Provably Correct Optimization and Exploration with Non-linear Policies

Fei Feng [1] **Wotao Yin** [1] **Alekh Agarwal** [2] **Lin Yang** [3]

Abstract

Policy optimization methods remain a powerful workhorse in empirical Reinforcement Learning (RL), with a focus on neural policies that can easily reason over complex and continuous state and/or action spaces. Theoretical understanding of strategic exploration in policy-based methods with non-linear function approximation, however, is largely missing. In this paper, we address this question by designing ENIAC, an actor-critic method that allows non-linear function approximation in the critic. We show that under certain assumptions, e.g., a bounded eluder dimension d for the critic class, the learner finds to a near-optimal policy in $\widetilde{O}(\mathrm{poly}(d))$ exploration rounds. The method is robust to model misspecification and strictly extends existing works on linear function approximation. We also develop some computational optimizations of our approach with slightly worse statistical guarantees, and an empirical adaptation building on existing deep RL tools. We empirically evaluate this adaptation, and show that it outperforms prior heuristics inspired by linear methods, establishing the value in correctly reasoning about the agent's uncertainty under non-linear function approximation.

1. Introduction

The success of reinforcement learning (RL) in many empirical domains largely relies on developing policy gradient methods with deep neural networks (Schulman et al., 2015; 2017; Haarnoja et al., 2018). The techniques have a long history in RL (Williams, 1992; Sutton et al., 1999; Konda & Tsitsiklis, 2000). A number of theoretical results study their convergence properties (Kakade & Langford, 2002; Scherrer & Geist, 2014; Geist et al., 2019; Abbasi-Yadkori et al., 2019; Agarwal et al., 2020c; Bhandari & Russo, 2019) when the agent has access to a distribution over states which is sufficiently exploratory, such as in a generative model. However, unlike their value- or model-based counterparts, the number of policy-based approaches which actively explore and provably find a near-optimal policy remains relatively limited, and restricted to tabular (Shani et al., 2020) and linear function approximation (Cai et al., 2020; Agarwal et al., 2020a) settings. Given this gap between theory and the empirical literature, it is natural to ask how we can design provably sample-efficient policy-based methods for RL that allow the use of general function approximation, such as via neural networks.

In this paper we design an actor-critic method with general non-linear function approximation: *Exploratory Non-linear Incremental Actor Critic (ENIAC)*. Our method follows a similar high-level framework as Agarwal et al. (2020a), but with a very different bonus function in order to reason about the uncertainty of our non-linear critic. In each iteration, we use the bonus to learn an optimistic critic, so that optimizing the actor with it results in exploration of the previously unseen parts of the environment. Unlike Agarwal et al. (2020a), we allow non-linear function approximation in the critic, which further parameterizes a non-linear policy class through Soft Policy Iteration (SPI) (Even-Dar et al., 2009; Haarnoja et al., 2018; Geist et al., 2019; Abbasi-Yadkori et al., 2019; Agarwal et al., 2020a) or Natural Policy Gradient (NPG) (Kakade, 2001; Peters & Schaal, 2008; Agarwal et al., 2020c) updates. Theoretically, we show that if the critic function class has a bounded *eluder dimension* (Russo & Van Roy, 2013) d, then our algorithm outputs a near-optimal policy in $\mathrm{poly}(d)$ number of interactions, with high probability, for both SPI and NPG methods.

Following the recent work on non-linear value-based methods by Wang et al. (2020), the bonus function is based on the *range of values* (or the width function) predicted at a particular state-action pair by the critic function which accurately predicts the observed returns. Hence, this function characterizes how uncertain we are about a state-action pair given the past observations. The value-based method in Wang et al. (2020) relies on solving the value iteration problem using the experience, which introduces dependence issues across different stages of the algorithm. But, we directly use the

[1]Department of Mathematics, University of California, Los Angeles, Los Angeles, CA, USA. [2]Microsoft Research, Redmond, WA, USA. [3]Department of Electrical and Computer Engineering, University of California, Los Angeles, Los Angeles, CA, USA. Correspondence to: Alekh Agarwal <alekha@microsoft.com>, Lin Yang <linyang@ee.ucla.edu>.

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

width function as our exploration bonus and have a simpler sub-sampling design that that in Wang et al. (2020). Under mild assumptions, our bonus function can be computed in a time polynomially depending on the size of the current dataset. We also provide a heuristic method to compute the bonus functions for neural networks. Furthermore, all our results are robust to model misspecification and do not require an explicit specification about the transition dynamics as used in Wang et al. (2020).

The technique for tracking the uncertainty of general non-linear setting is especially established with careful design of hyperparameters and is not a direct extension of the linear case as in Agarwal et al. (2020a) whose uncertainty is easily summarized using a covariance matrix and the potential function argument to analyze the evolution of that bonus is quite standard.

To further improve the efficiency, we develop variants of our methods that require no bonus computation in the execution of the actor. The key idea is to replace certain conditional exploration steps triggered by the bonus with a small uniform exploration. Note that this uniform exploration is in addition to the optimistic reasoning, thus different from vanilla ϵ-greedy methods. The bonus is later incorporated while updating the critic, which is a significant optimization in settings where the actor runs in real-time with resource constrained hardware such as robotic platforms (Pan et al., 2018), and plays well with existing asynchronous actor-critic updates (Mnih et al., 2016).

We complement our theoretical analysis with empirical evaluation on a continuous control domain requiring non-linear function approximation, and show the benefit of using a bonus systematically derived for this setting over prior heuristics from both theoretical and empirical literature.

Related Work The rich literature on exploration in RL primarily deals with tabular (Kearns & Singh, 2002; Brafman & Tennenholtz, 2002; Jaksch et al., 2010; Jin et al., 2018) and linear (Yang & Wang, 2020; Jin et al., 2020) settings with value- or model-based methods. Recent papers (Shani et al., 2020; Cai et al., 2020; Agarwal et al., 2020a) have developed policy-based methods also in the same settings. Of these, our work directly builds upon that of Agarwal et al. (2020a), extending it to non-linear settings.

For general non-linear function approximation, a series of papers provide statistical guarantees under structural assumptions (Jiang et al., 2017; Sun et al., 2019; Dann et al., 2018), but these do not lend themselves to computationally practical versions. Other works (Du et al., 2019; Misra et al., 2020; Agarwal et al., 2020b) study various latent variable models for non-linear function approximation in model-based settings. The notion of eluder dimension (Russo & Van Roy, 2013) used in our theory has been previously used to study RL in deterministic settings (Wen & Van Roy, 2013). Most related to our work is the recent value-based technique of Wang et al. (2020), which describes a UCB-VI style algorithm with statistical guarantees scaling with eluder dimension. In this paper, we instead study policy-based methods, which provide better robustness to misspecification in theory and are more amenable to practical implementation.

