Improved Regret Bound and Experience Replay in Regularized Policy Iteration

Nevena Lazić [1] **Dong Yin** [1] **Yasin Abbasi-Yadkori** [1] **Csaba Szepesvári** [1 2]

Abstract

In this work, we study algorithms for learning in infinite-horizon undiscounted Markov decision processes (MDPs) with function approximation. We first show that the regret analysis of the POLITEX algorithm (a version of regularized policy iteration) can be sharpened from $O(T^{3/4})$ to $O(\sqrt{T})$ under nearly identical assumptions, and instantiate the bound with linear function approximation. Our result provides the first high-probability $O(\sqrt{T})$ regret bound for a computationally efficient algorithm in this setting. The exact implementation of POLITEX with neural network function approximation is inefficient in terms of memory and computation. Since our analysis suggests that we need to approximate the average of the action-value functions of past policies well, we propose a simple efficient implementation where we train a single Q-function on a replay buffer with past data. We show that this often leads to superior performance over other implementation choices, especially in terms of wall-clock time. Our work also provides a novel theoretical justification for using experience replay within policy iteration algorithms.

1. Introduction

Model-free reinforcement learning (RL) algorithms combined with powerful function approximation have achieved impressive performance in a variety of application domains over the last decade. Unfortunately, the theoretical understanding of such methods is still quite limited. In this work, we study single-trajectory learning in infinite-horizon undiscounted Markov decision processes (MDPs), also known as average-reward MDPs, which capture tasks such as routing and the control of physical systems.

One line of works with performance guarantees for the average-reward setting follows the "online MDP" approach proposed by Even-Dar et al. (2009), where the agent selects policies by running an online learning algorithm in each state, typically mirror descent. The resulting algorithm is a version of approximate policy iteration (API), which alternates between (1) estimating the state-action value function (or Q-function) of the current policy and (2) setting the next policy to be optimal w.r.t. the sum of all previous Q-functions plus a regularizer. Note that, by contrast, standard API sets the next policy only based on the most recent Q-function. The policy update can also be written as maximizing the most recent Q-function minus KL-divergence to the previous policy, which is somewhat similar to recently popular versions of API (Schulman et al., 2015; 2017; Achiam et al., 2017; Abdolmaleki et al., 2018; Song et al., 2019a).

The original work of Even-Dar et al. (2009) studied this scheme with known dynamics, tabular representation, and adversarial reward functions. More recent works (Abbasi-Yadkori et al., 2019; Hao et al., 2020; Wei et al., 2020a) have adapted the approach to the case of unknown dynamics, stochastic rewards, and value function approximation. With linear value functions, the POLITEX algorithm of Abbasi-Yadkori et al. (2019) achieves $O(T^{3/4})$ high-probability regret in ergodic MDPs, and the results only scale in the number of features rather than states. Wei et al. (2020a) later show an $O(\sqrt{T})$ bound on *expected* regret for a similar algorithm named MDP-EXP2. In this work, we revisit the analysis of POLITEX and show that it can be sharpened to $O(\sqrt{T})$ under nearly identical assumptions, resulting in the first $O(\sqrt{T})$ *high-probability* regret bound for a computationally efficient algorithm in this setting.

In addition to improved analysis, our work also addresses practical implementation of POLITEX with neural networks. The policies produced by POLITEX in each iteration require access to the sum of all previous Q-function estimates. With neural network function approximation, exact implementation requires us to keep all past networks in memory and evaluate them at each step, which is inefficient in terms of memory and computation. Some practical implementation choices include subsampling Q-functions and/or optimizing a KL-divergence regularized objective w.r.t. a parametric policy at each iteration. We propose an alternative approach, where we approximate the average of all past Q-functions by training a single network on a replay buffer with past

[1]DeepMind [2]University of Alberta. Correspondence to: Nevena Lazic <nevena@google.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

data. We demonstrate that this choice often outperforms other approximate implementations, especially in terms of run-time. When available memory is constrained, we propose to subsample transitions using the notion of coresets (Bachem et al., 2017).

Our work also provides a novel perspective on the benefits of experience replay. Experience replay is a standard tool for stabilizing learning in modern deep RL, and typically used in *off-policy* methods like Deep Q-Networks (Mnih et al., 2013), as well as "value gradient" methods such as DDPG (Lillicrap et al., 2016) and SVG (Heess et al., 2015). A different line of *on-policy* methods typically does not rely on experience replay; instead, learning is stabilized by constraining consecutive policies to be close in terms of KL divergence (Schulman et al., 2015; Song et al., 2019a; Degrave et al., 2019). We observe that both experience replay and KL-divergence regularization can be viewed as approximate implementations of POLITEX. Thus, we provide a theoretical justification for using experience replay in API, as an approximate implementation of online learning in each state. Note that this online-learning view differs from the commonly used justifications for experience replay, namely that it "breaks temporal correlations" (Schaul et al., 2016; Mnih et al., 2013). Our analysis also suggests a new objective for subsampling or priority-sampling transitions in the replay buffer, which differs priority-sampling objectives of previous work (Schaul et al., 2016).

In summary, our main contributions are (1) an improved analysis of POLITEX, showing an $O(\sqrt{T})$ regret bound under the same assumptions as the original work, and (2) an efficient implementation that also offers a new perspective on the benefits of experience replay.

2. Setting and Notation

We consider learning in infinite-horizon undiscounted ergodic MDPs $(\mathcal{X}, \mathcal{A}, r, P)$, where $\mathcal{X}$ is the state space, $\mathcal{A}$ is a finite action space, $r : \mathcal{X} \times \mathcal{A} \to [0, 1]$ is an unknown reward function, and $P : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}}$ is the unknown probability transition function. A policy $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$ is a mapping from a state to a distribution over actions. Let $\{(x_t^\pi, a_t^\pi)\}_{t=1}^\infty$ denote the state-action sequence obtained by following policy π. The expected average reward of policy π is defined as

$$J_\pi := \lim_{T\to\infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=1}^{T} r(x_t^\pi, a_t^\pi)\right]. \tag{2.1}$$

Let μ_π denote the stationary state distribution of a policy π, satisfying $\mu_\pi = \mathbb{E}_{x\sim\mu_\pi, a\sim\pi}[P(\cdot|x, a)]$. We will sometimes write μ_π as a vector, and use $\nu_\pi = \mu_\pi \otimes \pi$ to denote the stationary state-action distribution. In ergodic MDPs, J_π and μ_π are well-defined and independent of the initial state. The optimal policy π_* is a policy that maximizes the expected average reward. We will denote by J_* and μ_* the expected average reward and stationary state distribution of π_*, respectively.

