 seeds. See App. 10.2 for details.

11.1 Hyperparameter ablations for Natural Gradient and Actor-Critic baselines

For a fair comparison, the NatGrad + LS and Actor-Critic baselines use the same hyperparameters as StaQ, except for the algorithm-specific parameters ϵ_{KL} and m, for which we provide sensitivity studies here.

NatGrad + LS. The baseline "NatGrad + LS" introduces an additional hyperparameter ϵ_{KL} compared to the standard PMD setting that StaQ follows. To properly set this hyperparameter we perform a preliminary study using 3 seeds on a subset of the MinAtar tasks. The results in Figure 8 show that on a few tasks, the value of the ϵ_{KL} does not impact greatly the performance, but when there are significant differences between the two algorithms, $\epsilon_{\text{KL}} = 0.001$ is often better than the value $\epsilon_{\text{KL}} = 0.01$ used in may of TRPO's implementations, e.g. in `stable-baselines3` (Raffin et al., 2021). As such, we use $\epsilon_{\text{KL}} = 0.001$ for all experiments involving "NatGrad + LS".

MDPO, ECPO and SAC. An important parameter for MDPO and ECPO is m, the number of gradient steps that are used to update the actor. We perform a 3 seed hyperparameter sensitivity analysis w.r.t. m on 4 MinAtar tasks. Results show in Fig. 9 that on some tasks, this parameter is actually not very crucial, but on some other tasks, using more gradient steps for the actor seems to improve performance. As such, for the remained of this section, we will use $m = 5000$ for both AC-DirectOpt (MDPO) and AC-MProj (ECPO), which corresponds to performing one actor gradient update per time-step in the environment. This is the same setting as in SAC, and we will use the same number of actor updates for the AC-NoKL (SAC) baseline.

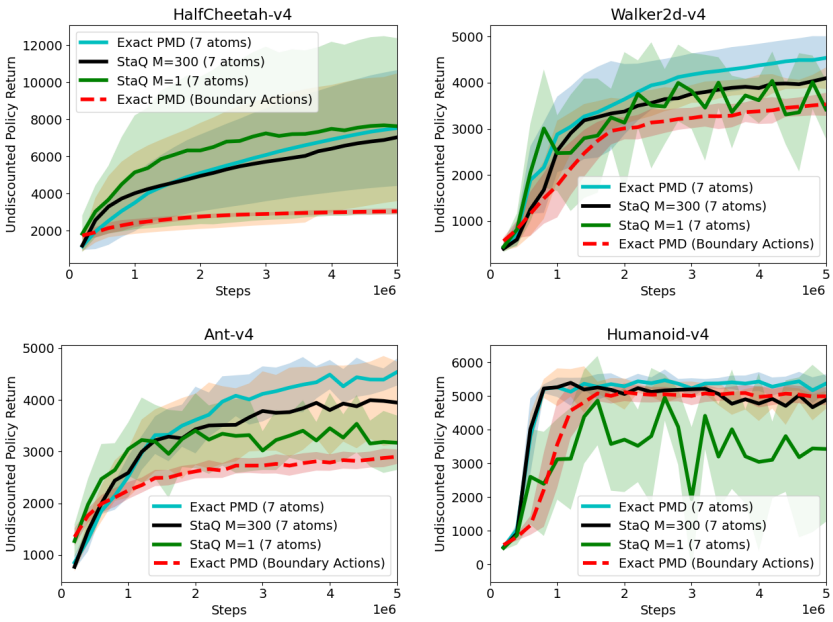


Figure 10: StaQ on MuJoCo with action space discretized by just considering the boundary actions, versus using the independence assumption from Tang & Agrawal (2020) with 7 atoms. Results averaged over 5 seeds.

12 Extension to continuous action spaces

While StaQ is inherently a discrete-actions algorithm, in this appendix we describe some promising ways in which we could extend it to continuous action spaces. One interesting research questions in this direction is to investigate the improvements a Boltzmann distribution could bring to exploration compared to the unimodal Gaussian distribution that is commonly used in continuous action deep RL algorithms.