Notation Given a set $\mathcal{A}$, we denote by $|\mathcal{A}|$ its cardinality, $\Delta(\mathcal{A})$ the set of all distributions over $\mathcal{A}$, and $\mathrm{Unif}(\mathcal{A})$ the uniform distribution over $\mathcal{A}$. We use $[n]$ for the integer set $\{1, \ldots, n\}$. Let $a, b \in \mathbb{R}^n$. We denote by $a^\top b$ the inner product between a and b and $\|a\|_2$ the Euclidean norm of a. Given a matrix A, we use $\|A\|_2$ for its spectral norm. Given a function $f : \mathcal{X} \to \mathbb{R}$ and a finite dataset $\mathcal{Z} \subset \mathcal{X}$, we define $\|f\|_{\mathcal{Z}} := \sqrt{\sum_{x \in \mathcal{Z}} f(x)^2}$. We abbreviate Kullback-Leibler divergence to $\mathbf{KL}$ and use O for leading orders in asymptotic upper bounds and $\widetilde{O}$ to hide the polylog factors.

2. Setting

Markov Decision Process In this paper, we focus on the discounted Markov Decision Process (MDP) with an infinite horizon. Each MDP is described as $\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ is a possibly infinite state space, $\mathcal{A}$ is a finite action space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ specifies a transition kernel, $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is a reward function, and $\gamma \in (0, 1)$ is a discount factor. At each time step, the agent observes a state s and selects an action a according to a *policy* $\pi : \mathcal{S} \to \Delta(\mathcal{A})$. The environment then transitions to a new state s' with probability $P(s'|s, a)$ and the agent receives an instant reward $r(s, a)$.

For a policy π, its Q-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as:

$$Q^\pi(s, a, r) := \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right],$$

where the expectation is taken over the trajectory following π. And the value function is $V^\pi(s, r) := \mathbb{E}_{a \sim \pi(\cdot|s)}[Q^\pi(s, a, r)]$. From V^π and Q^π, the advantage function of π is: $A^\pi(s, a, r) = Q^\pi(s, a, r) - V^\pi(s, r), \forall s \in \mathcal{S}, a \in \mathcal{A}$. We ignore r in V, Q or A, if it is clear from the context. Besides value, we also define the discounted state-action distribution $d^\pi_{\tilde{s}}(s, a)$ induced by π as:

$$d^\pi_{\tilde{s}}(s, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s, a_t = a | s_0 = \tilde{s}),$$

where $\mathrm{Pr}^\pi(s_t = s, a_t = a | s_0 = \tilde{s})$ is the probability of reaching (s, a) at the t_{th} step starting from $\tilde{s}$ following π. Similarly, we define $d^\pi_{\tilde{s},\tilde{a}}(s, a)$ if the agent starts from state $\tilde{s}$ followed by action $\tilde{a}$ and follows π thereafter. For any distribution $\nu \in \Delta(\mathcal{S} \times \mathcal{A})$, we denote by $d^\pi_\nu(s, a) := \mathbb{E}_{(\tilde{s},\tilde{a}) \sim \nu} [d^\pi_{(\tilde{s},\tilde{a})}(s, a)]$ and $d^\pi_\nu(s) := \sum_a d^\pi_\nu(s, a)$.

Given an initial distribution $\rho \in \Delta(\mathcal{S})$, we define $V_\rho^\pi := \mathbb{E}_{s_0\sim\rho}[V^\pi(s_0)]$. Similarly, if $\nu \in \Delta(\mathcal{S}\times\mathcal{A})$, we define $V_\nu^\pi := \mathbb{E}_{(s_0,a_0)\sim\nu}[Q^\pi(s_0,a_0)]$. The goal of RL is to find a policy in some policy space Π such that its value with respect to an initial distribution ρ_0 is maximized, i.e.,

$$\underset{\pi\in\Pi}{\text{maximize}}\, V_{\rho_0}^\pi.$$

Without loss of generality, we consider the RL problems starting from the unique initial state s_0 in the later context. All the results straightforwardly apply to arbitrary ρ_0.

Function Class and Policy Space Let $\mathcal{F} := \{f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}\}$ be a general function class. We denote by $\Pi_\mathcal{F} := \{\pi_f, f\in\mathcal{F}\}$ a policy space induced by applying the softmax transform to functions in $\mathcal{F}$, i.e.,

$$\pi_f(a|s) := \frac{\exp(f(s,a))}{\sum_{a'\in\mathcal{A}}\exp(f(s,a'))}.$$

For the ease of presentation, we assume there exists a function $f\in\mathcal{F}$ such that, for all s, $\pi_f(\cdot|s)$ is a uniform distribution[1] on $\mathcal{A}$. Given $\mathcal{F}$, we define its *function-difference* class $\Delta\mathcal{F} := \{\Delta f \mid \Delta f = f - f',\ f, f'\in\mathcal{F}\}$ and the width function on $\Delta\mathcal{F}$ for any $(s,a)\in\mathcal{S}\times\mathcal{A}$ as:

$$w(\Delta\mathcal{F}, s, a) := \sup_{\Delta f\in\Delta\mathcal{F}} \Delta f(s,a). \tag{1}$$

Our width is defined on $\Delta\mathcal{F}$ instead of $\mathcal{F}$, where the latter is adopted in (Russo & Van Roy, 2013) and (Wang et al., 2020). These two formulations are essentially equivalent.

If $\mathcal{F}$ can be smoothly parameterized by $\theta\in\mathbb{R}^d$, we further introduce the (centered) *tangent class* of $\mathcal{F}_\theta$ as:

$$\mathcal{G}_\mathcal{F} := \{g_\theta^u \mid g_\theta^u(s,a) := u^\top\nabla_\theta \log\pi_{f_\theta}(s,a), \\ u\in\mathcal{U}, f_\theta\in\mathcal{F}\}, \tag{2}$$

where $\mathcal{U}\subset\mathbb{R}^d$ is some bounded set. We define the function-difference class $\Delta\mathcal{G}_\mathcal{F}$ and the width function $w(\Delta\mathcal{G}_\mathcal{F}, s, a)$ for $\mathcal{G}_\mathcal{F}$ accordingly.

Next, given a function class $\mathcal{F}$, we consider RL on the induced policy space $\Pi_\mathcal{F}$. If $\mathcal{F}$ is non-smooth, we apply SPI as the policy optimization routine while approximating Q-values with $\mathcal{F}$; if $\mathcal{F}$ is smoothly parameterized by θ, we can alternatively apply NPG for policy optimization and use $\mathcal{G}_\mathcal{F}$ to approximate advantage functions. The function-difference classes will be used to design bonuses.

3. Algorithms

In this section, we describe our algorithm, *Exploratory Non-Linear Incremental Actor Critic* (ENIAC), which takes a function class $\mathcal{F}$ and interacts with an RL environment to learn a good policy. The formal pseudo-code is presented in Algorithm 1. We explain the high-level design and steps in the algorithm in this section and present the main results in the next section.

[1]This requirement is not strict, our algorithms and analysis apply for any distribution that are supported on all actions.

