PC-PG: Policy Cover Directed Exploration for Provable Policy Gradient Learning

Alekh Agarwal
Microsoft Research, Redmond
alekha@microsoft.com

Sham Kakade
University of Washington and Microsoft Research NYC
sham@cs.washington.edu

Mikael Henaff
Facebook AI Research
mikaelhenaff@fb.com

Wen Sun
Cornell University
ws455@cornell.edu

Abstract

Direct policy gradient methods for reinforcement learning are a successful approach for a variety of reasons: they are model free, they directly optimize the performance metric of interest, and they allow for richly parameterized policies. Their primary drawback is that, by being local in nature, they fail to adequately explore the environment. In contrast, while model-based approaches and Q-learning can, at least in theory, directly handle exploration through the use of optimism, their ability to handle model misspecification and function approximation is far less evident. This work introduces the the POLICY COVER GUIDED POLICY GRADIENT (PC-PG) algorithm, which provably balances the exploration vs. exploitation tradeoff using an ensemble of learned policies (the policy cover). PC-PG enjoys polynomial sample complexity and run time for both tabular MDPs and, more generally, linear MDPs in an infinite dimensional RKHS. Furthermore, PC-PG also has strong guarantees under model misspecification that go beyond the standard worst case ℓ_∞ assumptions; these include approximation guarantees for state aggregation under an average case error assumption, along with guarantees under a more general assumption where the approximation error under distribution shift is controlled. We complement the theory with empirical evaluation across a variety of domains in both reward-free and reward-driven settings.

1 Introduction

Policy gradient methods are a successful class of Reinforcement Learning (RL) methods, as they are amenable to parametric policy classes, including neural policies [55, 56]), and they directly optimizing the cost function of interest. While these methods have a long history in the RL literature [33, 37, 62, 68], only recently have their theoretical convergence properties been established: roughly when the objective function has wide coverage over the state space, global convergence is possible [1, 3, 12, 27]. In other words, the assumptions in these works imply that the state space is already well-explored. Conversely, without such coverage (and, say, with sparse rewards), policy gradients often suffer from the vanishing gradient problem.

With regards to exploration, at least in the tabular setting, there is an established body of algorithms which provably explore in order to achieve sample efficient reinforcement learning, including model based methods [9, 13, 20, 29, 35, 36], model free approaches such as Q-learning [22, 31, 40, 60], Thompson sampling [5, 47, 51], and, more recently, policy optimization approaches [17, 24]. There are also a number of provable reinforcement learning algorithms balancing exploration and exploitation for MDPs with linearly parameterized dynamics, including [7, 17, 30, 32, 70–72].

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

The motivation for our work is to develop algorithms and guarantees which are more robust to violations in the underlying modeling assumptions. Indeed, the primary practical motivation for policy gradient methods is that the overall methodology is disentangled from modeling (and Markovian) assumptions, since they are an "end-to-end" approach, directly optimizing the cost function of interest. These practical considerations are supported by a body of theoretical results, both on direct policy optimization approaches [10, 34, 53, 54] and more recently on policy gradient approaches [3], which show that such incremental policy improvement approaches are amenable to function approximation and violations of modeling assumptions, under certain coverage assumptions over the state space.

This work focuses on how policy gradient methods can be extended to handle exploration, while also retaining their favorable properties with regards to how they handle function approximation and model misspecification. The practical relevance of answering these questions is evident by the growing body of empirical techniques for exploration which combine policy gradient methods with exploration bonuses such as pseudocounts [11], dynamics model errors [49], or random network distillation (RND) [16].

Our Contributions. This work introduces the POLICY COVER GUIDED POLICY GRADIENT algorithm (PC-PG), a direct, model-free, policy optimization approach which addresses exploration through the use of a learned ensemble of policies, the latter provides a policy cover over the state space. The use of a learned policy cover addresses exploration, and also addresses what is known as the "catastrophic forgetting" problem in policy gradient approaches (which use reward bonuses); while the on-policy nature avoids the "delusional bias" inherent to Bellman backup-based approaches, where approximation errors due to model misspecification amplify (see [42] for discussion).

It is a conceptually different approach from the predominant prior (and provable) RL algorithms, which are either model-based — variants of UCB [9, 13, 29, 36] or based on Thompson sampling [5, 51] — or model-free and value based, such as Q-learning [31, 60]. Our work adds policy optimization methods to this list, as a direct alternative: the use of learned covers permits a *a model-free approach* by allowing the algorithm to plan in the real world, using the cover for initializing the underlying policy optimizer. We remark that only a handful of prior (provable) exploration algorithms [31, 60] are model-free in the tabular setting, and these are largely value based.

At a high-level, we design an exploratory actor-critic method which performs critic estimation using linear functions in a given featurization $\phi(s, a)$ for a state s and action a. Informally, our main result shows that for any policy π, we can compete with $V^{\pi}(s_0)$ in the initial state s_0, as long as we can estimate the critic functions for all actor policies discovered by the algorithm up to a small mean squared error, where the error is measured under the state-action distribution induced by the comparison policy π. While this distribution is unavailable to the algorithm, and the error cannot be explicitly reduced to zero by collecting more samples for fitting the critic, the assumption provides an expressivity condition on the features ϕ. The condition is directly inspired by the recent characterization of *transfer error under distribution shift* as a key quantity underlying the convergence of policy optimization methods with function approximation by Agarwal et al. [3], and our results extend these ideas to the exploration setting.

This abstract result provides a string of striking corollaries for linear MDPs [32], state aggregation, and other settings, both when the modeling assumptions hold exactly and approximately. In the exact case, PC-PG is provably sample and computationally efficient for *both tabular and linear MDPs*. In the approximate case, the key upshot of our theory is to permit the use of certain average case error measures instead of ℓ_∞ conditions on misspecification in several prior works. Some of the key results that we establish include:

- **RKHS in Linear MDPs**: For the linear MDPs proposed by [32], our results hold when the linear MDP features live in an infinite dimensional Reproducing Kernel Hilbert Space (RKHS) (Theorem 4.1). It is not immediately evident how to extend the prior work on linear MDPs (e.g. [32]) to this setting (due to concentration issues with data re-use).
- **Bounded transfer error and state aggregation**: When specialized to a state aggregation setting, we show that PC-PG provides a different approximation guarantee in comparison to prior works. In particular, the aggregation need only be good locally, under the visitations of the comparison policy (Theorem 4.2). More generally, we analyze PC-PG under a notion of a small transfer error in critic fitting [3]—a condition on the error of a best on-policy critic under a comparison policy's

state distribution—which generalizes the special case of state aggregation, and show that PC-PG enjoys a favorable sample complexity whenever this transfer error is small (Theorem 4.3).