The value function of a policy π is defined as:

$$V_\pi(x) := \mathbb{E}\left[\sum_{t=1}^{\infty}(r(x_t^\pi, a_t^\pi) - J_\pi)|x_1^\pi = x\right]. \tag{2.2}$$

The state-action value function $Q_\pi(x, a)$ is defined as

$$Q_\pi(x, a) := r(x, a) - J_\pi + \sum_{x'} P(x'|x, a)V_\pi(x'). \tag{2.3}$$

Notice that

$$V_\pi(x) = \sum_a \pi(a|x)Q_\pi(x, a). \tag{2.4}$$

Equations (2.3) and (2.4) are known as the Bellman equation. If we do not use the definition of $V_\pi(x)$ in Eq. (2.2) and instead solve for $V_\pi(x)$ and $Q_\pi(x, a)$ using the Bellman equation, we can see that the solutions are unique up to an additive constant. Therefore, in the following, we may use $V_\pi(x)$ and $Q_\pi(x, a)$ to denote the same function up to a constant. We will use the shorthand notation $Q_\pi(x, \pi') = \mathbb{E}_{a\sim\pi'(\cdot|x)}[Q_\pi(x, a)]$; note that $Q_\pi(x, \pi) = V_\pi(x)$. The advantage function of a policy π is defined as $A_\pi(x, a) = Q_\pi(x, a) - V_\pi(x)$.

The agent interacts with the environment as follows: at each round t, the agent observes a state $x_t \in \mathcal{X}$, chooses an action $a_t \sim \pi_t(\cdot|x_t)$, and receives a reward $r_t := r(x_t, a_t)$. The environment then transitions to the next state x_{t+1} with probability $P(x_{t+1}|x_t, a_t)$. Recall that π_* is the optimal policy and J_* is its expected average reward. The regret of an algorithm with respect to this fixed policy is defined as

$$R_T := \sum_{t=1}^{T}\Big(J_* - r(x_t, a_t)\Big). \tag{2.5}$$

The learning goal is to find an algorithm that minimizes the long-term regret R_T.

Our analysis will require the following assumption on the mixing rate of policies.

Assumption 2.1 (Uniform mixing)**.** Let H_π be the state-action transition matrix of a policy π. Let $\gamma(H_\pi)$ be the corresponding *ergodicity coefficient* (Seneta, 1979), defined as $\gamma(H_\pi) := \max_{z:z^\top\mathbf{1}=0, \|z\|_1=1} \|z^\top H_\pi\|_1$. We assume that there exists a scalar $\gamma < 1$ such that for any policy π, $\gamma(H_\pi) \le \gamma < 1$.

Assumption 2.1 implies that for any pair of distributions ν, ν', $\|(\nu - \nu')^\top H_\pi\|_1 \le \gamma\|\nu - \nu'\|_1$; see Lemma A.1 in Appendix A for a proof.

3. Related Work

Regret bounds for average-reward MDPs. Most no-regret algorithms for infinite-horizon undiscounted MDPs are only applicable to tabular representations and model-based (Bartlett, 2009; Jaksch et al., 2010; Ouyang et al., 2017; Fruit et al., 2018; Jian et al., 2019; Talebi & Maillard, 2018). For weakly-communicating MDPs with diameter D, these algorithms nearly achieve the minimax lower bound $\Omega(\sqrt{D|\mathcal{X}||\mathcal{A}|T})$ (Jaksch et al., 2010) with high probability. Wei et al. (2020b) provide model-free algorithms with regret bounds in the tabular setting. In the model-free setting with function approximation, the POLITEX algorithm (Abbasi-Yadkori et al., 2019) achieve $O(d^{1/2}T^{3/4})$ regret in uniformly ergodic MDPs, where d is the size of the compressed state-action space. Hao et al. (2020) improve these results to $O(T^{2/3})$. More recently, Wei et al. (2020a) present three algorithms for average-reward MDPs with linear function approximation. Among these, FOPO achieves $O(\sqrt{T})$ regret but is computationally inefficient, and OLSVI.FH is efficient but obtains $O(T^{3/4})$ regret. The MDP-EXP2 algorithm is computationally efficient, and under similar assumptions as in Abbasi-Yadkori et al. (2019) it obtains a $O(\sqrt{T})$ bound on *expected regret* (a weaker guarantee than the high-probability bounds in other works). Our analysis shows a high-probability $O(\sqrt{T})$ regret bound under the same assumptions.

KL-regularized approximate policy iteration. Our work is also related to approximate policy iteration algorithms which constrain each policy to be close to the previous policy in the sense of KL divergence. This approach was popularized by TRPO (Schulman et al., 2015), where it was motivated as an approximate implementation of conservative policy iteration (Kakade & Langford, 2002). Some of the related subsequent works include PPO (Schulman et al., 2017), MPO (Abdolmaleki et al., 2018), V-MPO (Song et al., 2019a), and CPO (Achiam et al., 2017). While these algorithms place a constraint on consecutive policies and are mostly heuristic, another line of research shows that using KL divergence as a regularizer has a theoretical justification in terms of either regret guarantees (Abbasi-Yadkori et al., 2019; Hao et al., 2020; Wei et al., 2020b;a) or error propagation (Vieillard et al., 2020a;b).

Experience replay. Experience replay (Lin, 1992) is one of the central tools for achieving good performance in deep reinforcement learning. While it is mostly used in off-policy methods such as deep Q-learning (Mnih et al., 2013; 2015), it has also shown benefits in value-gradient methods (Heess et al., 2015; Lillicrap et al., 2016), and has been used in some variants of KL-regularized policy iteration (Abdolmaleki et al., 2018; Tomar et al., 2020). Its success has been attributed to removing some temporal correlations from data fed to standard gradient-based optimization algorithms. Schaul et al. (2016) have shown that non-uniform replay sampling based on the Bellman error can improve performance. Unlike these works, we motivate experience replay from the perspective of online learning in MDPs (Even-Dar et al., 2009) with the goal of approximating the average of past value functions well.