Algorithm 1 Exploratory Non-Linear Incremental Actor Critic (ENIAC)

Input: Function class $\mathcal{F}$.
Hyperparameters: N, K, $\beta > 0$, $\alpha\in(0,1)$.
For all $s\in\mathcal{S}$, initialize $\pi^1(\cdot|s) = \text{Unif}(\mathcal{A})$.
Initialize experience buffer $\mathcal{Z}^0$ as an empty set.
for $n = 1$ **to** N **do**
  Generate K samples: $\{s_i, a_i\}_{i=1}^K \sim d_{s_0}^{\pi^n}$;
  Merge training set: $\mathcal{Z}^n \leftarrow \mathcal{Z}^{n-1}\cup\{s_i,a_i\}_{i=1}^K$;
  Let $\rho_{\text{cov}}^n := \text{Unif}(d_{s_0}^{\pi^1},\dots,d_{s_0}^{\pi^n})$;
  Define a bonus function b^n using (12) or (13);
  Update the policy using Algorithm 2: $\pi^{n+1}\leftarrow$ Policy Update(ρ_{cov}^n, b^n, α).
end for
Output: $\text{Unif}(\pi_2,\pi_3,\dots,\pi_{N+1})$

Algorithm 2 Policy Update

Input: Fitting distribution ρ, bonus function b, α.
Hyperparameters: $T > 0$, $M > 0$, $\eta > 0$.
Initialize π_0 using (3) or (4).
for $t = 0$ **to** $T-1$ **do**
  Generate M samples from ρ using (5) or (9);
  Fit critic to the M samples using (6) or (10);
  Actor update using (7), (8), or (11) to obtain π_{t+1};
end for
Output: $\text{Unif}(\pi_0,\pi_1,\dots,\pi_{T-1})$

3.1. High-level Framework

At a high-level, ENIAC solves a series of policy optimization problems in a sequence of carefully designed MDPs. Each MDP is based on the original MDP, but differs in the choice of an *initial state distribution* and a *reward bonus*. We use them to induce optimistic bias to encourage exploration. Through the steps of the algorithm, the initial distribution gains coverage, while the bonus shrinks so that good policies in the modified MDPs eventually yield good policies in the original MDP as well.

A key challenge in large state spaces is to quantify the notion of *state coverage*, which we define using the function class $\mathcal{F}$. We say a distribution ρ_{cov} provides a good coverage if any function $f\in\mathcal{F}$ that has a small prediction error on data sampled from ρ_{cov} also has a small prediction error under the state distribution d^π for any other policy π. In tabular settings, this requires ρ_{cov} to visit each state, while coverage in the feature space suffices for linear MDPs (Jin et al., 2020; Yang & Wang, 2020).

In ENIAC, we construct such a covering distribution ρ_{cov} iteratively, starting from the state distribution of a uniform policy and augmenting it gradually as new policies visit previously unexplored parts of the MDP. Concretely, we maintain a *policy cover* $\{\pi^1,\pi^2,\dots\}$, which initially con-

tains only a random policy, π^1, (Line 3 of Algorithm 1). At iteration n, the algorithm lets ρ^n_{cov} be a uniform mixture of $\{d^{\pi^1}_{s_0}, d^{\pi^2}_{s_0}, \ldots, d^{\pi^n}_{s_0}\}$ (line 8).

Having obtained the cover, we move on to induce the reward bonus by collecting a dataset of trajectories from ρ^n_{cov} (line 6).[2] These collected trajectories are used to identify a set $\mathcal{K}^n$ of state-action pairs covered by ρ^n_{cov}: any functions $f, g \in \mathcal{F}$ that are close under ρ^n_{cov} also approximately agree with each other for all $s, a \in \mathcal{K}^n$. We then create a *reward bonus*, b^n (Line 9, formally defined later), toward encouraging explorations outside the set $\mathcal{K}^n$.

Finally, taking ρ^n_{cov} as the initial distribution and the bonus augmented reward $r + b^n$ as the reward function, we find a policy π that approximately maximizes $V^{\pi}_{\rho^n_{\text{cov}}}(r + b^n)$ (line 10). It can be shown that this policy either *explores* by reaching new parts of the MDP or *exploits* toward identifying a near optimal policy. We then add this policy to our cover and proceed to the next epoch of the algorithm .

Within this high-level framework, different choices of the policy update and corresponding bonus functions induce different concrete variants of Algorithm 1. We describe these choices below.

3.2. Policy Optimization

In this section, we describe our policy optimization approach, given a policy cover ρ and a reward bonus b. We drop the dependence on epoch n for brevity, and recall that the goal is to optimize $V^{\pi}_{\rho}(r + b)$. We present two different actor critic style optimization approaches: Soft Policy Iteration (SPI) and Natural Policy Gradient (NPG), which offer differing tradeoffs in generality and practical implementation. SPI is amenable to arbitrary class $\mathcal{F}$, while NPG requires second-order smoothness. On the other hand, NPG induces fully convex critic objective for any class $\mathcal{F}$, and is closer to popular optimization methods like TRPO, PPO and SAC. Our presentation of both these methods is adapted from (Agarwal et al., 2020c), and we describe the overall outline of these approaches in Algorithm 2, with the specific update rules included in the rest of this section.

For each approach, we provide a *sample-friendly* version and a *computation-friendly* version for updating the policy. The two versions of updating methods only differ in the initialization and actor updating steps. The computation-friendly version provides a policy that can be executed efficiently while being played. The sample-friendly version requires to compute the bonus function during policy execution but saves samples up to $\text{poly}(|\mathcal{A}|)$ factors. We now describe these procedures in more details.

[2] In the Algorithm 1, only π^n is rolled out as the samples can be combined with historical data to form samples from ρ^n_{cov}.

3.2.1. Policy Initialization

For both SPI and NPG approaches, we use the following methods to initialize the policy.

Sample-friendly initialization. Given bonus b, we define $\mathcal{K} := \{(s, a) \mid b(s, a) = 0\}$. We abuse the notation $s \in \mathcal{K}$ if $b(s, a) = 0,\ \forall a \in \mathcal{A}$. We initialize the policy as follows.

$$\pi_0(\cdot|s) = \begin{cases} \text{Unif}(\mathcal{A}) & s \in \mathcal{K}; \\ \text{Unif}(\{a \in \mathcal{A} : (s, a) \notin \mathcal{K}\}) & \text{o.w.} \end{cases} \tag{3}$$

Here the policy selects actions uniformly for states where all actions have been well-explored under ρ and only plays actions that are not well-covered in other states. Note that such a policy can be represented by b and a function $f \in \mathcal{F}$.

Computation-friendly initialization. The computation-friendly method does not recompute the set $\mathcal{K}$ and initialize the policy to be purely random, i.e.,

$$\pi_0(\cdot|s) = \text{Unif}(\mathcal{A}),\ \forall s \in \mathcal{S}. \tag{4}$$

3.2.2. SPI Policy Update

For each iteration, t, we first generate M (some parameter to be determined) Q-value samples with the input distribution ρ as the initial distribution:

$$\{s_i, a_i, \widehat{Q}^{\pi_t}(s_i, a_i, r + b)\}_{i=1}^{M},\ (s_i, a_i) \sim \rho, \tag{5}$$

where $\widehat{Q}^{\pi_t}$ is an unbiased estimator of Q^{π_t} (see, e.g., Algorithm 3 in the Appendix). Then we fit a critic to the above samples by setting f_t as a solution of :

$$\underset{f \in \mathcal{F}}{\text{minimize}} \sum_{i=1}^{M} \left(\widehat{Q}^{\pi_t}(s_i, a_i, r + b) - b(s_i, a_i) - f(s_i, a_i)\right)^2. \tag{6}$$

Here we offset the fitting with the initial bonus to maintain consistency with linear function approximation results, where a non-linear bonus introduces an approximation error (Jin et al., 2020; Agarwal et al., 2020a). Note that for the SPI, we do not require f to be differentiable.

