Contrasting Exploration in Parameter and Action Space: A Zeroth-Order Optimization Perspective

Anirudh Vemula
Robotics Institute
Carnegie Mellon University
vemula@cmu.edu

Wen Sun
Robotics Institute
Carnegie Mellon University
wensun@cs.cmu.edu

J. Andrew Bagnell
Aurora Innovation
dbagnell@ri.cmu.edu

Abstract

Black-box optimizers that explore in parameter space have often been shown to outperform more sophisticated action space exploration methods developed specifically for the reinforcement learning problem. We examine these black-box methods closely to identify situations in which they are worse than action space exploration methods and those in which they are superior. Through simple theoretical analyses, we prove that complexity of exploration in parameter space depends on the dimensionality of parameter space, while complexity of exploration in action space depends on both the dimensionality of action space and horizon length. This is also demonstrated empirically by comparing simple exploration methods on several model problems, including Contextual Bandit, Linear Regression and Reinforcement Learning in continuous control.

1 Introduction

Model-free policy search is a general approach to learn parameterized policies from sampled trajectories in the environment without learning a model of the underlying dynamics. These methods update the parameters such that trajectories with higher returns (or total reward) are more likely to be obtained when following the updated policy (Kober et al., 2013). The simplicity of these approaches have made them popular in Reinforcement Learning (RL).

Policy gradient methods, such as REINFORCE (Williams, 1992) and its extensions (Kakade, 2002; Bagnell et al., 2004; Silver et al., 2014; Schulman et al., 2015), compute an estimate of a direction of improvement from sampled trajectories collected by executing a stochastic policy. In other words, these methods rely on randomized exploration in action space. These methods then leverage the Jacobian of the policy to update its parameters to increase the probability of good action sequences accordingly. Such a gradient estimation algorithm can be considered a combination of a *zeroth-order* approach and a *first-order* approach: (1) it never exploits the slope of the reward function or dynamics, with respect to actions, but rather relies only on random exploration in action space to discover potentially good sequences of actions; (2) however, it *exploits* the first order information of the parameterized policy for updating the policy's parameters. Note that the chance of finding a sequence of actions resulting in high total reward decreases (as much as exponentially (Kakade and Langford, 2002)) as the horizon length increases and thus policy gradient methods often exhibit high variance and a resulting large sample complexity (Peters and Schaal, 2008; Zhao et al., 2011).

Black-box policy search methods, on the other hand, seek to directly optimize the total reward in the space of parameters by employing , e.g., finite-difference-like methods to compute estimates of the gradient with respect to policy parameters (Bagnell and Schneider, 2001; Mannor et al., 2003; Heidrich-Meisner and Igel, 2008; Tesch et al., 2011; Sehnke et al., 2010; Salimans et al., 2017; Mania et al., 2018). Intuitively, these methods rely on exploration in parameter space: by searching in the parameter space, these methods may discover an improvement direction. Note that these methods are fully zeroth-order, *i.e.*, they exploit no first-order information of the parameterized policy, the reward, or the dynamics. Although policy gradient methods leverage *more* information, notably the Jacobian of the action with respect to policy, black-box policy search methods have at times demonstrated better

Proceedings of the 22[nd] International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

empirical performance (see the discussion in (Kober et al., 2013; Mania et al., 2018)). These perhaps surprising results motivate us to analyze: *In what situations should we expect parameter space policy search methods to outperform action space methods?*

To do so, we leverage prior work in zeroth-order optimization methods. In the convex setting, (Flaxman et al., 2005; Agarwal et al., 2010; Nesterov and Spokoiny, 2017) showed that one can construct gradient estimates using zeroth order oracles and derived upper bounds on the number of samples needed. But for most RL tasks, the return as a function of parameters, or action sequence, is highly non-convex (Sutton and Barto, 1998). Hence we focus on the non-convex setting and analyze convergence to stationary points. Ghadimi and Lan (2013); Nesterov and Spokoiny (2017) studied zeroth order non-convex optimization by providing upper bounds on the number of samples needed to close in on a stationary point. Computing lower bounds in zeroth order non-convex optimization is still an open problem (Carmon et al., 2017a,b).