Continual learning. Continual learning (CL) is the paradigm of learning a classifier or regressor that performs well on a set of tasks, where each task corresponds to a different data distribution. The tasks are observed sequentially, and the learning goal is to avoid forgetting past tasks without storing all the data in memory. This is quite similar to our goal of approximating the average of sequentially-observed Q-functions, where the data for approximating each Q-function has different distribution. In general, approaches to CL can be categorized as regularization-based (Kirkpatrick et al., 2017; Zenke et al., 2017; Farajtabar et al., 2020; Yin et al., 2020), expansion-based (Rusu et al., 2016), and replay-based (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2018; Borsos et al., 2020), with the approaches based on experience replay typically having superior performance over other methods.

4. Algorithm

Our algorithm is similar to the POLITEX schema (Abbasi-Yadkori et al., 2019). In each phase k, POLITEX obtains an estimate $\widehat{Q}_{\pi_k}$ of the action-value function Q_{π_k} of the current policy π_k, and then sets the next policy using the mirror descent update rule:

$$\begin{aligned}\pi_{k+1}(\cdot|x) &= \operatorname*{argmax}_{\pi} \widehat{Q}_{\pi_k}(x,\pi) - \eta^{-1} D_{KL}(\pi \,\|\, \pi_k(\cdot|x)) \\ &= \operatorname*{argmax}_{\pi} \sum_{i=1}^{k} \widehat{Q}_{\pi_i}(x,\pi) + \eta^{-1}\mathcal{H}(\pi) \\ &\propto \exp\left(\eta \sum_{i=1}^{k} \widehat{Q}_{\pi_i}(x,\cdot)\right), \end{aligned} \tag{4.1}$$

where $\mathcal{H}(\cdot)$ is the entropy function. When the functions $\{\widehat{Q}_{\pi_i}\}_{i=1}^{k}$ are tabular or linear, the above update can be implemented efficiently by simply summing all the table entries or weight vectors. However, with neural network function approximation, we need to keep all networks in memory and evaluate them at each step, which quickly becomes inefficient in terms of storage and computation. Some of the efficient implementations proposed in literature include keeping a subset of the action-value functions (Abbasi-Yadkori et al., 2019), and using a parametric policy and optimizing the KL-regularized objective w.r.t. the parameters over the available data (Tomar et al., 2020).

Our proposed method, presented in Algorithm 1, attempts to directly approximate the average of all previous action-value

Algorithm 1 Schema for policy iteration with replay

Input: phase length τ, num. phases K, parameter η
Initialize: $\pi_1(a|x) = 1/|\mathcal{A}|$, empty replay buffer $\mathcal{R}$
for $k = 1, \dots, K$ **do**
  Execute π_k for τ time steps and collect data $\mathcal{D}_k$
  Compute $\widehat{Q}_k$, an estimate of $Q_k = \frac{1}{k}\sum_{i=1}^{k} Q_{\pi_i}$, from data $\mathcal{D}_k$ and replay $\mathcal{R}$
  Set $\pi_{k+1}(a|x) \propto \exp\left(\eta k \widehat{Q}_k(x,a)\right)$
  Update replay $\mathcal{R}$ with $\mathcal{D}_k$
end for
Output: π_{K+1}

functions

$$Q_k(x,a) := \frac{1}{k}\sum_{i=1}^{k} Q_{\pi_i}(x,a).$$

To do so, we only use a single network, and continually train it to approximate Q_k. At each iteration k, we obtain a dataset of tuples $\mathcal{D}_k = \{(x_t, a_t, R_t)\}$, where R_t is the empirical return from the state-action pair (x_t, a_t). We initialize $\widehat{Q}_k$ to $\widehat{Q}_{k-1}$ and update it by minimizing the squared error over the union of $\mathcal{D}_k$ and the replay buffer $\mathcal{R}$. We then update the replay buffer with all or a subset of data in $\mathcal{D}_k$.

In the sequel, in Section 5, we first show that by focusing on estimating the average Q-function, we can improve the regret bound of POLITEX under nearly identical assumption; in Section 6, we instantiate this bound for linear value functions, where we can estimate the average Q-function simply by weight averaging; in Section 7, we focus on practical implementations, in particular, we discuss the limitations of weight averaging, and provide details on how to leverage replay data when using non-linear function approximation; in Section 8 we present our experimental results; and in Section 9 we make final remarks.

5. Regret Analysis of POLITEX

In this section, we revisit the regret analysis of POLITEX (Abbasi-Yadkori et al., 2019) in ergodic average-reward MDPs, and show that it can be improved from $O(T^{3/4})$ to $O(\sqrt{T})$ under similar assumptions. Our analysis relies in part on a simple modification of the regret decomposition. Namely, instead of including the estimation error of each value-function Q_{π_k} in the regret, we consider the error in the running average Q_k. When this error scales as $O(1/\sqrt{k})$ in a particular weighted norm, the regret of POLITEX is $O(\sqrt{T})$. As we show in the following section, this bound can be instantiated for linear value functions under the same assumptions as Abbasi-Yadkori et al. (2019).

Assumption 5.1 (Boundedness)**.** Let $\widehat{Q}_{\pi_k} := k\widehat{Q}_k - (k-1)\widehat{Q}_{k-1}$. We assume that there exists a constant $Q_{\max}$ such that for all $k = 1, ..., K$ and for all $x \in \mathcal{X}$,

$$\max_a \widehat{Q}_{\pi_k}(x,a) - \min_a \widehat{Q}_{\pi_k}(x,a) \le Q_{\max}.$$

For the purpose of our algorithm, functions $\widehat{Q}_{\pi_k}$ are unique up to a constant. Thus we can equivalently assume that $\|\widehat{Q}_{\pi_k}(x,\cdot)\|_\infty \le Q_{\max}$ for all x.

Define $\widehat{V}_{\pi_k}(x) := \widehat{Q}_{\pi_k}(x, \pi_k)$. Let $V_k := \frac{1}{k}\sum_{i=1}^{k} V_{\pi_i}(x)$ and $\widehat{V}_k(x) := \sum_{i=1}^{k} \widehat{V}_{\pi_i}(x)$ be the average of the state-value functions and its estimate. We will require the estimation error of the running average to scale as in the following assumption.