Based on the critic, we update the actor to a new policy. There are two update versions: one is more sample-efficient, the other is more computational-convenient.

Sample-friendly version. For this version, we only update the policy on states $s \in \mathcal{K}$ since our critic is unreliable elsewhere. For $s \notin \mathcal{K}$, we keep exploring previously unknown actions by simply sticking to the initial policy. Then the policy update rule is:

$$\pi_{t+1}(a|s) \propto \pi_t(a|s) \exp\left(\eta f_t(s, a) \mathbf{1}\{s \in \mathcal{K}\}\right), \tag{7}$$

where $\eta > 0$ is a step size to be specified. Note that since $b(s, a) \equiv 0$ for $s \in \mathcal{K}$, Equation (7) is equivalent to $\pi_{t+1}(a|s) \propto \pi_t(a|s) \exp\left(\eta(f_t(s, a) + b(s, a)) \mathbf{1}\{s \in \mathcal{K}\}\right)$ where the initial bonus is added back.

Computation-friendly version. For this version, we remove the indicator function while allowing some probability of uniform exploration:

$$\pi'_{t+1}(a|s) \propto \pi'_t(a|s)\exp\big(\eta f_t(s,a)\big),$$
$$\pi_{t+1}(\cdot|s) = (1-\alpha)\cdot\pi'_{t+1} + \alpha\cdot\mathrm{Unif}(\mathcal{A}). \quad (8)$$

Above, $\{\pi'_t\}$ is an auxiliary sequence of policies initialized as π_0 and $\alpha > 0$. Note that for $s \in \mathcal{K}$, since $b(s,a) \equiv 0$ we still have $\pi'_{t+1}(a|s) \propto \pi'_t(a|s)\exp\big(\eta(f_t(s,a)+b(s,a))\big)$, i.e., the offset initial bonus is added back. Thus, compared with Equation (7), Equation (8) differs at: 1. α-probability random exploration for $s \in \mathcal{K}$; 2. update policy for $s \notin \mathcal{K}$ with a possibly not correct value (if $b(s,a) \neq 0$) but guarantees at least α-probability random exploration. Such a change creates a polynomial scaling with $|\mathcal{A}|$ in the sample complexity but saves us from computing bonuses during policy execution which is required by the sample-friendly version.

3.2.3. NPG Policy Update

NPG update shares the same structure as that for SPI. Recall that now the function class $\mathcal{F}$ is smoothly parameterized by θ. At each iteration t, we first generate M (some parameter to be determined) advantage samples from the input distribution ρ,

$$\{s_i, a_i, \widehat{A}^{\pi_t}(s_i,a_i,r+b)\}_{i=1}^M,\ (s_i,a_i)\sim\rho \quad (9)$$

where $\widehat{A}^{\pi_t}$ is an unbiased estimator of A^{π_t} (using Algorithm 3). We define $\bar{b}_t(s,a) := b(s,a) - \mathbb{E}_{a\sim\pi_t(\cdot|s)}[b(s,a)]$ as a centered version of the original bonus and $g_t(s,a) := \nabla_\theta \log \pi_{f_{\theta_t}}(s,a)$ to be the tangent features at θ_t. We then fit a critic to the bonus offset target $\widehat{A}^{\pi_t} - \bar{b}_t$ by setting u_t as a solution of:

$$\underset{u\in\mathcal{U}}{\text{minimize}} \sum_{i=1}^M \big(\widehat{A}^{\pi_t}(s_i,a_i,r+b) - \bar{b}_t(s_i,a_i) - u^\top g_t(s_i,a_i)\big)^2. \quad (10)$$

Compared to SPI, a big advantage is that the above critic objective is a linear regression problem, for which any off-the-shelf solver can be used, even with a large number of samples in high dimensions.

With the critic, we update the actor to generate a new policy as below.

Sample-friendly version. Similar to the sample-friendly version of SPI, we only update the policy on $s \in \mathcal{K}$ as:

$$\theta_{t+1} = \theta_t + \eta u_t,$$
$$\pi_{t+1}(a|s) \propto \exp(f_{\theta_{t+1}}(s,a)\mathbf{1}\{s\in\mathcal{K}\}), \quad (11)$$

where $\eta > 0$ is a step size to be specified.

We omit the details of the computation-friendly version, which is obtained similar to the counterpart in SPI.

3.3. Bonus Function

In this section, we describe the bonus computation given a dataset $\mathcal{Z}^n$ generated from some covering distribution ρ_{cov}. As described in previous subsections, the bonus assigns value 0 to state-action pairs that are well-covered by ρ_{cov} and a large value elsewhere. To measure the coverage, we use a width function (defined in Equation (1)) dependent on $\mathcal{Z}^n$. The bonus differs slightly for the SPI and NPG updates since SPI uses $\mathcal{F}$ for critic fit while NPG use $\mathcal{G}_\mathcal{F}$. Specifically, for the sample-friendly version, we take the following bonus function

$$b^n(s,a) = \mathbf{1}\{w(\widetilde{\mathcal{F}}^n, s, a) \geq \beta\}\cdot\frac{1}{1-\gamma}, \quad (12)$$

where for SPI,

$$\widetilde{\mathcal{F}}^n := \{\Delta f \in \Delta\mathcal{F} \mid \|\Delta f\|_{\mathcal{Z}^n} \leq \epsilon\}$$

and for NPG,

$$\widetilde{\mathcal{F}}^n := \{\Delta g \in \Delta\mathcal{G}_\mathcal{F} \mid \|\Delta g\|_{\mathcal{Z}^n} \leq \epsilon\}$$

with $\mathcal{G}_\mathcal{F}$ being the tangent class defined in Equation (2). Here β, ϵ are positive parameters to be determined. For the computation-friendly version, we scale up the bonus by a factor of $|\mathcal{A}|/\alpha$ to encourage more exploration, i.e.,

$$b^n(s,a) := \mathbf{1}\{w(\widetilde{\mathcal{F}}^n, s, a) \geq \beta\}\cdot\frac{|\mathcal{A}|}{(1-\gamma)\alpha}. \quad (13)$$

Remark 1. *The bonus can be computed efficiently by reducing the width computation to regression (Foster et al., 2018). We can additionally improve the computational efficiency using the sensitivity sampling technique developed in Wang et al. (2020), which significantly subsamples the dataset $\mathcal{Z}$. We omit the details for brevity. For neural networks, we provide a heuristic to approximate the bonus in Section 5.*

3.4. Algorithm Name Conventions

Since Algorithm 1 provides different options for subroutines, we specify different names for them as below.

- ENIAC-SPI-SAMPLE (ENIAC with sample-friendly SPI update): initialize with (3), collect data with (5), fit critic using (6), and update actor using (7);
- ENIAC-SPI-COMPUTE (ENIAC with computation-friendly SPI update): initialize with (4), collect data with (5), fit critic using (6), and update actor using (8);
- ENIAC-NPG-SAMPLE (ENIAC with sample-friendly NPG update): initialize with (3), collect data with (9), fit critic using (10), and update actor using (11);
- ENIAC-NPG-COMPUTE (ENIAC with computation-friendly NPG update): initialize with (4), collect data with (9), fit critic using (10), and update actor using a similar fashion as (8) modified from (11).