In our work, we extend the analysis proposed in (Ghadimi and Lan, 2013) to the policy search setting and analyze the sample complexity of parameter and action space exploration methods in policy search. We begin with a degenerate, one-step control problem of online linear regression with partial feedback, (Flaxman et al., 2005), where the objective is to learn the parameters of the linear regressor without access to the true scalar regression targets. We show that for parameter space exploration methods, to achieve ϵ-optimality, requires $O(b^2/\epsilon^4)$ samples, where b is the input feature dimensionality. By contrast, an action space exploration method requires $O(1/\epsilon^4)$ many samples with a sample complexity *independent* of input feature dimensionality b. This is tested empirically on two simple tasks: Bandit Multi-class learning on MNIST with policies parameterized by convolutional neural networks which can be seen as a Contextual Bandit problem with rich observations, and Online Linear Regression with partial information. The results demonstrate action space exploration methods outperform parameter space methods when the parameter dimensionality is substantially larger than action dimensionality.

We present similar analysis for the multi-step control problem of model-free policy search in reinforcement learning, (Kober et al., 2013), by considering the objective of reaching ϵ-close to a stationary point in the sense that $\|\nabla J(\theta)\|_2^2 \leq \epsilon$ for the non-convex objective $J(\theta)$. Our results show that, under certain assumptions, parameter space exploration methods need $\mathcal{O}(\frac{d^2}{\epsilon^3})$ samples to reach ϵ close to a stationary point, where d is the policy parameter dimensionality. On the other hand, action space exploration methods need $\mathcal{O}(\frac{p^2H^4}{\epsilon^4})$ samples to achieve the same objective, where p is the action dimensionality and H is the horizon length of the task. This shows that action space exploration methods have a dependence on the horizon length H while parameter space exploration methods depend only on parameter space dimensionality dOngoing work by Tu and Recht (2018) demonstrated through asymptotic lower bounds that the dependence of sample complexity of action space exploration methods on horizon H is unavoidable in the LQR setting. This is tested empirically on popular RL benchmarks from OpenAI gym (Brockman et al., 2016a), and the results show that as horizon length increases, parameter space methods outperform action space exploration methods. This matches the intuition and results presented in recent works like (Bagnell and Schneider, 2001; Szita and Lörincz, 2006; Tesch et al., 2011; Salimans et al., 2017; Mania et al., 2018) that show parameter space black-box policy search methods outperforming state-of-the-art action space methods for tasks with long horizon lengths.

In summary, our analysis and experimental results suggests that the complexity of exploration in action space depends on both the dimensionality of action space and *horizon*, while the complexity of exploration in parameter space solely depends on dimensionality of parameter space, providing a natural way to trade-off between these approaches.

2 Problem Setup

2.1 Multi-step Control: Reinforcement Learning

We consider the problem setting of model-free policy search with the goal of minimizing sum of costs (or maximizing sum of rewards) over a fixed, finite horizon H. In reinforcement learning (RL), this is typically formulated using Markov Decision Processes (MDP) (Sutton and Barto, 1998). Denote the state space of the MDP as $\mathcal{S} \subset \mathbb{R}^b$, action space as $\mathcal{A} \subset \mathbb{R}^p$, transition probabilities as $\mathbb{P}_{sa} = \mathbb{P}(\cdot|s,a)$ (which is the distribution of next state after executing action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$), an initial state distribution μ, and a cost function $c(s,a) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Note that the cost can be interpreted as negative of the reward. In addition to this, we assume a restricted class of deterministic, stationary policies Π parameterized by $\theta \in \mathbb{R}^d$ where each $\pi(\theta, \cdot) \in \Pi$ is differentiable at all θ and is a mapping from $\mathcal{S}$ to $\mathcal{A}$, i.e. $\pi(\theta, \cdot) : \mathcal{S} \to \mathcal{A}$. The distribution of states at timestep t induced by running the policy $\pi(\theta, \cdot)$ until and including t, is defined $\forall s_t : d^t_{\pi_\theta}(s_t) =$

$\sum_{\{s_i\}_{i\leq t-1}} \mu(s_0) \prod_{i=0}^{t-1} \mathbb{P}(s_{i+1}|s_i, a_i = \pi(\theta, s_i))$, where by definition $d^0_{\pi_\theta}(s) = \mu(s)$ for any π. We define the value function $V^t_{\pi_\theta}(s)$ for $t \leq H-1$ as

$$V^t_{\pi_\theta}(s) = \mathbb{E}[\sum_{i=t}^{H} c(s_i, \pi(\theta, s_i))|s_t = s]$$

and state-action value function $Q^t_{\pi_\theta}(s, a)$ as

$$Q^t_{\pi_\theta}(s, a) = c(s, a) + \mathbb{E}_{s' \sim \mathbb{P}_{sa}}[V^{t+1}_{\pi_\theta}(s')]$$

Throughout this work, we assume the total cost is upper bounded by a constant, i.e., $\sup_{c_1,\ldots,c_T} \sum_t c_t \leq \mathcal{Q} \in \mathbb{R}^+$, to prevent confounding due to just a change in the scale of total costs. We have then that $Q^t_{\pi_\theta}$ is upper bounded by a constant $\mathcal{Q}$ for all t and θ.