- **Empirical evaluation**: We provide experiments showing the viability of PC-PG in settings where prior bonus based approaches such as Random Network Distillation [16] do not recover optimal policies with high probability. Our experiments show our basic approach complements and leverages existing deep learning approaches, implicitly also verifying the robustness of PC-PG outside the regime where the sample complexity bounds provably hold.

2 Setting

A Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma, s_0)$ is specified by a state space $\mathcal{S}$; an action space $\mathcal{A}$; a transition model $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ (where $\Delta(\mathcal{S})$ denotes a distribution over states), a reward function $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$, a discount factor $\gamma \in [0, 1)$, and a starting state s_0. We assume $\mathcal{A}$ is discrete and denote $A = |\mathcal{A}|$. Our results generalize to a starting state distribution $\mu_0 \in \Delta(\mathcal{S})$ but we use a single starting state s_0 to emphasize the need to perform exploration. A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ specifies a decision-making strategy in which the agent chooses actions based on the current state, i.e., $a \sim \pi(\cdot|s)$.

The value function $V^\pi(\cdot, r) : \mathcal{S} \to \mathbb{R}$ is defined as the expected discounted sum of future rewards, under reward function r, starting at state s and executing π, i.e. $V^\pi(s; r) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|\pi, s_0 = s\right]$, where the expectation is taken with respect to the randomness of the policy and environment $\mathcal{M}$. The *state-action* value function $Q^\pi(\cdot, \cdot; r) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as $Q^\pi(s, a; r) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|\pi, s_0 = s, a_0 = a\right]$.

We define the discounted state-action distribution d_s^π of a policy π: $d_{s'}^\pi(s, a) := (1 - \gamma)\sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s, a_t = a|s_0 = s')$, where $\mathrm{Pr}^\pi(s_t = s, a_t = a|s_0 = s')$ is the probability that $s_t = s$ and $a_t = a$, after we execute π from $t = 0$ onwards starting at state s' in model $\mathcal{M}$. Similarly, we define $d_{s',a'}^\pi(s, a)$ as: $d_{s',a'}^\pi(s, a) := (1 - \gamma)\sum_{t=0}^{\infty} \gamma^t \mathrm{Pr}^\pi(s_t = s, a_t = s|s_0 = s', a_0 = a')$. For any state-action distribution ν, we write $d_\nu^\pi(s, a) := \sum_{(s',a') \in \mathcal{S} \times \mathcal{A}} \nu(s', a') d_{s',a'}^\pi(s, a)$. For ease of presentation, we assume that the agent can reset to s_0 at any point in the trajectory.[1] We denote $d_\nu^\pi(s) = \sum_a d_\nu^\pi(s, a)$. The goal of the agent is to find a policy π that maximizes the expected value from the starting state s_0, i.e. the optimization problem is: $\max_\pi V^\pi(s_0)$, where the $\max$ is over some policy class. For completeness, we specify a d_ν^π-sampler and an unbiased estimator of $Q^\pi(s, a; r)$ in Algorithm 1, which are standard in discounted MDPs. The d_ν^π sampler samples (s, a) i.i.d from d_ν^π, and the Q^π sampler returns an unbiased estimate of $Q^\pi(s, a; r)$ for a given triple (s, a, r) by a single roll-out from (s, a).

We assume access to a feature map $\phi(s, a)$ for each $s \in \mathcal{S}$ and $a \in \mathcal{A}$, with $\|\phi(s, a)\| \leq 1$, and $\|\cdot\|$ is a norm in some Reproducing Kernel Hilbert Space (RKHS). The features are primarily used for fitting the critic functions Q^π in our algorithm, though some of our theoretical results also make stronger assumptions on the MDP dynamics in terms of ϕ (e.g., linear MDP).

Notation. When clear from context, we write $d^\pi(s, a)$ and $d^\pi(s)$ to denote $d_{s_0}^\pi(s, a)$ and $d_{s_0}^\pi(s)$ respectively, where s_0 is the starting state in our MDP. For iterative algorithms which obtain policies at each episode, we let V^n,Q^n, A^n and d^n denote the corresponding quantities associated with episode n. For a vector v, we denote $\|v\|_2 = \sqrt{\sum_i v_i^2}$, $\|v\|_1 = \sum_i |v_i|$, and $\|v\|_\infty = \max_i |v_i|$. For a matrix V, we define $\|V\|_2 = \sup_{x:\|x\|_2 \leq 1} \|Vx\|_2$, and $\det(V)$ as the determinant of V. We use $\mathrm{Uniform}(\mathcal{A})$ (in short $\mathrm{Unif}_\mathcal{A}$) to represent a uniform distribution over the set $\mathcal{A}$.

3 The PC-PG Algorithm

To motivate the algorithm, first consider the original objective function: $\max_{\pi \in \Pi} V^\pi(s_0; r)$, where r is the true reward function. Simply doing policy gradient ascent on this objective function may easily lead to poor stationary points due to lack of coverage (i.e. lack of exploration). In such cases, a more desirable objective function is of the form:

$$\text{A wide coverage objective:} \quad \max_{\pi \in \Pi} \mathbb{E}_{s_0, a_0 \sim \rho_{\text{cov}}} \left[Q^\pi(s_0, a_0; r)\right] \tag{1}$$

[1] This can be replaced with a termination at each step with probability $1 - \gamma$.