4. Theory

In this section, we provide convergence results of ENIAC with both the SPI and NPG options in the update rule. We

only present the main theorems and defer all proofs to the Appendix. We use superscript n for the n-th epoch in Algorithm 1 and the subscript t for the t-th iteration in Algorithm 2. For example, π_t^n is the output policy of the t-th iteration in the n-th epoch.

The sample complexities of our algorithms depend on the complexity of the function class for critic fit (and also the policy, implicitly). To measure the latter, we adopt the notion of eluder dimension which is first introduced in (Russo & Van Roy, 2013).

Definition 1 (Eluder Dimension). *Given a class $\mathcal{F}$, $\epsilon \geq 0$, and $\mathcal{Z} := \{(s_i, a_i)\}_{i=1}^n$ be a sequence of state-action pairs.*

- *A state-action pair (s, a) is ϵ-dependent on $\mathcal{Z}$ with respect to $\mathcal{F}$ if any $f, f' \in \mathcal{F}$ satisfying $\|f - f'\|_{\mathcal{Z}} := \sqrt{\sum_{(s',a')\in\mathcal{Z}}(f(s', a') - f'(s', a'))^2} \leq \epsilon$ also satisfy $|f(s, a) - f'(s, a)| \leq \epsilon$.*
- *An (s, a) is ϵ-independent of $\mathcal{Z}$ with respect to $\mathcal{F}$ if (s, a) is not ϵ-dependent on $\mathcal{Z}$.*
- *The ϵ-eluder dimension $dim_E(\mathcal{F}, \epsilon)$ of a function class $\mathcal{F}$ is the length of the longest sequence of elements in $\mathcal{S} \times \mathcal{A}$ such that, for some $\epsilon' \geq \epsilon$, every element is ϵ'-independent of its predecessors.*

It is well known (Russo & Van Roy, 2013) that if $f(z) = g(w^T z)$, where $z \in \mathbb{R}^d$, and g is a smooth and strongly monotone link function, then the eluder dimension of $\mathcal{F}$ is $O(d)$, where the additional constants depend on the properties of g. In particular, it is at most d for linear functions, and hence provides a strict generalization of results for linear function approximation. One can find more low eluder dimension examples from theories of invertible functions, non-linear regression, as well as roots of system of equations.

Based on this measure, we now present our main results for the SPI and NPG in the following subsections. For the sake of presentation, we provide the complexity bounds for ENIAC-SPI-SAMPLE and ENIAC-NPG-SAMPLE. The analysis for the rest of the algorithm options is similar and will be provided in the Appendix.

4.1. Main Results for ENIAC-SPI

At a high-level, there are two main sources of suboptimality. First is the error in the critic fitting, which further consists of both the *estimation error* due to fitting with finite samples, as well as an *approximation error* due to approximating the Q function from a restricted function class $\mathcal{F}$. Second, we have the suboptimality of the policy in solving the induced optimistic MDPs at each step. The latter is handled using standard arguments from the policy optimization literature (e.g. (Abbasi-Yadkori et al., 2019; Agarwal et al., 2020c)), while the former necessitates certain assumptions on the representability of the class $\mathcal{F}$. To this end, we begin with a closedness assumption on $\mathcal{F}$. For brevity, given a policy π we denote by

$$\mathcal{T}^\pi f(s, a) := \mathbb{E}^\pi[r(s, a) + \gamma f(s', a')|s, a]. \quad (14)$$

Assumption 4.1 ($\mathcal{F}$-closedness). *For all $\pi \in \{\mathcal{S} \to \Delta(\mathcal{A})\}$ and $g : \mathcal{S} \times \mathcal{A} \to [0, \frac{2}{(1-\gamma)^2}]$, we have $\mathcal{T}^\pi g \in \mathcal{F}$.*

Assumption 4.1 is a policy evaluation analog of a similar assumption in (Wang et al., 2020). For linear f, the assumption always holds if the MDP is a linear MDP (Jin et al., 2020) under the same features. We also impose regularity and finite cover assumptions on $\mathcal{F}$.

Assumption 4.2. $\max(1/(1-\gamma), \sup_{f\in\mathcal{F}} \|f\|_\infty) \leq W$.

Assumption 4.3 (ϵ-cover). *For any $\epsilon > 0$, there exists an ϵ-cover $\mathcal{C}(\mathcal{F}, \epsilon) \subseteq \mathcal{F}$ with size $|\mathcal{C}(\mathcal{F}, \epsilon)| \leq \mathcal{N}(\mathcal{F}, \epsilon)$ such that for any $f \in \mathcal{F}$, there exists $f' \in \mathcal{C}(\mathcal{F}, \epsilon)$ with $\|f - f'\|_\infty \leq \epsilon$.*

With the above assumptions, we have the following sample complexity result for ENIAC-SPI-SAMPLE.

Theorem 4.1 (Sample Complexity of ENIAC-SPI-SAMPLE). *Let $\delta \in (0, 1)$ and $\varepsilon \in (0, 1/(1-\gamma))$. Suppose Assumptions 4.1, 4.2, and 4.3 hold. With proper hyperparameters, ENIAC-SPI-SAMPLE returns a policy π satisfying $V^\pi \geq V^{\pi^\star} - \varepsilon$ with probability at least $1 - \delta$ after taking at most*

$$\tilde{\mathcal{O}}\Big(\frac{W^8 \cdot \big(dim_E(\mathcal{F}, \beta)\big)^2 \cdot \big(\log(\mathcal{N}(\mathcal{F}, \epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big)$$

samples, where $\beta = \varepsilon(1-\gamma)/2$ and $\epsilon' = \text{poly}(\varepsilon, \gamma, 1/W, 1/dim_E(\mathcal{F}, \beta))$[3].

One of the technical challenges of proving this theorem is to establish an eluder dimension upper bound on the sum of the error sequence. Unlike that in (Russo & Van Roy, 2013) and (Wang et al., 2020), who apply the eluder dimension argument directly to a sequence of data points, we prove a new bound that applies to the sum of expectations over a sequence of distributions. This bound is then carefully combined with the augmented MDP argument in (Agarwal et al., 2020a) to establish our exploration guarantee. The proof details are displayed in Appendix C. We now make a few remarks about the result.

Linear case. When $f(s, a) = u^T\phi(s, a)$ with $u, \phi(s, a) \in \mathbb{R}^d$, $\dim_E(\mathcal{F}, \beta) = O(d)$. Our result improves that of (Agarwal et al., 2020a) by using Bernstein concentration inequality to bound the generalization error. If Hoeffding inequality is used instead, our complexity will match that of (Agarwal et al., 2020a), thereby strictly generalizing their work to the non-linear setting.

[3] The formal definition of ϵ' can be found in Theorem C.1

Model misspecification Like the linear case, ENIAC-SPI (both SAMPLE and COMPUTE) is robust to the failure of Assumption 4.1. In Appendix C, we provide a *bounded transfer error* assumption, similar to that of Agarwal et al. (2020a), under which our guarantees hold up to an approximation error term. Informally, this condition demands that for any policy π_t, the best value function estimator f_t^* computed from *on-policy samples* also achieves a small approximation error for Q^{π_t} under the distribution $d_{s_0}^{\pi^*}$. A formal version is presented in the Appendix.