Assumption 5.2 (Estimation error)**.** Let μ_* be the stationary state distribution of the optimal policy π_*. With probability at least $1-\delta$, for a problem-dependent constant C, the errors in $\widehat{Q}_K$ and $\widehat{V}_K$ are bounded as

$$\mathbb{E}_{x\sim\mu_*}[\widehat{V}_K(x) - V_K(x)] \le C\sqrt{\log(1/\delta)/K}$$
$$\mathbb{E}_{x\sim\mu_*}[Q_K(x,\pi_*) - \widehat{Q}_K(x,\pi_*)] \le C\sqrt{\log(1/\delta)/K}\,.$$

Define $S_\delta(|\mathcal{A}|, \mu_*)$ as in Abbasi-Yadkori et al. (2019):

$$S_\delta(|\mathcal{A}|, \mu_*) := \sqrt{\frac{\log|\mathcal{A}|}{2}} + \langle \mu_*, \sqrt{\frac{1}{2}\log\frac{1}{\delta\mu_*}}\rangle.$$

We bound the regret of POLITEX in the following theorem. Here, recall that τ is the length of each phase, and γ and η are defined in Assumption 2.1 and Eq. (4.1), respectively.

Theorem 5.3 (Regret of POLITEX)**.** Let Assumptions 2.1, 5.2, and 5.1 hold. For $\tau \ge \frac{\log T}{2\log(1/\gamma)}$ and $\eta = \frac{\sqrt{8\log|\mathcal{A}|}}{Q_{\max}\sqrt{K}}$, for a constant C_1, with probability at least $1-4\delta$, the regret of POLITEX in ergodic average-reward MDPs is bounded as

$$R_T \le \frac{C_1(1+Q_{\max})S_\delta(|\mathcal{A}|,\mu_*)\sqrt{\tau}}{(1-\gamma)^2}\sqrt{T}\,.$$

Proof. We start by decomposing the cumulative regret, following similar steps as Abbasi-Yadkori et al. (2019):

$$R_T = \sum_{k=1}^{K}\sum_{t=(k-1)\tau+1}^{k\tau} (J_* - J_{\pi_k}) + (J_{\pi_k} - r_t). \quad (5.1)$$

The second term $V_T = \sum_{k=1}^{K}\sum_{t=(k-1)\tau+1}^{k\tau}(J_{\pi_k} - r_t)$ captures the sum of differences between observed rewards and their long term averages. In previous work, this term was shown to scale as $O(K\sqrt{\tau})$. We show that the analysis can in fact be tightened to $O(\sqrt{T})$ using improved concentration bounds and the slow-changing nature of the policies. See Lemma B.1 in Appendix B for precise details.

The first term, which is also called *pseudo-regret* in literature, measures the difference between the expected reward

of the reference policy and the policies produced by the algorithm. Applying the performance difference lemma (Cao, 1999), we can write each pseudo-regret term as

$$J_* - J_{\pi_k} = \mathbb{E}_{x\sim\mu_*}\left[Q_{\pi_k}(x,\pi_*) - Q_{\pi_k}(x,\pi_k)\right].$$

Now, notice that POLITEX is running exact mirror descent for loss functions $\widehat{Q}_{\pi_k}$. We bridge the pseudo-regret by the $\widehat{Q}_{\pi_k}$ terms:

$$R_{T1a} = \tau\sum_{k=1}^{K}\mathbb{E}_{x\sim\mu_*}\left[Q_{\pi_k}(x,\pi_*) - \widehat{Q}_{\pi_k}(x,\pi_*)\right] \quad (5.2)$$

$$R_{T1b} = \tau\sum_{k=1}^{K}\mathbb{E}_{x\sim\mu_*}\left[\widehat{Q}_{\pi_k}(x,\pi_k) - Q_{\pi_k}(x,\pi_k)\right] \quad (5.3)$$

$$R_{T2} = \tau\sum_{k=1}^{K}\mathbb{E}_{x\sim\mu_*}\left[\widehat{Q}_{\pi_k}(x,\pi_*) - \widehat{Q}_{\pi_k}(x,\pi_k)\right]. \quad (5.4)$$

R_{T2} can be bounded using the regret of mirror descent as in previous work (Abbasi-Yadkori et al., 2019). Setting $\eta = \frac{\sqrt{\log|\mathcal{A}|}}{Q_{\max}\tau\sqrt{2K}}$, and using a union bound over all states, with probability at least $1-\delta$, R_{T2} is bounded as

$$R_{T2} \le \tau Q_{\max} S_\delta(|\mathcal{A}|,\mu_*)\sqrt{K}\,.$$

Bounding regret due to estimation error. We now focus on bounding $R_{T1} = R_{T1a} + R_{T1b}$ under Assumption 5.2. We have the following:

$$\begin{aligned}
R_{T1a} &= \tau\sum_{k=1}^{K}\mathbb{E}_{x\sim\mu_*}\left[Q_{\pi_k}(x,\pi_*) - \widehat{Q}_{\pi_k}(x,\pi_*)\right]\\
&= (\tau K)\mathbb{E}_{x\sim\mu_*,a\sim\pi_*}\left[Q_K(x,a) - \widehat{Q}_K(x,a)\right]\\
&\le C\tau\sqrt{K\log(1/\delta)}.\\
R_{T1b} &= \tau\sum_{k=1}^{K}\mathbb{E}_{x\sim\mu_*}\left[\widehat{V}_{\pi_k}(x) - V_{\pi_k}(x)\right]\\
&= \tau K\mathbb{E}_{x\sim\mu_*}\left[\widehat{V}_K(x) - V_K(x)\right]\\
&\le C\tau\sqrt{K\log(1/\delta)}.
\end{aligned}$$

We can then obtain the final result by combining the bounds on $R_{T1a}, R_{T1b}, R_{T2}, V_T$ from Appendix B, and using union bound as well as the fact that $K = T/\tau$. □

6. Linear Value Functions

In this section, we show that the estimation error condition in Assumption 5.2 (and thus $O(\sqrt{T})$ regret) can be achieved under similar assumptions as in Abbasi-Yadkori et al. (2019) and Wei et al. (2020a), which we state next.

Assumption 6.1 (Linear value functions)**.** The action-value function Q_π of any policy π is linear: $Q_\pi(x,a) = w_\pi^\top\phi(x,a)$, where $\phi : \mathcal{S}\times\mathcal{A}\to\mathbb{R}^d$ is a known feature function such that $\max_{x,a}\|\phi(x,a)\| \le C_\Phi$.