Algorithm 1 d^π sampler and Q^π estimator

```
function $d^\pi_\nu$-SAMPLER
  Input: $\nu \in \Delta(\mathcal{S} \times \mathcal{A}), \pi, r(s,a)$
   Sample $s_0, a_0 \sim \nu$
   Execute $\pi$ from $s_0, a_0$; at any step $t$ with $(s_t, a_t)$, terminate the episode with probability $1-\gamma$
  Return: $s_t, a_t$
end function
function $Q^\pi$-ESTIMATOR
  Input: current state-action $(s,a)$, reward $r(s,a)$, $\pi$
   Execute $\pi$ from $(s_0, a_0) = (s,a)$; at step $t$ with $(s_t, a_t)$, terminate with probability $1-\gamma$
  Return: $\widehat{Q}^\pi(s,a) = \sum_{i=0}^{t} r(s_i, a_i)$ where $(s_0, a_0) = (s,a)$
end function
```

Algorithm 2 Policy Cover guided Policy Gradient (PC-PG)

```
1: Input: iterations $N$, threshold $\beta$, regularizer $\lambda$
2: Initialize $\pi^0(a|s)$ to be uniform
3: for episode $n = 0, \dots N-1$ do
4:    Estimate the covariance of $\pi^n$ as $\widehat{\Sigma}^n = \sum_{i=1}^{K} \phi(s_i,a_i)\phi(s_i,a_i)^\top / K$ with $\{s_i, a_i\}_{i=1}^K \sim d^n$
5:    Estimate the covariance of the policy cover as $\widehat{\Sigma}^n_{\text{cov}} := \sum_{i=0}^{n} \widehat{\Sigma}^i + \lambda I$
6:    Set the exploration bonus $b^n$ to reward infrequently visited state-action under $\rho^n_{\text{cov}}$ (3)

          $b^n(s,a) = \frac{\mathbf{1}\{(s,a)\ :\ \phi(s,a)^\top(\widehat{\Sigma}^n_{\text{cov}})^{-1}\phi(s,a) \ge \beta\}}{1-\gamma}. \quad (2)$

7:    Update $\pi^{n+1} = \text{NPG-Update}(\rho^n_{\text{cov}}, b^n)$ (Algorithm 3)
8: end for
```

where ρ_{cov} is some initial state-action distribution which has wider coverage over the state space. As argued in [3, 34, 53, 54], wide coverage initial distributions ρ_{cov} are critical to the success of policy optimization methods. However, in the RL setting, our agent can only start from s_0. Concretely, for the linear function approximation in terms of a given feature map ϕ that we employ here, from Theorem 6.1 of Agarwal et al. [3] (see also Abbasi-Yadkori et al. [1, 2]), we know that the coverage of ρ_{cov} can be measured in terms of the smallest eigenvalue of its feature covariance matrix,[2] an object which plays a central role in our algorithm design.

The idea of our iterative algorithm, PC-PG (Algorithm 2), is to successively improve *both* the current policy π and the coverage distribution ρ_{cov}. At episode n, we have $n+1$ previous policies $\pi^0, \dots \pi^n$. Each of these policies π^i induces a distribution $d^i := d^{\pi^i}$ over the state-action space. Define ρ^n_{cov} as:

$$\rho^n_{\text{cov}}(s,a) = \sum_{i=0}^{n} d^i(s,a)/(n+1) \tag{3}$$

Intuitively, ρ^n_{cov} reflects the coverage the algorithm has over the state-action space at the start of the n-th episode. PC-PG then uses ρ^n_{cov} in the previous objective (1) with two modifications: PC-PG modifies the instantaneous reward function r with a bonus b^n in order to search for a policy π^{n+1} which covers a novel part of space. It also modifies the policy class from Π to Π_{bonus}, where all policies $\pi \in \Pi_{\text{bonus}}$ are constrained to simply take a random rewarding action for those states where the bonus is already large (see Eq 4 in Alg. 3). With this, PC-PG's objective at the n-th episode is: $\max_{\pi \in \Pi_{\text{bonus}}} \mathbb{E}_{s_0, a_0 \sim \rho^n_{\text{cov}}}\left[Q^\pi(s_0, a_0; r + b^n)\right]$. The idea is that PC-PG can effectively optimize over the region where ρ^n_{cov} has coverage. Furthermore, by construction of the bonus, the algorithm is encouraged to escape the current region of coverage.

Reward bonus construction. At each episode n, PC-PG maintains an estimate of feature covariance of the policy cover ρ^n_{cov} (Line 5 of Algorithm 2). Next we use this covariance matrix to identify

[2] For infinite dimensions, the more general notion of relative condition number is required, but minimum eigenvalue provides reasonable intuition and is sufficient in the finite dimensional case.

Algorithm 3 Natural Policy Gradient (NPG) Update

```
1: Input $\rho^n_{\text{cov}}$, $b^n$, learning rate $\eta$, sample size $M$ for critic fitting, iterations $T$
2: Define $\mathcal{K}^n = \{s : \forall a \in \mathcal{A}, b^n(s,a) = 0\}$
3: Initialize policy $\pi^0 : \mathcal{S} \to \Delta(\mathcal{A})$, such that
       $\pi^0(\cdot|s) = \begin{cases} \text{Uniform}(\mathcal{A}) & s \in \mathcal{K}^n \\ \text{Uniform}(\{a \in \mathcal{A} : b^n(s,a) > 0\}) & s \notin \mathcal{K}^n. \end{cases}$
4: for $t = 0 \to T-1$ do
5:     Draw $M$ i.i.d samples $\left\{ s_i, a_i, \widehat{Q}^{\pi^t}(s_i, a_i; r + b^n) \right\}_{i=1}^{M}$ with $s_i, a_i \sim \rho^n_{\text{cov}}$ (see Alg 1)
6:     Critic fit:
           $\theta^t = \operatorname*{argmin}_{\|\theta\| \le W} \sum_{i=1}^{M} \left( \theta \cdot \phi(s_i, a_i) - \left( \widehat{Q}^{\pi^t}(s_i, a_i; r + b^n) - b^n(s_i, a_i) \right) \right)^2$
7:     Actor update
           $\pi^{t+1}(\cdot|s) \propto \pi^t(\cdot|s) \exp\left( \eta \left( b^n(s,\cdot) + \theta^t \cdot \phi(s,\cdot) \right) \mathbf{1}\{s \in \mathcal{K}^n\} \right)$    (4)
8: end for
9: return $\pi := \text{argmax}_{\pi \in \{\pi^0, \ldots, \pi^{T-1}\}} V^\pi(s_0; r + b^n)$
```

state-action pairs which are adequately covered by ρ^n_{cov}, inspired by prior policy optimization results for linear function approximation as mentioned before. The goal of the reward bonus is to identify state, action pairs whose features are less explored by ρ^n_{cov} and incentivize visiting them. The bonus $b^n(s,a)$ defined in Eq 2 achieves this. If $\widehat{\Sigma}^n_{\text{cov}}$ has a small eigenvalue along $\phi(s,a)$, then we assign the largest possible reward-to-go (i.e., $1/(1-\gamma)$) for this (s,a) pair to encourage exploration.[3] Note that reward bonus is only explicitly computed on states along the trajectory roll-outs during the execution of the algorithm (see Algorithm 3).