Comparison to value-based methods. Like the comparison between LSVI-UCB and PC-PG in the linear case, our results have a poorer scaling with problem and accuracy parameters than the related work of Wang et al. (2020). However, they are robust to a milder notion of model misspecification as stated above and readily lend themselves to practical implementations as our experiments demonstrate.

Sample complexity of ENIAC-SPI-COMPUTE. As remarked earlier, a key computational bottleneck in our approach is the need to compute the bonus while executing our policies. In Appendix C.2 we analyze ENIAC-SPI-COMPUTE, which avoids this overhead and admits a

$$\tilde{\mathcal{O}}\Big(\frac{W^{10}\cdot|\mathcal{A}|^2\cdot\big(\dim_E(\mathcal{F},\beta)\big)^2\cdot\big(\log(\mathcal{N}(\mathcal{F},\epsilon'))\big)^2}{\varepsilon^{10}(1-\gamma)^{10}}\Big)$$

sample complexity under the same assumptions. The worse sample complexity of ENIAC-SPI-COMPUTE arises from: 1. the uniform sampling over all actions instead of targeted randomization only over unknown actions for exploration; 2. α-probability uniform exploration even on known states.

4.2. Main Results for ENIAC-NPG

The results for ENIAC-NPG are qualitatively similar to those for ENIAC-SPI. However, there are differences in details as we fit the advantage function using the *tangent class* $\mathcal{G}_{\mathcal{F}}$ now, and this also necessitates some changes to the underlying assumptions regarding closure for Bellman operators and other regularity assumptions. We start with the former, and recall the definition of the tangent class $\mathcal{G}_{\mathcal{F}}$ in Equation (2). For a particular function $f\in\mathcal{F}$, we further use $\mathcal{G}_f\subseteq\mathcal{G}_{\mathcal{F}}$ to denote the subset of linear functions induced by the features $\nabla_\theta\log\pi_{f_\theta}$.

Assumption 4.4 ($\mathcal{G}_f$-closedness)**.** *For any $f\in\mathcal{F}$, let $\pi_f(a|s)\propto\exp(f(s,a))$. For any measurable set $\mathcal{K}\in\mathcal{S}\times\mathcal{A}$ and $g:\mathcal{S}\times\mathcal{A}\to[0,\frac{4}{(1-\gamma)^2}]$, we have $\mathcal{T}^{\pi_{f,\mathcal{K}}}g-\mathbb{E}_{\pi_{f,\mathcal{K}}}[\mathcal{T}^{\pi_{f,\mathcal{K}}}g]\in\mathcal{G}_f$, where $\pi_{f,\mathcal{K}}(\cdot|s)=\pi_f(\cdot|s)$ if for all $a\in\mathcal{A}$, $(s,a)\in\mathcal{K}$; otherwise, $\pi_{f,\mathcal{K}}(\cdot|s)=\mathrm{Unif}(\{a|(s,a)\notin\mathcal{K}\})$. The operator $\mathcal{T}$ is defined in Equation (14).*

One may notice that the policy $\pi_{f,\mathcal{K}}$ complies with our actor update in (11) since $b=0$ for $s\in\mathcal{K}$. We also impose regularity and finite cover assumptions on $\mathcal{G}_{\mathcal{F}}$ as below.

Assumption 4.5. *We assume that $\|u\|_2\le B$ for all $u\in\mathcal{U}\subset\mathbb{R}^d$, and f_θ is twice differentiable for all $f_\theta\in\mathcal{F}$, and further satisfies: $\|f_\theta\|_\infty\le W$, $\|\nabla f_\theta\|_2\le G$ and $\|\nabla^2 f_\theta\|_2\le\Lambda$. We denote by $D:=\max(BG,1/(1-\gamma))$.*

Assumption 4.6 (ϵ-cover)**.** *For the function class $\mathcal{G}_{\mathcal{F}}$, for any $\epsilon>0$, there exists an ϵ-cover $\mathcal{C}(\mathcal{G}_{\mathcal{F}},\epsilon)\subseteq\mathcal{G}_{\mathcal{F}}$ with size $|\mathcal{C}(\mathcal{G}_{\mathcal{F}},\epsilon)|\le\mathcal{N}(\mathcal{G}_{\mathcal{F}},\epsilon)$ such that for any $g\in\mathcal{G}_{\mathcal{F}}$, there exists $g'\in\mathcal{C}(\mathcal{G}_{\mathcal{F}},\epsilon)$ with $\|g-g'\|_\infty\le\epsilon$.*

We provide the sample complexity guarantee for ENIAC-NPG-SAMPLE as below.

Theorem 4.2 (Sample Complexity of ENIAC-NPG-SAMPLE)**.** *Let $\delta\in(0,1)$ and $\varepsilon\in(0,1/(1-\gamma))$. Suppose Assumptions 4.4, 4.5, and 4.6 hold. With proper hyperparameters, ENIAC-NPG-SAMPLE returns a policy π satisfying $V^\pi\ge V^{\pi^*}-\varepsilon$ with probability at least $1-\delta$ after taking at most*

$$\tilde{\mathcal{O}}\Big(\frac{D^6(D^2+\Lambda B^2)\big(dim_E(\mathcal{G}_{\mathcal{F}},\beta)\big)^2\big(\log(\mathcal{N}(\mathcal{G}_{\mathcal{F}},\epsilon'))\big)^2}{\varepsilon^8(1-\gamma)^8}\Big)$$

samples, where $\beta=\varepsilon(1-\gamma)/2$ and $\epsilon'=poly(\varepsilon,\gamma,1/D,1/dim_E(\mathcal{G}_{\mathcal{F}},\beta))$[4].

Notice that the differences between Theorems 4.1 and 4.2 only arise in the function class complexity terms and the regularity parameters, where the NPG version pays the complexity of the tangent class instead of the class $\mathcal{F}$ as in the SPI case. NPG, however, offers algorithmic benefits as remarked before, and the result here extends to a more general form under a bounded transfer error condition that we present in Appendix D. As with the algorithms, the theorems essentially coincide in the linear case. One interesting question for further investigation is the relationship between the eluder dimensions of the classes $\mathcal{F}$ and $\mathcal{G}_{\mathcal{F}}$, which might inform statistical preferences between the two approaches.

5. Experiments

We conduct experiments to testify the effectiveness of ENIAC. Specifically, we aim to show that

1. ENIAC is competent to solve RL exploration.
2. Compared with PC-PG which uses linear feature for bonus design, the idea of width in ENIAC performs better when using complex neural networks.

Check our code at https://github.com/FlorenceFeng/ENIAC.

[4]The formal definition of ϵ' can be found in Theorem D.1.

5.1. Implementation of ENIAC

We implement ENIAC using PPO (Schulman et al., 2017) as the policy update routine and use fully-connected neural networks (FCNN) to parameterize actors and critics. As for the width functions, recall that $w^n(s,a)$ is defined as:

$$\max_{f,f' \in \mathcal{F}} f(s,a) - f'(s,a), \text{ s.t. } \|f - f'\|_{\mathcal{Z}^n} \leq \epsilon. \quad (15)$$

To approximate this value in a more stable and efficient manner, we make several revisions to (15):

1. we fix f' and only train f;
2. due to 1., we change the objective from $f - f'$ to $(f - f')^2$ for symmetry;
3. we do not retrain f for every query point (s,a) but gather a batch of query points $\mathcal{Z}_Q^n$ and train in a finite-sum formulation.