We seek to minimize the performance objective given by $J(\theta) = \mathbb{E}_{s\sim\mu}[V^0_{\pi_\theta}(s)]$. Given this objective, the optimization problem can be formulated as:

$$\min_\theta J(\theta) \tag{1}$$

The goal is to find parameters θ^* that minimize the expected sum of costs $J(\theta)$, given no access to the underlying dynamics of the environment other than samples from the distribution $\mathbb{P}_{sa}$ by executing the policy $\pi(\theta, \cdot)$. However, the objective $J(\theta)$ can be highly non-convex and finding a global minima could be intractable. Thus, in this work, we hope to find a stationary point θ^* of the objective $J(\theta)$, *i.e.* a point where $\nabla_\theta J(\theta) \approx 0$.

2.2 One-Step Control: Online Linear Regression with Partial Information

The online linear regression problem is defined as follows: We denote $\mathcal{S} \subset \mathbb{R}^b$ as the feature space, and $\Theta \subset \mathbb{R}^d = \mathbb{R}^b$ as the linear policy parameter space where each $\theta \in \Theta$ represents a policy $\pi(\theta, s) = \theta^\top s$. Online linear regression operates in an adversarial online learning fashion: every round i, nature presents a feature vector $s_i \in \mathcal{S}$, the learner makes a decision by choosing a policy $\theta_i \in \Theta$ and predicts the scalar action $\hat{a}_i = \theta_i^\top s_i$; nature then reveals the loss $(\hat{a}_i - a_i)^2 \in \mathbb{R}^+$, which is just a scalar, to the learner, where a_i is ground truth selected by nature and is never revealed to the learner. We do not place any statistical assumption on the nature's process of generating feature vector s_i and ground truth a_i, which could be completely adversarial. Other than the adversarial aspect of the problem, note that the above setup is a special setting of RL with horizon $H = 1$, linear policy $\theta^\top s_i$, one-dimension action space, and a cost function $c_i(\theta) = (\theta^\top s_i - a_i)^2$. In this setting, we consider the *regret* with respect to the optimal solution in hindsight,

$$\text{Regret} = \sum_{i=1}^{T} c_i(\theta_i) - \min_{\theta^\star \in \Theta} \sum_{i=1}^{T} c_i(\theta^\star) \tag{2}$$

3 Online Linear Regression with Partial Information

3.1 Exploration in Parameter Space

We can apply a zeroth-order online gradient descent algorithm for the sequence of loss functions $\{c_i\}_{i=1}^T$, which is summarized in Algorithm 1. The main idea is to add random noise u, sampled from a unit sphere in b-dim space $\mathbb{S}_b$, to the parameter θ, and querying loss at $\theta + \delta u$ for some $\delta > 0$. Using the received loss $c_i(\theta + \delta u)$, one can form an estimation of $\nabla_\theta c_i(\theta)$ as $\frac{c_i b}{\delta} u$ (Flaxman et al., 2005).

Algorithm 1 Random Search in Parameter Space (BGD Flaxman et al. (2005))

Input: $\alpha \in \mathbb{R}^+$, $\delta \in \mathbb{R}^+$.
Learner initializes $\theta_1 \in \Theta$.
for $i = 1$ to T **do**
  Learner samples $u \sim \mathbb{S}_b$.
  Learner chooses predictor $\theta'_i = \theta_i + \delta u$.
  Learner only receives loss signal $c_i(\theta'_i)$.
  Learner update: $\theta'_{i+1} = \theta_i - \alpha \frac{c_i b}{\delta} u$.
  Projection $\theta_{i+1} = \arg\min_{\theta\in\Theta} \|\theta'_{i+1} - \theta\|_2^2$.
end for