The simplest approach is to discretise the continuous action space by only considering the "boundary actions", similarly to (Seyde et al., 2021). To illustrate, the discrete version of a Mujoco task with action space $A = [-1, 1]^d$ consists in several $2d$ dimensional vectors that have zeroes everywhere except at entry $i \in \{1, \dots, d\}$ that can either take a value of 1 or -1; to that we add the zero action, for a total of $2d + 1$ actions.

Another direction to tackle continuous action spaces is to leverage an independence assumption in the policy following Tang & Agrawal (2020). Concretely, Tang & Agrawal discretize each action dimension into a finite number of atoms and assume that each action dimension follows an independent distribution—which is also assumed by continuous action RL algorithms using diagonal Gaussian policies. In our setting, this translates to the logit space, which is also the space of Q-functions, by assuming separability of the Q-function approximation i.e. that q_k, the approximation of the Q-function Q_k, is given by

$$q_k(s, a) = \sum_{i=1}^{\dim(\mathrm{A})} q_k^{[i]}(s, a^{[i]}), \tag{151}$$

where $a^{[i]}$ is entry i of the action vector a and $\{q_k^{[i]}\}_{i=1}^{\dim(A)}$ is a set of real functions. This independence assumption makes it possible to consider fine grained discretizations with extremely large but finite action spaces, from which one can sample exactly and efficiently by sampling each action dimension independently. Incidentally, this separability assumption could allow a more efficient implementation of the Gumbel-max trick discussed above.

To illustrate how this could benefit StaQ, we have implemented the discretization from Tang & Agrawal (2020) in combination with StaQ using $K = 7$ atoms by action dimension. Concretely, the Q-network outputs K values for each action dimension, and the Q-function for a given action is computed using Eq. 151. We have run 5 seeds of StaQ using this Q-network architecture on the 4 MuJoCo environments shown in Fig. 10, comparing it to the simple Boundary Actions discretisation described previously. All hyperparameters of StaQ are the same except for the target update rate of the Q-target (1000 for HalfCheetah and Walker, 2500 for Ant and Humanoid). From the plot, we can see that the Boundary Actions discretisation works rather well for most environments, and the finer-grained discretization brings further improvements on all but the Humanoid-v4 task. Moreover, using StaQ improves over previous results reported by Tang & Agrawal (2020), that used PPO and TRPO as base learners on most MuJoCo tasks, showing the impact our work could have in this line of research.

13 Pseudocode of StaQ

We provide in this section the pseudocode of StaQ in Alg. 1. As an approximate policy iteration algorithm, StaQ comprises three main steps: i) data collection, ii) policy evaluation iii) policy improvement. Data collection (Line 4-5) consist in interacting with the environment to collect transitions of type (state, action, reward, next state) that are stored in a replay buffer. A policy evaluation algorithm (Eq. 149) is then called to evaluate the current Q-function Q_τ^k using the replay buffer. Finally, the policy update is optimization-free and simply consists in stacking the Q-function as discussed in Sec. 5.1. After K iterations, the last policy is returned.

Algorithm 1 StaQ (Finite-memory entropy regularized policy mirror descent)

1: **Input:** An MDP $\mathcal{M}$, a memory-size M, Number of samples per iteration N, Replay buffer size D, Initial behavior policy π_0^b, entropy weight τ, D_{KL} weight η, ϵ-softmax exploration parameter

2: **Output:** Policy $\pi_K \propto \exp(\xi_K)$
3: **for** $k = 0$ **to** $K - 1$ **do**
4: Interact with $\mathcal{M}$ using the behavior policy π_k^b for N times steps
5: Update replay buffer $\mathcal{D}_k$ to contain the last D transitions
6: Learn Q_τ^k from $\mathcal{D}_k$ using a policy evaluation algorithm (Eq. 149)
7: Obtain logits ξ_{k+1} by stacking the last M Q-functions (see Sec. 5.1) following the finite-memory EPMD update of Eq. 6.
8: Set $\pi_{k+1} \propto \exp(\xi_{k+1})$ and π_{k+1}^b to an ϵ-softmax policy over π_{k+1}
9: **end for**