We initialize f as a neural network with the same structure as the critic network (possibly different weights and biases) and initialize f' as a copy of f. Then we fix f' and train f by maximizing:

$$\sum_{(s,a)\in\mathcal{Z}_Q^n} \lambda\big(f(s,a) - f'(s,a)\big)^2/|\mathcal{Z}_Q^n| - \sum_{(s',a')\in\mathcal{Z}^n} \big(f(s',a') - f'(s',a')\big)^2/|\mathcal{Z}^n| - \sum_{(s,a)\in\mathcal{Z}_Q^n} \lambda_1\big(f(s,a) - f'(s,a)\big)/|\mathcal{Z}_Q^n|,$$

where the last term is added to avoid a zero gradient (since f and f' are identical initially). We generate $\mathcal{Z}_Q^n$ by using the current policy-cover as the initial distribution then rolling out with π^n. The training loss can be roughly regarded as a Lagrangian form of (15) with regularization. The intuition is that we want the functions to be close on frequently visited area (the second term) and to be as far as possible on the query part (the first term). After training for several steps of stochastic gradient descent, we freeze both f and f' and return $|f(s,a) - f'(s,a)|$ as $w^n(s,a)$. During the experiments, we set bonus as $0.5 \cdot \frac{w^n(s,a)}{\max_{\mathcal{Z}_Q^n} w^n}$ without thresholding. More details can be found in Appendix F.

We remark that in practice width training can be fairly flexible and customized for different environments. For example, one can design alternative loss functions as long as they follow the intuition; f and f' can be initialized differently; $\mathcal{Z}_Q^n$ can be generated with various distributions as long as it has a relatively wide coverage.

5.2. Environment and Baselines

We test on a continuous control task which requires exploration: continuous control MountainCar[5] from OpenAI Gym (Brockman et al., 2016). This environment has a 2-dimensional continuous state space and a 1-dimensional continuous action space $[-1, 1]$. The agent only receives a large reward $(+100)$ if it can reach the top of the hill and small negative rewards for any action. A locally optimal policy is to do nothing and avoid action costs. The length of horizon is 100 and $\gamma = 0.99$.

We compare five algorithms: ENIAC, vanilla PPO, PPO-RND, PC-PG, and ZERO. All algorithms use PPO as their policy update routine and the same FCNN for actors and critics. The vanilla PPO has no bonus; PPO-RND uses RND bonus (Burda et al., 2019) throughout training; PC-PG iteratively constructs policy cover and uses linear features (kernel-based) to compute bonus as in the implementation of Agarwal et al. (2020a), which we follow here; ZERO uses policy cover as in PC-PG and the bonus is all-zero. For ENIAC, PC-PG, and ZERO, instead of adding bonuses to rewards, we directly take the larger ones, i.e., the agent receives $\max(r, b)$ during exploration[6]. In ENIAC, we use uniform distribution to select policy from the cover set, i.e., $\rho_{\text{cov}}^n = \text{Unif}(d^{\pi^1}, \ldots, d^{\pi^n})$ as in the main algorithm; PC-PG optimizes the selection distribution based on the policy coverage (see (Agarwal et al., 2020a) for more details).

We evaluate all methods on varying depths of networks: 2-layer stands for (64, 64) hidden units, 4-layer for (64, 128, 128, 64), and 6-layer for (64, 64, 128, 128, 64, 64). Layers are connected with ReLU non-linearities. Hyperparameters for all methods are provided in Appendix F.

5.3. Results

In Figure 1, we see that ENIAC robustly achieves high performance consistently in all cases. Both PC-PG and ZERO perform well for depth 2, but as we increase the depth, the heuristic kernel-based bonus and the 0-offset bonus do not provide a good representation of the critic's uncertainty and its learning gets increasingly slower and unreliable. PPO and PPO-RND perform poorly, consistent with the results of Agarwal et al. (2020a). One can also regard the excess layers as masks on the true states and turn them into high-dimensional observations. When observations become increasingly complicated, more non-linearity is required for information processing and ENIAC is a more appealing choice.

We visualize ENIAC's policies in Figure 2, where we plot the state visitations of the exploration policies from the cover, as well as the exploitation policies trained using the cover with just the external reward. We see that ENIAC quickly attains exploration in the vicinity of the optimal state, allowing the exploitation policy to become optimal.

[5] https://gym.openai.com/envs/MountainCarContinuous-v0/

[6] This is simply for implementation convenience and does not change the algorithm. One can also adjust bonus as $\max(r, b) - r$.

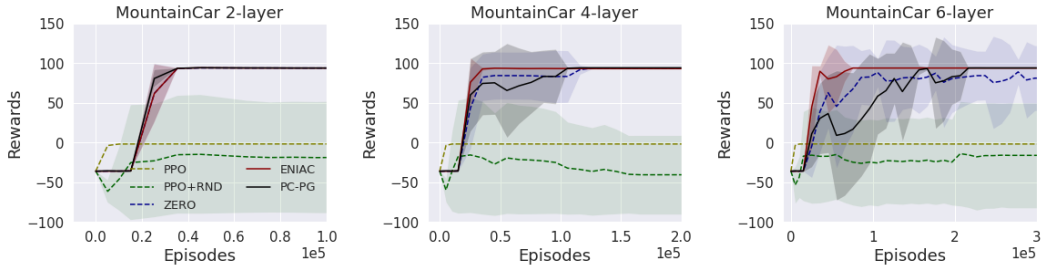

Figure 1. Performance of different methods on MountainCar as we vary the netural network depth. The performances are evaluated over 10 random seeds where lines are means and shades represent standard deviations. We stop training once the policy can obtain rewards$>$ 93.

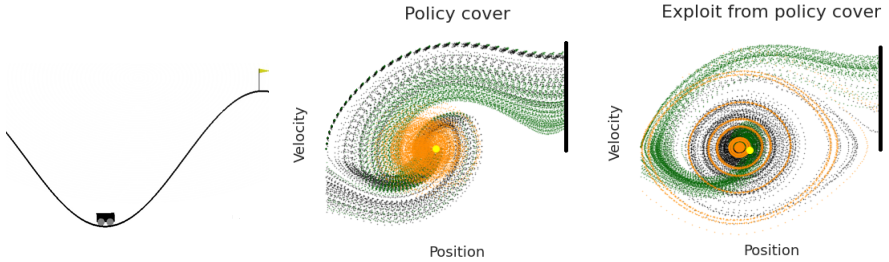

Figure 2. The MountainCar environment (left). Trajectories of exploration (middle) and exploitation (right) policies of ENIAC, with colors denoting different epochs: orange for the first policy in the cover set, black for the second, and green for the third. Agent starts from the centric area (near the yellow circle) and the black vertical line on the right represents goal positions.