Assumption 6.2 (Feature excitation)**.** There exists a constant σ^2 such that for any policy π, $\lambda_{\min}\left(\mathbb{E}_{(x,a)\sim\mu_\pi\otimes\pi}[\phi(x,a)\phi(x,a)^\top]\right) \ge \sigma^2 > 0$.

We now describe a simple procedure for estimating the average action-value functions $Q_k(x,a) = \phi(x,a)^\top w_k$, where $w_k = \frac{1}{k}\sum_{i=1}^{k} w_{\pi_i}$, such that the conditions of Assumption 5.2 are satisfied. Essentially, we estimate each Q_{π_i} using least-squares Monte Carlo and then average the weights. We will use the shorthand notation $\phi_t = \phi(x_t,a_t)$. Let $\mathcal{H}_i$ and $\mathcal{T}_i$ be subsets of time indices in phase i (defined later). We estimate Q_{π_i} as follows:

$$\widehat{w}_{\pi_i} = \left(\sum_{t\in\mathcal{H}_i}\phi_t\phi_t^\top + \alpha I\right)^{-1}\sum_{t\in\mathcal{H}_i}\phi_t R_t \quad (6.1)$$

where R_t are the empirical b-step returns (b is specified later), computed as

$$R_t = \sum_{j=t}^{t+b}(r_t - \widehat{J}_{\pi_i}), \quad \widehat{J}_{\pi_i} = \frac{1}{|\mathcal{T}_i|}\sum_{t\in\mathcal{T}_i} r_t. \quad (6.2)$$

We then estimate w_k as $\widehat{w}_k = \frac{1}{k}\sum_{i=1}^{k}\widehat{w}_{\pi_i}$. Note that for this special case of linear value functions, we do not need to use the replay buffer in Algorithm 1. For analysis purposes, we divide each phase of length τ into $2m$ blocks of size b and let $\mathcal{H}_i$ ($\mathcal{T}_i$) be the starting indices of odd (even) blocks in phase i. Due to the gaps between indices and fast mixing, this makes the data almost independent (we make this precise in Appendix C) and the error easier to analyze. In practice, one may simply want to use all data.

For a distribution μ, let $\|x\|_\mu$ denote the distribution-weighted norm such that $\|x\|_\mu^2 = \sum_i \mu_i x_i^2$. Using Jensen's inequality, we have that $(\mathbb{E}_{x\sim\mu}[q(x)])^2 \le \mathbb{E}_{x\sim\mu}[q(x)^2]$. Thus, it suffices to bound the Q-function error in the distribution-weighted norm, $\|Q_K - \widehat{Q}_K\|_{\mu_*\otimes\pi_*}$. Furthermore, given bounded features,

$$\|\widehat{Q}_K - Q_K\|_{\mu_*\otimes\pi_*} \le C_\Phi\|\widehat{w}_K - w_K\|_2\,,$$

so it suffices to bound the error in the weights. We bound this error in the following Lemma, proven in Appendix C.

Lemma 6.3 (Estimation error for linear functions)**.** Suppose that Assumptions 2.1, 6.1 and 6.2 hold and that true action-value weights are bounded as $\|w_{\pi_i}\|_2 \le C_w$ for all $i = 1,\dots K$. Then for any policy π, for $\alpha = \sqrt{\tau/K}$, $m \ge 72C_\Phi^4\sigma^{-2}(1-\gamma)^{-2}\log(d/\delta)$, and $b \ge \frac{\log(T\delta^{-1}(1-\gamma)^{-1})}{\log(1/\gamma)}$,

there exists an absolute constant c such that with probability at least $1-\delta$,

$$\|\widehat{w}_K - w_K\|_2 \le c\sigma^{-2}(C_w + C_\Phi) b \sqrt{\frac{\log(2d/\delta)}{Km}}.$$

Furthermore, in Appendix D, we show that the error in the average state-value function satisfies the following:

$$\mathbb{E}_{\mu_*}[\widehat{V}_K(x) - V_K(x)] \le cC_\Phi |\mathcal{A}| (C_w + C_\Phi) \frac{b}{\sigma^2} \sqrt{\frac{\log(2d/\delta)}{Km}}.$$

We have demonstrated that Assumption 5.2 can be satisfied with linear value functions. For Assumption 5.1, it suffices for the weight estimates $\{\widehat{w}_{\pi_i}\}$ to be bounded. This will be true, since we assume that the true weights are bounded and we can bound the error in the weight space. Thus POLITEX has an $O(\sqrt{T})$ regret in this setting (though note that we incur an extra $|\mathcal{A}|$ factor coming from the $\widehat{V}_K$ error bound).

7. Practical Implementation

As mentioned earlier, the key idea in our policy update is to obtain an estimate $\widehat{Q}_k$ of the average of all the Q-functions in previous phases. We have seen that when we use linear functions to approximate the Q-functions, we can simply average the weights in order to get an estimate of the average Q-function. However, in practice, we often need to use non-linear functions, especially neural networks, to approximate complex Q-functions. In this section, we discuss how to efficiently implement our algorithm with non-linear function approximation.

7.1. Weight Averaging

The simplest idea may be averaging the weights of neural networks. However, a crucial difference from the linear setting is that averaging the weights of the neural networks is not equivalent to averaging the functions that they represent: the function that a neural network represent is invariant to the permutation of the hidden units, and thus two networks with very different weights can represent similar functions. Therefore, this implementation may only succeed when all the Q-function approximations are around the same local region in the weight space. Thus, when we learn the new Q-function $\widehat{Q}_{\pi_k}$ in phase k, we should initialize it with $\widehat{Q}_{k-1}$, run SGD with new data, and then average with $\widehat{Q}_{k-1}$.