Policy Optimization. With the bonus, we update the policy via T steps of natural policy gradient (Algorithm 3). In the NPG update, we first approximate the value function $Q^{\pi^t}(s,a; r+b^n)$ under the policy cover ρ^n_{cov} (line 6). Specifically, we use linear function approximator to approximate $Q^{\pi^t}(s,a; r+b^n) - b^n(s,a)$ via constrained linear regression (line 6), and then approximate $Q^{\pi^t}(s,a; r+b^n)$ by adding bonus back: $\overline{Q}^t_{b^n}(s,a) := b^n(s,a) + \theta^t \cdot \phi(s,a)$, Note that the error of $\overline{Q}^t_{b^n}(s,a)$ to $Q^{\pi^t}(s,a; r+b^n)$ is simply the prediction error of $\theta^t \cdot \phi(s,a)$ to the regression target $Q^{\pi^t}(s,a; r+b^n) - b^n$. The purpose of structuring the value function estimation this way, instead of directly approximating $Q^t(s,a; r+b^n)$ with a linear function, for instance, is that the regression problem defined in line 6 will have a good linear solution for the special case linear MDPs, while we cannot guarantee the same for $Q^t(s,a; r+b^n)$ due to the non-linearity of the bonus. We then use the critic $\overline{Q}^t_{b^n}$ for updating policy (Eq. (4)). These are the exponential gradient updates (as in [3, 33]), but are constrained for $s \in \mathcal{K}^n$ (see line 2 for the definition of $\mathcal{K}^n$). The initialization and the update ensure that π^t chooses actions uniformly from $\{a : b^n(s,a) > 0\} \subseteq \mathcal{A}$ at any state s with $|\{a : b^n(s,a) > 0\}| > 0$ (the policy is restricted to act uniformly among positive bonus actions). Our policy update ensures that for any state s, π^{t+1} is not that different from π^t. Similar to CPI [34] and PSDP [10], such incremental policy update is the key to the robustness in model-misspecification that goes beyond the worst case ℓ_∞ assumption, which we discuss in detail in Sec. 4

4 Theory

We first state sample complexity results for linear MDPs. We then demonstrate the robustness of PC-PG to model misspecification. We show that in state aggregation, error incurred is only an average model error from aggregation averaged over the fixed comparator's abstracted state distribution.

[3] For an infinite dimensional RKHS, the bonus can be computed in the dual using the kernel trick (e.g., [66]).

For more general agnostic setting we show that our algorithm is robust to model-misspecification which is measured in a new concept of *transfer error* introduced by [3] recently. In Appendix G, we further provide model-misspecification examples and show why most prior approaches fail due to the delusional bias of Bellman backups under function approximation and model misspecification [42].

4.1 Well specified case: Linear MDPs in RKHS

We directly work on linear MDPs in a general Reproducing Kernel Hilbert space (RKHS).

Definition 4.1 (Linear MDP in RKHS). *Let $\mathcal{H}$ be an RKHS, and define a feature mapping $\phi : \mathcal{S} \times \mathcal{A} \to \mathcal{H}$. An MDP $(\mathcal{S}, \mathcal{A}, P, r, \gamma, s_0)$ is called a linear MDP if the reward function lives in $\mathcal{H}$: $r(s, a) = \langle \theta, \phi(s, a)\rangle_{\mathcal{H}}$, and the transition operator $P(s'|s, a)$ also lives in $\mathcal{H}$: $P(s'|s, a) = \langle \mu(s'), \phi(s, a)\rangle_{\mathcal{H}}$ for all (s, a, s'). Denote μ as a matrix whose each row corresponds to $\mu(s)$. We assume the parameter norms*[4] *are bounded as $\|\theta\| \leq \omega$, $\|v^\top \mu\| \leq \xi$ for all $v \in \mathbb{R}^{|\mathcal{S}|}$ with $\|v\|_\infty \leq 1$, $\sup_{s,a} \|\phi(s, a)\| \leq 1$. we assume that the initial state s_0 has lower bounded non-zero norm $\min_{a\in\mathcal{A}} \|\phi(s_0, a)\| \geq c_0 \in (0, 1]$ (other states could have arbitrarily small norms).*

As our feature vector ϕ could be infinite dimensional, to measure the sample complexity, we define the *maximum information gain* of the underlying MDP $\mathcal{M}$. First, denote the covariance matrix of any policy π as $\Sigma^\pi = \mathbb{E}_{(s,a)\sim d^\pi}\left[\phi(s, a)\phi(s, a)^\top\right]$.We define the maximum information gain below:

Definition 4.2 (Maximum Information Gain $\mathcal{I}_N(\lambda)$). *We define the maximum information gain as: $\mathcal{I}_N(\lambda) := \max_{\{\pi^i\}_{i=0}^{N-1}} \log\det\left(\frac{1}{\lambda}\sum_{i=0}^{N-1}\Sigma^{\pi^i} + I\right)$, where $\lambda \in \mathbb{R}^+$.*

Remark 4.1. This quantity is identical to the maximum information gain in Gaussian Process bandits [59] from a Bayesian perspective. A related quantity occurs in a more restricted linear MDP model [69]. Note that when $\phi(s, a) \in \mathbb{R}^d$, we have that $\log\det\left(\sum_{i=1}^n \Sigma^{\pi^i}/\lambda + I\right) \leq d\log(nB^2/\lambda + 1)$ assuming $\|\phi(s, a)\|_2 \leq B$, which means that the information gain is always at most $\widetilde{O}(d)$.

Theorem 4.1 (Sample Complexity of PC-PG for Linear MDPs). *Fix $\epsilon, \delta \in (0, 1)$ and an arbitrary comparator policy $\pi^\star$ (not necessarily an optimal policy). Suppose that $\mathcal{M}$ is a linear MDP (4.1). There exists a setting of the parameters such that PC-PG uses a number of samples at most poly $\left(\frac{1}{1-\gamma}, \log(A), \frac{1}{\epsilon}, \mathcal{I}_N(1), \omega, \xi, \ln\left(\frac{1}{\delta}\right)\right)$ and, with probability greater than $1 - \delta$, returns a policy $\widehat{\pi}$ such that $V^{\widehat{\pi}}(s_0) \geq V^{\pi^\star}(s_0) - \epsilon$.*

A few remarks are in order:

Remark 4.2. For tabular MDPs, as ϕ is a $|\mathcal{S}||\mathcal{A}|$ indictor vector, the theorem above immediately extends to tabular MDPs with $\mathcal{I}_N(1)$ being replaced by $|\mathcal{S}|\mathcal{A}|\log(N + 1)$.