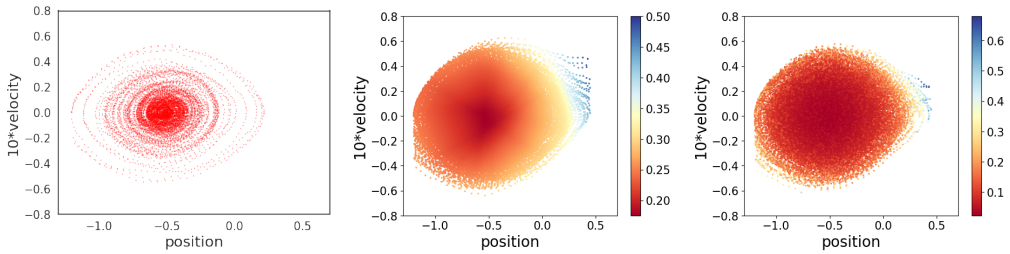

Figure 3. Bonus function comparison. [Left]: The trajectories of a chosen policy . [Middle]: the bonus function built by ENIAC upon the policy. [Right]: The bonus built by PC-PG upon the policy. See text for details.

Since the bonus in our experiments is smaller than the maximum reward, the effect of the bonus dissipates once we reach the optimal state. We also visualize typical landscapes of bonus functions in ENIAC and PC-PG in Figure 3. Both bonuses grant small values on frequently visited area and large values on scarsely visited part. But the bonus in ENIAC changes more smoothly. This might inspire future study on the shaping of bonuses.

6. Conclusion

In this paper, we present the first set of policy-based techniques for RL with non-linear function approximation. Our methods provide interesting tradeoffs between sample and computational complexities, while also inspire an extremely practical implementation. Empirically, our results demonstrate the benefit of correctly reasoning about the learner's uncertainty under a non-linear function class, while prior heuristics based on linear function approximation fail to robustly work as we vary the function class. Overall, our results open several interesting avenues of investigation for both theoretical and empirical progress. In theory, it is quite likely that our sample complexity results have scope for a significant improvement. A key challenge here is to enable better sample reuse, typically done with bootstrapping techniques for off-policy learning, while preserving the robustness to model misspecification that our theory exhibits. Empirically, it would be worthwhile to scale these methods to complex state and action spaces such as image-based inputs, and evaluate them on more challenging exploration tasks with a longer effective horizon.

Acknowledgements

We thank all reviewers for their constructive feedback. Lin Yang acknowledges the support from the Simons Institute for the Theory of Computing at UC Berkeley (Theory of Reinforcement Learning Program).

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 *International Conference on Machine Learning*, pp. 3692–3702. PMLR, 2019.

Agarwal, A., Henaff, M., Kakade, S., and Sun, W. PC-PG: Policy cover directed exploration for provable policy gradient learning. In *Advances in Neural Information Processing Systems*, 2020a.

Agarwal, A., Kakade, S., Krishnamurthy, A., and Sun, W. Flambe: Structural complexity and representation learning of low rank MDPs. In *Advances in Neural Information Processing Systems*, 2020b.

Agarwal, A., Kakade, S. M., Lee, J. D., and Mahajan, G. On the theory of policy gradient methods: Optimality, approximation, and distribution shift, 2020c.

Bhandari, J. and Russo, D. Global optimality guarantees for policy gradient methods. *CoRR*, abs/1906.01786, 2019. URL http://arxiv.org/abs/1906.01786.

Brafman, R. I. and Tennenholtz, M. R-max: A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct): 213–231, 2002.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Cai, Q., Yang, Z., Jin, C., and Wang, Z. Provably efficient exploration in policy optimization. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2020.

Dann, C., Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. On oracle-efficient PAC reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems 31*, 2018.

Du, S. S., Luo, Y., Wang, R., and Zhang, H. Provably efficient Q-learning with function approximation via distribution shift error checking oracle. In *Advances in Neural Information Processing Systems*, 2019.

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

Foster, D., Agarwal, A., Dudik, M., Luo, H., and Schapire, R. Practical contextual bandits with regression oracles. In *International Conference on Machine Learning*, pp. 1539–1548. PMLR, 2018.

Geist, M., Scherrer, B., and Pietquin, O. A theory of regularized Markov decision processes. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pp. 1861–1870. PMLR, 2018.

Jaksch, T., Ortner, R., and Auer, P. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(4), 2010.

Jiang, N., Krishnamurthy, A., Agarwal, A., Langford, J., and Schapire, R. E. Contextual decision processes with low Bellman rank are PAC-learnable. In *International Conference on Machine Learning*, 2017.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. Is Q-learning provably efficient? In *Advances in neural information processing systems*, pp. 4863–4873, 2018.

Jin, C., Yang, Z., Wang, Z., and Jordan, M. I. Provably efficient reinforcement learning with linear function approximation. In *Proceedings of Thirty Third Conference on Learning Theory*, Proceedings of Machine Learning Research, 2020.

Kakade, S. and Langford, J. Approximately optimal approximate reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, volume 2, pp. 267–274, 2002.

Kakade, S. M. A natural policy gradient. *Advances in neural information processing systems*, 14, 2001.

Kakade, S. M. *On the sample complexity of reinforcement learning*. PhD thesis, University of College London, 2003.

Kearns, M. and Singh, S. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Konda, V. R. and Tsitsiklis, J. N. Actor-critic algorithms. In *Advances in neural information processing systems*, pp. 1008–1014, 2000.

Misra, D., Henaff, M., Krishnamurthy, A., and Langford, J. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pp. 6961–6971. PMLR, 2020.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E., and Boots, B. Agile autonomous driving using end-to-end deep imitation learning. In *Robotics: science and systems*, 2018.

Peters, J. and Schaal, S. Natural Actor-Critic. *Neurocomput.*, 71(7-9):1180–1190, 2008. ISSN 0925-2312.

Russo, D. and Van Roy, B. Eluder dimension and the sample complexity of optimistic exploration. *Advances in Neural Information Processing Systems*, 26:2256–2264, 2013.

Scherrer, B. and Geist, M. Local policy search in a convex space and conservative policy iteration as boosted policy search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 35–50. Springer, 2014.

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.

Shangtong, Z. Modularized implementation of deep RL algorithms in pytorch, 2018.

Shani, L., Efroni, Y., Rosenberg, A., and Mannor, S. Optimistic policy optimization with bandit feedback. In *International Conference on Machine Learning*, pp. 8604–8613. PMLR, 2020.

Sun, W., Jiang, N., Krishnamurthy, A., Agarwal, A., and Langford, J. Model-based RL in contextual decision processes: Pac bounds and exponential improvements over model-free approaches. In *Conference on Learning Theory*, pp. 2898–2933. PMLR, 2019.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 99, pp. 1057–1063, 1999.

Wang, R., Salakhutdinov, R. R., and Yang, L. Reinforcement learning with general value function approximation: Provably efficient approach via bounded eluder dimension. *Advances in Neural Information Processing Systems*, 33, 2020.

Wen, Z. and Van Roy, B. Efficient exploration and value function generalization in deterministic systems. *Advances in Neural Information Processing Systems*, 26: 3021–3029, 2013.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Yang, L. and Wang, M. Reinforcement learning in feature space: Matrix bandit, kernels, and regret bound. In *International Conference on Machine Learning*, pp. 10746–10756. PMLR, 2020.