3.2 Exploration in Action Space

The *key difference* between exploration in action space and exploration in parameter space is that we are going to leverage our knowledge of the policy $\pi(\theta, s) = \theta^\top s$. Since we design the policy class, we can compute its *Jacobian* with respect to its parameters θ without interaction with the environment. The Jacobian of the policy gives us a locally linear relationship between a small change in parameter and the resulting change in policy's action space. The main idea then in this approach is to explore with randomization in action space, and then leverage the Jacobian of the policy to update the parameters θ accordingly so that the policy's output moves towards better actions. Intuitively, we expect that random exploration in action space will result in smaller regret, as in our setting the action space is just 1-dimensional, while the parameter space is b-dimensional. The approach is summarized in Algorithm 2. Denote $\ell_i = (\hat{a}_i - a_i)^2$ and $\hat{a}_i = \pi(\theta_i, s_i) = \theta_i^\top s_i$. The main idea is that we can compute $\nabla_\theta c_i(\theta_i)$ via a chain rule as $\nabla_\theta c_i(\theta_i) = \frac{\partial \ell_i}{\partial \hat{a}_i} \nabla_\theta \pi(\theta_i, s_i)$. Note that $\nabla_\theta \pi(s_i, \theta_i) = \nabla_\theta \theta_i^\top s_i = s_i$ is the Jacobian of the

policy to which we have full access. We then use zeroth order approximation method to approximate $\partial \ell_i / \partial \hat{a}_i$ at $\hat{a}_i = \pi(\theta_i, s_i)$.

Algorithm 2 Random Search in Action Space

1: **Input:** $\alpha \in \mathbb{R}^+$, $\delta \in \mathbb{R}^+$.
2: Learner initializes $\theta_1 \in \Theta$.
3: **for** $i = 1$ to T **do**
4: Learner receives feature s_i.
5: Learner samples e uniformly from $\{-1, 1\}$.
6: Learner makes a prediction $\hat{a}_i = \theta_i^\top s_i + \delta e$
7: Learner only receives loss signal $c_i = (\hat{a}_i - a_i)^2$.
8: Learner update: $\theta'_{i+1} = \theta_i - \alpha \frac{c_i e}{\delta} s_i$.
9: Projection $\theta_{i+1} = \arg\min_{\theta \in \Theta} \|\theta'_{i+1} - \theta\|_2^2$.
10: **end for**

3.3 Analysis

We analyze the regret of the exploration in parameter space algorithm (Alg. 1) and the exploration in action space algorithm (Alg. 2) in this section. For analysis, we assume that Θ is bounded, i.e., $\sup_{\theta \in \Theta} \|\theta\|_2 \leq C_\theta \in \mathbb{R}^+$, $\mathcal{S}$ is bounded, i.e., $\sup_{s \in \mathcal{S}} \|s\|_2 \leq C_s \in \mathbb{R}^+$, and the ground truth a_i is bounded, i.e., $|a_i| \leq C_a$ for any i. Under the above assumptions, we can make sure that the loss is bounded as well, $(\theta^\top s - a)^2 \leq C \in \mathbb{R}^+$. The loss function is also Lipschitz continuous with Lipschitz constant $L \leq (C_\theta C_s + C_a) C_s$. We call these constants C_s, C_θ, and C_a as problem dependent constants, which are independent of feature dimension b and number of rounds T. In regret bounds, we absorb problem dependent constants into $\mathcal{O}$ notations, but the bounds will be explicit in b and T. The theorem below presents the average regret analysis for these methods,

Theorem 3.1. *After T rounds, with $\alpha = \frac{C_\theta \delta}{b(C^2 + C_s^2)\sqrt{T}}$ and $\delta = T^{-0.25} \sqrt{\frac{C_\theta b (C^2 + C_s^2)}{2L}}$, Alg. 1 incurs average regret:*

$$\frac{1}{T}(\mathbb{E}[\sum_{i=1}^{T} c_i(\theta_i)] - \min_{\theta^\star \in \Theta} \sum_{i=1}^{T} c_i(\theta^\star)) \leq \mathcal{O}(\sqrt{b} T^{-\frac{1}{4}}), \quad (3)$$

and with $\alpha = \frac{C_\theta \delta}{(C^2+1) C_s \sqrt{T}}$ and $\delta = T^{-0.25} \sqrt{\frac{C_\theta (C^2+1) C_s}{2C}}$, Alg. 2 incurs average regret:

$$\frac{1}{T}(\mathbb{E}[\sum_{i=1}^{T} c_i(\theta_i)] - \min_{\theta^\star \in \Theta} \sum_{i=1}^{T} c_i(\theta^\star)) \leq \mathcal{O}(T^{-\frac{1}{4}}), \quad (4)$$

for any $\theta \in \Theta$.