Remark 4.3. In contrast with LSVI-UCB [32], PC-PG works for infinite dimensional ϕ with a polynomial dependency on the maximum information gain $\mathcal{I}_N(1)$.

Instead of proving Theorem 4.1 directly, we will state and prove a general theorem of PC-PG for general MDPs with model-misspecification measured in a new concept *transfer error* (Assumption 4.1) introduced by [3] in Section 4.3. Theorem 4.1 can be understood as a corollary of a more general agnostic theorem (Theorem 4.3). Detailed proof of Theorem 4.1 is included in Appendix E.

4.2 State-Aggregation under Model Misspecification

Consider a simple model-misspecified setting where the model error is introduced due to state action aggregation. Suppose we have an aggregation function $\phi : \mathcal{S}\times\mathcal{A} \to \mathcal{Z}$, where $\mathcal{Z}$ is a finite categorical set, the "state abstractions", which we typically think of as being much smaller than the (possibly infinite) number of state-action pairs. Intuitively, we aggregate state-action pairs that have similar transitions and rewards to an abstracted state z. This aggregation introduces model-misspecification.

Definition 4.3. *We define model-misspecification ϵ_z for any $z \in \mathcal{Z}$ as*

$$\epsilon_z := \max_{(s,a),(s',a') \text{ s.t. } \phi(s,a)=\phi(s',a')=z} \left\{ \left\|P(\cdot|s, a) - P(\cdot|s', a')\right\|_1, |r(s, a) - r(s', a')| \right\}.$$

[4] The norms are induced by the inner product in the Hilbert space $\mathcal{H}$, unless stated otherwise.

The model-misspecification measures the maximum possible disagreement in terms of transition and rewards of two state-action pairs which are mapped to the same abstracted state.

The folklore result is that with the definition $\|\epsilon_{misspec}\|_\infty := \max_{z\in\mathcal{Z}} \epsilon_z$, algorithms such as UCB and Q-learning, and regular policy gradient [52] succeed with an additional additive error of $\|\epsilon_{misspec}\|_\infty/(1-\gamma)^2$, and will have sample complexity guarantees that are polynomial in only $|\mathcal{Z}|$. Interestingly, see [22, 40, 52] for conditions limited to only $Q^\star$ are still *global* in nature. The following theorem shows PC-PG only requires a more local guarantee where our aggregation needs to be only good under the distribution of abstracted states where an optimal policy tends to visit.

Theorem 4.2 (Misspecified, State-Aggregation Bound)**.** *Fix* $\epsilon, \delta \in (0,1)$. *Let* $\pi^\star$ *be an arbitrary comparator policy. There exists a setting of the parameters such that PC-PG (Algorithm 2) uses a total number of samples at most poly* $\left(|\mathcal{Z}|, \log(A), \frac{1}{1-\gamma}, \frac{1}{\epsilon}, \ln\left(\frac{1}{\delta}\right)\right)$ *and, with probability greater than* $1-\delta$*, returns a policy* $\widehat{\pi}$ *such that,* $V^{\widehat{\pi}}(s_0) \geq V^{\pi^\star}(s_0) - \epsilon - \frac{2\mathbb{E}_{s\sim d^{\pi^\star}} \max_a\left[\epsilon_{\phi(s,a)}\right]}{(1-\gamma)^3}$.

Here, it could be that $\mathbb{E}_{s\sim d^{\pi^\star}} \max_a[\epsilon_{\phi(s,a)}] \ll \|\epsilon_{misspec}\|_\infty$ due to that our error notion is an average case one under the comparator. We refer readers to Appendix F for detailed proof of the above theorem which can also be regarded as a corollary of a more general agnostic theorem (Theorem 4.3) that we present in the next section. Note that here we pay an additional $1/(1-\gamma)$ factor in the approximation error due to the fact that after reward bonus, we have $r(s,a)+b^n(s,a) \in [0, 1/(1-\gamma)]$. [5] One point worth reflecting on is how few guarantees there are in the more general RL setting (beyond dynamic programming), which address model-misspecification beyond global ℓ_∞ bounds. Our conjecture is that this is not merely an analysis issue but an algorithmic one, where incremental algorithms such as PC-PG are required for strong misspecified algorithmic guarantees. We return to this point in Appendix G, with an example showing why this might be the case.

4.3 Agnostic Guarantees with Bounded Transfer Error

We now consider a general MDP in this section, where we do not assume the linear MDP modeling assumptions hold. As $Q-b^n$ may not be linear with respect to the given feature ϕ, we need to consider model misspecification due to the linear function approximation with features ϕ. We use the new concept of transfer error from [3] below. We use the shorthand notation: $Q^t_{b^n}(s,a) = Q^{\pi^t}(s,a;r+b^n)$ below. We capture model misspecification using the following assumption.

Assumption 4.1 (Bounded Transfer Error)**.** *With respect to a target function* $f : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, *define the critic loss function* $L(\theta; d, f)$ *with* $d \in \Delta(\mathcal{S}\times\mathcal{A})$ *as:* $L(\theta; d, f) := \mathbb{E}_{(s,a)\sim d}\left(\theta\cdot\phi(s,a) - f\right)^2$, *which is the square loss of using the critic* $\theta\cdot\phi$ *to predict a given target function* f*, under distribution* d*. Consider an arbitrary comparator policy* $\pi^\star$ *(not necessarily an optimal policy) and denote the state-action distribution* $d^\star(s,a) := d^{\pi^\star}(s) \circ \mathit{Unif}_{\mathcal{A}}(a)$. *For all episode* n *and all iteration* t *inside episode* n*, define:* $\theta^t_\star \in \operatorname{argmin}_{\|\theta\|\leq W} L\left(\theta; \rho^n_{cov}, Q^t_{b^n} - b^n\right)$. *Then we assume that (when running Algorithm 2),* $\theta^t_\star$ *has a bounded prediction error when transferred to* $d^\star$ *from* ρ^n_{cov}*; more formally:*

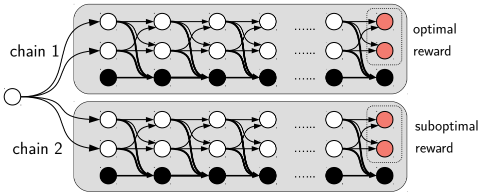
$$L\left(\theta^t_\star; d^\star, Q^t_{b^n} - b^n\right) \leq \epsilon_{bias} \in \mathbb{R}^+.$$

Note that the transfer error ϵ_{bias} measures the prediction error, at episode n and iteration t, of a best on-policy fit $\overline{Q}^t_{b^n}(s,a) := b^n(s,a) + \theta^t_\star\cdot\phi(s,a)$ measured under a fixed distribution $d^\star$ from the fixed comparator (note $d^\star$ is different from the training distribution ρ^n_{cov} hence the name *transfer*).