7.2. Experience Replay

Another natural idea is to leverage the data from replay buffer to obtain an estimate of the average Q-function. We elaborate the details below. We use simple b-step Monte Carlo estimate for the Q value of each state-action pair. For any $(i-1)\tau + 1 \le t \le i\tau - b$, we can estimate the state-action value of (x_t, a_t) by b-step cumulative reward[1]

$$R_t = \sum_{j=t}^{t+b} (r_j - \widehat{J}_{\pi_i}), \quad \widehat{J}_{\pi_i} = \frac{1}{\tau} \sum_{j=(i-1)\tau+1}^{i\tau} r_j.$$

In the following, we denote by $\tau' := \tau - b$ the maximum number of data that we can collect from every phase. At the end of each phase, we store all or a subset of the (x_t, a_t, R_t) tuples in our replay buffer $\mathcal{R}$. We extract feature $\phi(x,a) \in \mathbb{R}^d$ for the state-action pair (x,a) and let $\mathcal{F} \subseteq \{f : \mathbb{R}^d \mapsto \mathbb{R}\}$ be a class of functions that we use to approximate the Q-functions. For phase i, we propose the following method to estimate Q_{π_i}: $\widehat{Q}_{\pi_i}(x,a) = \widehat{f}(\phi(x,a))$, where $\widehat{f} \in \arg\min_{f \in \mathcal{F}} \ell_i(f)$ and

$$\ell_i(f) := \frac{1}{\tau'} \sum_{t=(i-1)\tau+1}^{i\tau-b} (f(\phi(x_t,a_t)) - R_t)^2. \tag{7.1}$$

Suppose that we store all the data from the previous phases in the replay buffer, then in order to estimate the average of the Q-functions of the first k phases, i.e., $\widehat{Q}_k$, we propose to use the heuristic that minimizes the average of the k squared loss functions defined in Eq. (7.1), i.e., $\frac{1}{k}\sum_{i=1}^k \ell_i(f)$.

Subsampling and coreset. In practice, due to the high memory cost, it may be hard to store all the data from the previous phases in the replay buffer. We found that a simple strategy to resolve this issue is to begin with storing all the data from every phase, and add a limit on size of the replay buffer. When the buffer size exceeds the limit, we eliminate a subset of the data uniformly at random.

Another approach is to sample a subset of size s from the data collected in each phase. Denote this subset by $\mathcal{R}_i$ for phase i. Thus in the k-th phase, we have τ' data $\mathcal{D}_k$ from the current phase as well as $s(k-1)$ data from the replay buffer $\mathcal{R}$. First, suppose that the s data points are sampled uniformly at random. We can then minimize the following objective:

$$\min_{f \in \mathcal{F}} \frac{1}{k} \Big(\ell_i(f) + \sum_{i=1}^{k-1} \widehat{\ell}_i(f) \Big), \tag{7.2}$$

where $\widehat{\ell}_i(f) := \frac{1}{s}\sum_{(x_t,a_t,R_t)\in\mathcal{R}_i} (f(\phi(x_t,a_t)) - R_t)^2$ is an unbiased estimate of $\ell_i(f)$. Further, uniform sampling is not the only way to construct an unbiased estimate. In fact, for any discrete distribution, with PMF $q = \{q_t\}$, over the τ' data in $\mathcal{D}_i$, we can sample s data points according to q and construct

$$\widehat{\ell}_i(f) := \frac{1}{\tau'} \sum_{(x_t,a_t,R_t)\in\mathcal{R}_i} \frac{1}{q_t} (f(\phi(x_t,a_t)) - R_t)^2, \tag{7.3}$$

[1] This is a practical implementation of Eq. (6.2), i.e., we do not split the data in each phase into blocks.

in order to obtain an unbiased estimate of $\ell_i(f)$. As shown by Bachem et al. (2017), by choosing $q_t \propto (f(\phi(x_t, a_t)) - R_t)^2$, we can minimize the variance of $\widehat{\ell}_i(f)$ for any fixed f. In the following, we call the subset of data sampled according to this distribution a *coreset* of the data. In the experiments in Section 8, we show that thanks to the variance reduction effect, sampling a coreset often produces better performance than sampling a subset uniformly at random, especially when the rewards are sparse.

Comparison to Abbasi-Yadkori et al. (2019). Our algorithm can be considered as an efficient implementation of POLITEX (Abbasi-Yadkori et al., 2019) via experience replay. In the original POLITEX algorithm, the Q-functions are estimated using Eq. (7.1) for each phase, and all the functions are stored in memory. When an agent interacts with the environment, it needs to evaluate all the Q-functions in order to obtain the probability of each action. This implies that the time complexity of computing action probabilities increases with the number of phases, and as a result the algorithm is hard to scale to a large number of phases. Although we can choose to evaluate a random subset of the Q-functions to estimate the action probabilities in POLITEX, our implementation via experience replay can still be faster since we only need to evaluate a single function, trained with replay data, to take actions.

8. Experiments

In this section, we evaluate our implementations empirically. We make comparisons with several baselines in two control environments.

Environments. We use two control environments with the simulators described in Tassa et al. (2018). Both environments are episodic with episode length 1000. The environments we evaluate are:

- *Cart-pole* (Barto et al., 1983): The goal of this environment is to balance an unactuated pole by applying forces to a cart at its base. We discretize the continuous force to 5 values: $\{-2, -1, 0, 1, 2\}$. The reward at each time step is a real number in $[0, 1]$. We end episodes early when the pole falls (we use rewards less than 0.5 an indicator for falling), and assume zero reward for the remaining steps when reporting results.
- *Ball-in-cup*: Here, an actuated planar receptacle can translate in the vertical plane in order to swing and catch a ball attached to its bottom. This task has a sparse reward: 1 when the ball is in the cup, and 0 otherwise. We discretize the two-dimensional continuous action space to a 3×3 grid, i.e., 9 actions in total.