This assumption first appears in the recent work of Agarwal et al. [3] in order to analyze NPG under linear function approximation. This is a milder notion of model misspecification than ℓ_∞-variants more prevalent in the literature, as it is an average-case quantity which can be significantly smaller in favorable cases. With the above assumption on the transfer error, the next theorem states an agnostic result for the sample complexity of PC-PG:

Theorem 4.3 (Agnostic Guarantee of PC-PG)**.** *Fix* $\epsilon, \delta \in (0,1)$ *and consider an arbitrary comparator policy* $\pi^\star$ *(not necessarily an optimal policy). Assume Assumption 4.1 holds, and* $\sup_{s,a}\|\phi(s,a)\| \leq 1$, *and* $\sup_a \|\phi(s_0), a\| \geq c_0 \in (0,1]$. *There exists a setting of the parameters* $(\beta, \lambda, K, M, \eta, N, T)$

[5] We note that instead of using reward bonus, we could construct absorbing MDPs to make rewards scale $[0,1]$. This way we will pay $1/(1-\gamma)^2$ in the approximation error instead.



Algorithm	Horizon			
	2	5	10	15
PPO	1.0	0.0	0.0	0.0
PPO+RND	0.75	0.40	0.50	0.55
PC-PG	1.0	1.0	1.0	1.0

Figure 1: **Left** panel shows the Bidirectional Diabolical Combination Lock domain (see text for details). **Right** panel shows success rate of different algorithms averaged over 20 different seeds.

such that PC-PG uses a number of samples at most poly $\left(\frac{1}{1-\gamma}, \log(A), \frac{1}{\epsilon}, \mathcal{I}_N(1), W, \ln\left(\frac{1}{\delta}\right)\right)$ *and, with probability greater than* $1-\delta$*, returns a policy* $\widehat{\pi}$ *such that* $V^{\widehat{\pi}}(s_0) \geq V^{\pi^\star}(s_0) - \epsilon - \frac{\sqrt{2A\epsilon_{bias}}}{1-\gamma}$.

The precise polynomial of the sample complexity, along with the settings of all the hyperparameters — β (threshold for bonus), λ, K (samples for estimating cover's covariance), M (samples for fitting critic), η (learning rate in NPG), N (number of episodes), and T (number of NPG iterations per episode) — is provided in Theorem D.1 (Appendix D), where we discuss two examples of ϕ—finite dimensional $\phi \in \mathbb{R}^d$, and infinite dimensional ϕ in RKHS with RBF kernel (Remark D.1).

For well-specified cases such as tabular MDPs and linear MDPs, due to $Q^\pi(\cdot,\cdot; r + b^n) - b^n$ is always a linear function with respect to the features, one can easily show that $\epsilon_{bias} = 0$ (which we show in Appendix E), as one can pick the best on-policy fit $\theta^t_\star$ to be the exact linear representation of $Q^\pi(s, a; r+b^n) - b^n(s, a)$. Further, in the state-aggregation example, we can show that ϵ_{bias} is upper bounded by the expected model-misspecification with respect to the comparator policy's distribution (Appendix F). We refer readers to [3] for detailed discussion with respect to the comparison of transfer error and usual concentrability assumptions [3, 34, 53].

5 Experiments

We provide experiments illustrating PC-PG's performance on problems requiring exploration, and focus on showing the algorithm's flexibility to leverage existing policy gradient algorithms with neural networks (e.g., PPO [56]). Specifically, we show that for challenging exploration tasks, our algorithm combined with PPO significantly outperforms both vanilla PPO as well as PPO augmented with the popular RND exploration bonus [16]. For all experiments, we use policies parameterized by fully-connected or convolutional neural networks. We use a kernel $\phi(s, a)$ to compute bonus as $b(s, a) = \phi(s, a)^\top \hat{\Sigma}_{\text{cov}}^{-1} \phi(s, a)$, where $\hat{\Sigma}_{\text{cov}}$ is the empirical covariance matrix of the policy cover. In order to prune any redundant policies from the cover, we use a rebalancing scheme to select a policy cover which induces maximal coverage over the state space. This is done by finding weights $\alpha^{(n)} = (\alpha_1^{(n)}, ..., \alpha_n^{(n)})$ on the simplex at each episode which solve the optimization problem: $\alpha^{(n)} = \text{argmax}_\alpha \log\det\left[\sum_{i=1}^n \alpha_i \hat{\Sigma}_i\right]$ where $\hat{\Sigma}_i$ is the empirical covariance matrix of π_i. Details of the implemented algorithm, network architectures and kernels can be found in Appendix I.

Bidirectional Diabolical Combination Lock We first provide experiments on an exploration problem designed to be particularly difficult: the Bidirectional Diabolical Combination Lock (a harder version of the problem in [43], see Figure 1). In this problem, the agent starts at an initial state s_0 (left most state), and based on its first action, transitions to one of two combination locks of length H. Each combination lock consists of a chain of length H, at the end of which are two states with high reward. At each level in the chain, 9 out of 10 actions lead the agent to a dead state (black) from which it cannot recover and lead to zero reward. The problem is challenging for exploration for several reasons: (1) *Sparse positive rewards*: Uniform exploration has a 10^{-H} chance of reaching a high reward state; (2) *Dense antishaped rewards*: The agent receives a reward of $-1/H$ for transitioning to a good state and 0 to a dead state. A locally optimal policy is to transition to a dead state quickly; (3) *Forgetting*: At the end of one of the locks, the agent receives a maximal reward of $+5$, and at the end of the other lock it receives a reward of $+2$. Since there is no indication which lock has the optimal reward, agent needs to remember to visit both chains. For both the policy network input and the kernel we used a binary vector encoding the current lock, state and time step

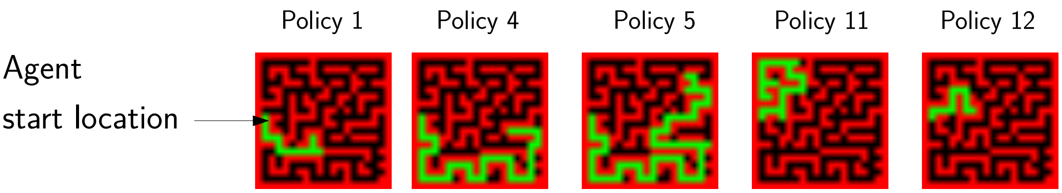


Figure 2: Different policies' trajectory (green) in the policy cover for the maze environment.

as one-hot components. We compared to two other methods: a PPO agent, and a PPO agent with a RND exploration bonus, all of which used the same representation as input.

Figure 1 shows the PPO agent succeeds for the shortest problem of horizon $H = 2$. The PPO+RND agent succeeds roughly 50% of the time: it avoids the local minimum and explores to the end of one of the chains. However the agent's policy quickly becomes deterministic and the agent forgets to go back and explore the other chain after it has reached the reward at the end of the first (as shown in Figure 5 (a)), PC-PG succeeds over all seeds and horizon lengths. We found that the policy cover provides near uniform coverage over both chains (see Figure 5 (b)).

Reward-free Exploration in Mazes We evaluated PC-PG in a reward-free setting using maze environments adapted from [46]. The agent's observation consists of an RGB-image of the maze with the red channel representing the walls and the green channel representing the location of the agent (an example is shown in Figure 2). We compare PC-PG, PPO and PPO+RND where the agent receives a constant environment reward of 0 (note that PPO receives zero gradient; PC-PG and PPO+RND learn from their reward bonus). Figure 3 (left) shows the percentage of locations in the maze visited by each of the agents over the course of 10 million steps. The proportion of states visited by the PPO agent stays relatively constant, while the PPO+RND agent is able to explore to some degree. PC-PG quickly visits a significantly higher proportion of locations. Visualizations of traces from policies can be seen in Figure 2.

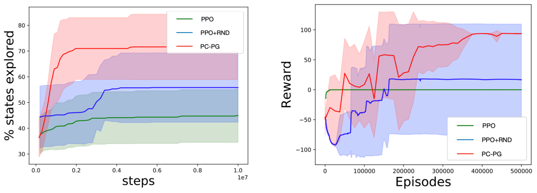


Figure 3: Results for maze (left) & control (right).

Continuous Control We further evaluated PC-PG on continuous control MountainCar from OpenAI Gym [15]. Note here actions are continuous in $[-1, 1]$ and incur a small negative reward. Since the agent only receives a large reward $(+100)$ if it reaches the top of the hill, a locally optimal policy is to do nothing (e.g., PPO never escapes this local optimality in our experiments). Results for PPO, PPO+RND and PC-PG are shown in Figure 3(right). The PPO agent quickly learns the locally optimal policy. The PPO+RND agent exhibits wide variability across seeds: some seeds solve the task while others not. The PC-PG agent consistently discovers a good policy across all seeds. In Figure 6, we show the traces of policies in the policy cover constructed by PC-PG.

6 Discussion

This work proposes a new policy gradient algorithm for balancing the exploration-exploitation tradeoff in RL, which enjoys provable sample efficiency guarantees in the linear and kernelized settings. Our experiments provide evidence that the algorithm can be combined with neural policy optimization methods and be effective in practice. An interesting direction for future work would be to combine our approach with feature learning [4]. in rich observation settings to learn a good feature representation.

Broader Impact

This paper provides a provably efficient policy gradient algorithm. Though the nature of the paper is mostly theoretical and the paper heavily focuses on understanding the theoretical foundations of one of the most popular RL algorithms, i.e., policy gradient, we believe that our theoretical findings and the proposed new algorithm will have a broader impact on the society. Due to the high sample complexity of existing PG methods, they are often limited to applications related to video games. We believe that by providing global optimality and sample efficiency to PG methods, we will significantly broaden the application scope of PG methods. Specifically, our work potentially could enable PG methods to be deployed in real-world applications such as precision medicine, human robot interaction, and personalized eduction systems where safety and robustness are critical.

Acknowledgments and Disclosure of Funding

Part of the work was done when MH and WS were at Microsoft Research NYC. The authors would like to thank Andrea Zanette, Ching-An Cheng, and Xuezhou Zhang for carefully reviewing the proofs, and Akshay Krishnamurthy for helpful discussions. Sham Kakade gratefully acknowledges funding from the ONR award N00014-18-1-2247, and NSF Awards CCF-1703574 and CCF-1740551.

References

[1] Y. Abbasi-Yadkori, P. Bartlett, K. Bhatia, N. Lazic, C. Szepesvari, and G. Weisz. Politex: Regret bounds for policy iteration using expert prediction. In *International Conference on Machine Learning*, pages 3692–3702, 2019.

[2] Y. Abbasi-Yadkori, N. Lazic, C. Szepesvari, and G. Weisz. Exploration-enhanced politex. *arXiv preprint arXiv:1908.10479*, 2019.

[3] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift, 2019.

[4] A. Agarwal, S. Kakade, A. Krishnamurthy, and W. Sun. Flambe: Structural complexity and representation learning of low rank mdps. *arXiv preprint arXiv:2006.10814*, 2020.

[5] S. Agrawal and R. Jia. Optimistic posterior sampling for reinforcement learning: worst-case regret bounds. In *Advances in Neural Information Processing Systems*, pages 1184–1194, 2017.

[6] A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1): 89–129, 2008.

[7] A. Ayoub, Z. Jia, C. Szepesvári, M. Wang, and L. F. Yang. Model-based reinforcement learning with value-targeted regression. abs/2006.01107, 2020. URL https://arxiv.org/abs/2006.01107.

[8] M. G. Azar, V. Gómez, and H. J. Kappen. Dynamic policy programming. *J. Mach. Learn. Res.*, 13(1), Nov. 2012. ISSN 1532-4435.

[9] M. G. Azar, I. Osband, and R. Munos. Minimax regret bounds for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 263–272. JMLR. org, 2017.

[10] J. A. Bagnell, S. M. Kakade, J. G. Schneider, and A. Y. Ng. Policy search by dynamic programming. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 831–838. MIT Press, 2004.

[11] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1471–1479. Curran Associates, Inc., 2016.