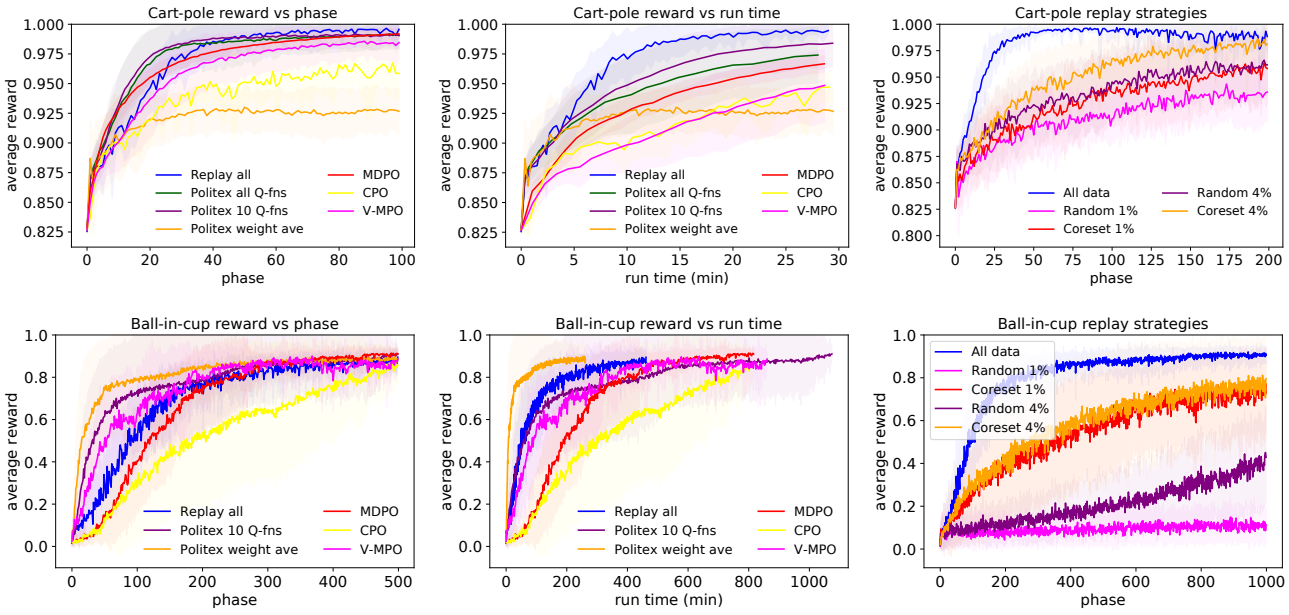
Algorithms. We compare the following algorithms: our proposed implementation of POLITEX using experience replay, POLITEX using weight averaging, the original POLITEX algorithm, the variant of POLITEX that averages 10 randomly selected Q-functions, the mirror descent policy optimization (MDPO) algorithm (Tomar et al., 2020), the constrained policy optimization (CPO) (Achiam et al., 2017), and the V-MPO algorithm (Song et al., 2019b).

For all the algorithms, we extract Fourier basis features (Konidaris et al., 2011) from the raw observations for both environments: for Cart-pole, we use 4 bases and for Ball-in-cup we use 2 bases. For variants of POLITEX we construct the state-action features $\phi(x, a)$ by block one-hot encoding, i.e., we partition the $\phi(x, a)$ vector into $|\mathcal{A}|$ blocks, and set the a-th block to be the Fourier features of x and other blocks to be zero. We approximate Q-functions using neural networks with one hidden layer and ReLU activation: the width of the hidden layer is 50 for Cart-pole and 250 for Ball-in-cup. For MDPO, CPO and V-MPO, the algorithms use a policy network whose input and output are the state feature and action probability, respectively. We use one hidden layer networks with width 50 for Cart-pole and 250 for Ball-in-cup. These three algorithms also need a value network, which takes the state feature as input and outputs its value for the current policy. For both environments, we use one hidden layer network with width 50.

For Cart-pole, we choose phase length $\tau = 10^4$ and for Ball-in-cup, we choose $\tau = 2 \times 10^4$. Since the environments are episodic, we have multiple episodes in each phase. For Cart-pole, since the training usually does not need a large number of phases, we do not set limit on the size of the replay buffer, whereas for Ball-in-cup, we set the limit as 2×10^6. For our algorithm and the variants of POLITEX, we choose parameter η in $\{5, 10, 20, 40, 80, 160\}$ and report the result of each algorithm using the value of η that produces the best average reward. For MDPO, CPO and V-MPO, we note that according to Eq. (4.1), the KL regularization coefficient between policies is the reciprocal of the η parameter in our algorithm, we also run these baseline algorithms in the same range of η and report the best result. In addition, in CPO, we set the limit in KL divergence between adjacent policies 0.001η, which, in our experiments, leads to good performance. We treat the length of Monte Carlo estimate b as a hyper parameter, so we choose it in $\{100, 300\}$ and report the best performance.

Results. We run each algorithm in each environment 20 times and report the average results as well as the standard deviation of the 20 runs (shaded area). We report the average reward in each phase before training on the data collected during the phase. The results are presented in Figure 1. Every run is conducted on a single P100 GPU.

From Fig. 1(a), we can see that the iteration complexity (average reward vs number of phases) and final performance of the experience replay implementation using all the data from



(a) Iteration complexity (b) Training speed (c) Replay subsampling strategies

Figure 1. Experiments on the Cart-pole environment (top) and Ball-in-cup (bottom).

past phases (blue curve) are similar to many state-of-the-art algorithms, such as POLITEX and V-MPO. Meanwhile, according to Fig. 1(b), the experience replay implementation achieves strong performance in training speed, i.e., best performance at 30 minute in Cart-pole and second best performance at 200 minute in Ball-in-cup. We note here that the training time includes both the time for data collection, i.e., interacting with the environment simulator and the training of a new policy. The observation that the experience replay based implementation is faster than the original POLITEX and the implementation that uses 10 Q-functions can be explained by the fact that the replay-based algorithm only uses a single Q-function and thus is faster at data collection, as discussed in Section 7.2. In addition, that the replay-based algorithm runs faster than MDPO, CPO, and V-MPO can be explained by its simplicity: in variants of POLITEX, we only need to approximate the Q-function, whereas in MDPO, CPO, and V-MPO, we need to train both the policy and value networks.

The POLITEX weight averaging scheme achieves the best performance on Ball-in-cup in both iteration complexity and training speed, but does not converge to a solution that is sufficiently close to the optimum on Cart-pole. Notice that for weight averaging, we used the initialization technique mentioned in Section 7.1. Considering the consistent strong performance of experience replay in both environments, we still recommend applying experience replay when using non-linear function approximation.

In Fig. 1(c), we can see that when we only sample a small subset (1% or 4%) of the data to store in the replay buffer, the coreset technique described in Section 7.2 achieves better performance due to the variance reduction effect. This improvement is quite significant in the Ball-in-cup environment, which has sparse rewards. Notice that sampling coreset only adds negligible computational overhead during training, and thus we recommend the coreset technique when there exists a strict memory constraint.

9. Discussion

The main contributions of our work are an improved analysis and practical implementation of POLITEX. On the theoretical side, we show that POLITEX obtains an $O(\sqrt{T})$ high-probability regret bound in uniformly mixing average-reward MDPs with linear function approximation, which is the first such bound for a computationally efficient algorithm. The main limitation of these result is that, similarly to previous works, they hold under somewhat strong assumptions which circumvent the need for exploration. An interesting future work direction would be to relax these assumptions and incorporate explicit exploration instead.