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

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

[14] R. I. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

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

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

[17] Q. Cai, Z. Yang, C. Jin, and Z. Wang. Provably efficient exploration in policy optimization. *arXiv preprint arXiv:1912.05830*, 2020.

[18] J. Chen and N. Jiang. Information-theoretic considerations in batch reinforcement learning. In *International Conference on Machine Learning*, pages 1042–1051, 2019.

[19] V. Dani, T. P. Hayes, and S. M. Kakade. Stochastic linear optimization under bandit feedback. In *COLT*, pages 355–366, 2008.

[20] C. Dann and E. Brunskill. Sample complexity of episodic fixed-horizon reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2818–2826, 2015.

[21] C. Dann, N. Jiang, A. Krishnamurthy, A. Agarwal, J. Langford, and R. E. Schapire. On polynomial time PAC reinforcement learning with rich observations. *arXiv preprint arXiv:1803.00606*, 2018.

[22] S. Dong, B. Van Roy, and Z. Zhou. Provably efficient reinforcement learning with aggregated states. *arXiv preprint arXiv:1912.06366*, 2019.

[23] S. S. Du, Y. Luo, R. Wang, and H. Zhang. Provably efficient q-learning with function approximation via distribution shift error checking oracle. *arXiv preprint arXiv:1906.06321*, 2019.

[24] Y. Efroni, L. Shani, A. Rosenberg, and S. Mannor. Optimistic policy optimization with bandit feedback. *arXiv preprint arXiv:2002.08243*, 2020.

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

[26] M. Fazel, R. Ge, S. M. Kakade, and M. Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. *arXiv preprint arXiv:1801.05039*, 2018.

[27] M. Geist, B. Scherrer, and O. Pietquin. A theory of regularized markov decision processes. *arXiv preprint arXiv:1901.11275*, 2019.

[28] M. Henaff. Explicit explore-exploit algorithms in continuous state spaces. In *Advances in Neural Information Processing Systems*, pages 9377–9387, 2019.

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

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

[31] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan. Is q-learning provably efficient? In *Advances in Neural Information Processing Systems*, pages 4863–4873, 2018.

[32] C. Jin, Z. Yang, Z. Wang, and M. I. Jordan. Provably efficient reinforcement learning with linear function approximation. *arXiv preprint arXiv:1907.05388*, 2019.

[33] S. Kakade. A natural policy gradient. In *NIPS*, 2001.

[34] S. Kakade and J. Langford. Approximately Optimal Approximate Reinforcement Learning. In *Proceedings of the 19th International Conference on Machine Learning*, volume 2, pages 267–274, 2002.

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

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

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

[38] T. Lattimore and M. Hutter. Near-optimal pac bounds for discounted mdps. volume 558, pages 125–143. Elsevier, 2014.

[39] T. Lattimore and M. Hutter. Near-optimal pac bounds for discounted mdps. *Theoretical Computer Science*, 558:125–143, 2014.

[40] L. Li. *A unifying framework for computational reinforcement learning theory*. PhD thesis, Rutgers, The State University of New Jersey, 2009.

[41] B. Liu, Q. Cai, Z. Yang, and Z. Wang. Neural proximal/trust region policy optimization attains globally optimal policy. *CoRR*, abs/1906.10306, 2019. URL http://arxiv.org/abs/1906.10306.

[42] T. Lu, D. Schuurmans, and C. Boutilier. Non-delusional q-learning and value-iteration. In *Advances in neural information processing systems*, pages 9949–9959, 2018.

[43] D. Misra, M. Henaff, A. Krishnamurthy, and J. Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, 2020.

[44] R. Munos. Error bounds for approximate value iteration. 2005.

[45] G. Neu, A. Jonsson, and V. Gómez. A unified view of entropy-regularized markov decision processes. *CoRR*, abs/1705.07798, 2017.

[46] J. Oh, S. Singh, and H. Lee. Value prediction network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6118–6128. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7192-value-prediction-network.pdf.

[47] I. Osband, B. Van Roy, and Z. Wen. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[49] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[50] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1313–1320. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3495-weighted-sums-of-random-kitchen-sinks-replacing-minimization-with-randomization-in-lea pdf.

[51] D. Russo. Worst-case regret bounds for exploration via randomized value functions. In *Advances in Neural Information Processing Systems*, pages 14410–14420, 2019.

[52] D. Russo. Approximation benefits of policy gradient methods with aggregated states. *arXiv preprint arXiv:2007.11684*, 2020.

[53] B. Scherrer. Approximate policy iteration schemes: A comparison. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14. JMLR.org, 2014.

[54] B. Scherrer and M. Geist. 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*, pages 35–50. Springer, 2014.

[55] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

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

[57] Z. Shangtong. Modularized implementation of deep rl algorithms in pytorch. https://github.com/ShangtongZhang/DeepRL, 2018.

[58] L. Shani, Y. Efroni, and S. Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps, 2019.

[59] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: no regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 1015–1022, 2010.

[60] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 881–888. ACM, 2006.

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

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

[63] C. Szepesvári and R. Munos. Finite time bounds for sampling based fitted value iteration. In *Proceedings of the 22nd international conference on Machine learning*, pages 880–887. ACM, 2005.

[64] I. Szita and C. Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 1031–1038, 2010.

[65] J. A. Tropp et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

[66] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini. Finite-time analysis of kernelised contextual bandits. *arXiv preprint arXiv:1309.6869*, 2013.

[67] Z. Wen and B. Van Roy. Efficient exploration and value function generalization in deterministic systems. In *Advances in Neural Information Processing Systems*, pages 3021–3029, 2013.

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

[69] L. F. Yang and M. Wang. Reinforcement leaning in feature space: Matrix bandit, kernels, and regret bound. *arXiv preprint arXiv:1905.10389*, 2019.

[70] L. F. Yang and M. Wang. Sample-optimal parametric q-learning using linearly additive features. In *International Conference on Machine Learning*, pages 6995–7004, 2019.

[71] A. Zanette, D. Brandfonbrener, E. Brunskill, M. Pirotta, and A. Lazaric. Frequentist regret bounds for randomized least-squares value iteration. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1954–1964, Online, 26–28 Aug 2020. PMLR.

[72] D. Zhou, J. He, and Q. Gu. Provably efficient reinforcement learning for discounted mdps with feature mapping. *arXiv preprint arXiv:2006.13165*, 2020.

[73] M. Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 928–936, 2003.