On the practical side, we propose an efficient implementation with neural networks that relies on experience replay, a standard tool in modern deep RL. Our work shows that experience replay and KL regularization can both be viewed as approximately implementing mirror descent policy updates within a policy iteration scheme. This provides an online-learning justification for using replay buffers in policy iteration, which is different than the standard explanation for their success. Our work also suggests a new objective

for storing and prioritizing replay samples, with the goal of approximating the average of value functions well. This goal has some similarities with continual learning, where experience replay has also lead to empirical successes. One interesting direction for future work would be to explore other continual learning techniques in the approximate implementation of mirror descent policy updates.

References

Abbasi-Yadkori, Y., Bartlett, P., Bhatia, K., Lazic, N., Szepesvari, C., and Weisz, G. POLITEX: Regret bounds for policy iteration using expert prediction. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3692–3702. PMLR, 09–15 Jun 2019.

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International Conference on Machine Learning*, pp. 22–31, 2017.

Bachem, O., Lucic, M., and Krause, A. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

Bartlett, P. L. Regal: A regularization based algorithm for reinforcement learning in weakly communicating mdps. In *In Proceedings of the 25th Annual Conference on Uncertainty in Artificial Intelligence*, 2009.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, (5):834–846, 1983.

Borsos, Z., Mutnỳ, M., and Krause, A. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020.

Cao, X.-R. Single sample path-based optimization of Markov chains. *Journal of optimization theory and applications*, 100(3):527–548, 1999.

Cesa-Bianchi, N. and Lugosi, G. *Prediction, learning, and games*. Cambridge university press, 2006.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Cho, G. E. and Meyer, C. D. Comparison of perturbation bounds for the stationary distribution of a Markov chain. *Linear Algebra and its Applications*, 335(1-3):137–150, 2001.

Degrave, J., Abdolmaleki, A., Springenberg, J. T., Heess, N., and Riedmiller, M. Quinoa: a Q-function you infer normalized over actions. *arXiv preprint arXiv:1911.01831*, 2019.

Even-Dar, E., Kakade, S. M., and Mansour, Y. Online Markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773, 2020.

Fruit, R., Pirotta, M., Lazaric, A., and Ortner, R. Efficient bias-span-constrained exploration-exploitation in reinforcement learning. In *International Conference on Machine Learning*, 2018.

Hao, B., Lazic, N., Abbasi-Yadkori, Y., Joulani, P., and Szepesvari, C. Provably efficient adaptive approximate policy iteration. *arXiv preprint arXiv:2002.03069*, 2020.

Heess, N., Wayne, G., Silver, D., Lillicrap, T., Erez, T., and Tassa, Y. Learning continuous control policies by stochastic value gradients. *Advances in Neural Information Processing Systems*, 28:2944–2952, 2015.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

Jian, Q., Fruit, R., Pirotta, M., and Lazaric, A. Exploration bonus for regret minimization in discrete and continuous average reward mdps. In *Advances in Neural Information Processing Systems*, pp. 4891–4900, 2019.

Jin, C., Netrapalli, P., Ge, R., Kakade, S. M., and Jordan, M. I. A short note on concentration inequalities for random vectors with subgaussian norm. *arXiv preprint arXiv:1902.03736*, 2019.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pp. 267–274, 2002.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Konidaris, G., Osentoski, S., and Thomas, P. Value function approximation in reinforcement learning using the fourier

basis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In *ICLR (Poster)*, 2016.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pp. 6467–6476, 2017.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Neu, G., György, A., Szepesvári, C., and Antos, A. Online Markov decision processes under bandit feedback. *IEEE Transactions on Automatic Control*, 59(3):676–691, December 2014.

Ouyang, Y., Gagrani, M., Nayyar, A., and Jain, R. Learning unknown Markov decision processes: A Thompson sampling approach. In *Advances in Neural Information Processing Systems*, pp. 1333–1342, 2017.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *International Conference on Learrning Representations (ICLR)*, 2016.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Seneta, E. Coefficients of ergodicity: structure and applications. *Advances in applied probability*, pp. 576–590, 1979.

Seneta, E. Perturbation of the stationary distribution measured by ergodicity coefficients. *Advances in Applied Probability*, 20(1):228–230, 1988.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., Heess, N., Belov, D., Riedmiller, M. A., and Botvinick, M. M. V-MPO: on-policy maximum a posteriori policy optimization for discrete and continuous control. *CoRR*, abs/1909.12238, 2019a. URL http://arxiv.org/abs/1909.12238.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., et al. V-mpo: on-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019b.

Talebi, M. S. and Maillard, O.-A. Variance-aware regret bounds for undiscounted reinforcement learning in mdps. In *Algorithmic Learning Theory*, 2018.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., Casas, D. d. L., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., et al. DeepMind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Tomar, M., Shani, L., Efroni, Y., and Ghavamzadeh, M. Mirror descent policy optimization. *arXiv preprint arXiv:2005.09814*, 2020.

Tropp, J. A. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12 (4):389–434, 2012.

Vieillard, N., Kozuno, T., Scherrer, B., Pietquin, O., Munos, R., and Geist, M. Leverage the average: an analysis of regularization in rl. *arXiv preprint arXiv:2003.14089*, 2020a.

Vieillard, N., Scherrer, B., Pietquin, O., and Geist, M. Momentum in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 2529–2538, 2020b.

Wei, C.-Y., Jafarnia-Jahromi, M., Luo, H., and Jain, R. Learning infinite-horizon average-reward mdps with linear function approximation. *arXiv preprint arXiv:2007.11849*, 2020a.

Wei, C.-Y., Jahromi, M. J., Luo, H., Sharma, H., and Jain, R. Model-free reinforcement learning in infinite-horizon average-reward Markov decision processes. In *International Conference on Machine Learning*, pp. 10170–10180. PMLR, 2020b.

Yin, D., Farajtabar, M., and Li, A. SOLA: Continual learning with second-order loss approximation. *arXiv preprint arXiv:2006.10974*, 2020.

Yu, B. Rates of convergence for empirical processes of stationary mixing sequences. *The Annals of Probability*, 22(1):94–116, 1994.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70:3987, 2017.