The above regret analysis essentially shows that exploration in action space delivers a regret bound that is independent of parameter space dimension b, while the regret of the exploration in parameter space algorithm will have explicit polynomial dependency on feature dimension b. Converting the regret bounds to sample complexity bounds, we have that for any $\epsilon \in (0, 1)$, to achieve ϵ-average regret, Alg. 1 needs $\mathcal{O}(\frac{b^2}{\epsilon^4})$ many rounds, while Alg. 2 requires $\mathcal{O}(1/\epsilon^4)$ many rounds.

Note that in general if we have a multivariate regression problem, i.e., $a \in \mathbb{R}^p$, regret of Algorithm 2 will depend on $\sqrt{p}$ as well. But from our extreme case with $p = 1$, we clearly demonstrate the sharp advantage of exploration in action space: *when the action space's dimension is smaller than the dimension of parameter space*, we should prefer the strategy of exploration in action space.

4 Reinforcement Learning

In this section, we study exploration in parameter space versus exploration in action space for multi-step control problem of model-free policy search in RL. As explained in Section 2, we are interested in rates of convergence to a stationary point of $J(\theta)$.

4.1 Exploration in Parameter Space

The objective defined in Section 2.1 can be optimized directly over the space of parameters $\mathbb{R}^d$. Since we do not use first-order (or gradient) information about the objective, this is equivalent to derivative-free (or zeroth-order) optimization with noisy function evaluations. More specifically, for a parameter vector θ, we can execute the corresponding policy $\pi(\theta, \cdot)$ in the environment, to obtain a noisy estimate of $J(\theta)$. This noisy function evaluation can be used to construct a gradient estimate and an iterative stochastic gradient descent approach can be used to optimize the objective. An algorithm that closely follows the ones proposed in (Agarwal et al., 2010; Mania et al., 2018) and optimizes over the space of parameters is shown in Algorithm 3. Since we are working in episodic RL setting, we can use a two-point estimate to form a gradient estimation (Line 7 & 8 in Alg. 3), which in general will reduce the variance of gradient estimation (Agarwal et al., 2010), compared to one-point estimates. We will analyze the finite rate of convergence of Algorithm 3 to a stationary point of the non-convex objective $J(\theta)$. First, we will lay out the assumptions and then present the convergence analysis.

Assumptions and Analysis To analyze convergence to stationary point of a nonconvex objective, we make several assumptions about the objective. Firstly, we assume that $J(\theta)$ is differentiable with respect to θ over the entire domain. We also assume that $J(\theta)$ is G-lipschitz and L-smooth, i.e. for all $\theta_1, \theta_2 \in \mathbb{R}^d$, we have $|J(\theta_1) - J(\theta_2)| \leq G\|\theta_1 - \theta_2\|$

Algorithm 3 Policy Search in Parameter Space

1: **Input:** Learning rate $\alpha \in \mathbb{R}^+$, standard deviation of exploration noise $\delta \in \mathbb{R}$
2: Initialize parameters $\theta_1 \in \mathbb{R}^d$
3: **for** $i = 1$ to T **do**
4: Sample $u \sim \mathbb{S}_d$, a d-dimensional unit sphere
5: Construct parameters $\theta_i + \delta u$, $\theta_i - \delta u$
6: Execute policies $\pi(\theta_i + \delta u, \cdot), \pi(\theta_i - \delta u, \cdot)$
7: Obtain noisy estimates of the objective $J_i^+ = J(\theta_i + \delta u) + \eta_i^+$ and $J_i^- = J(\theta_i - \delta u) + \eta_i^-$ where η_i^+, η_i^- are zero mean random i.i.d noise
8: Compute gradient estimate $g_i = \frac{d(J_i^+ - J_i^-)}{2\delta} u$
9: Update $\theta_{i+1} = \theta_i - \alpha g_i$
10: **end for**

and $\|\nabla_\theta J(\theta_1) - \nabla_\theta J(\theta_2)\| \leq L\|\theta_1 - \theta_2\|$. Note that these assumptions are similar to the assumptions made in other zeroth-order analysis works, (Flaxman et al., 2005; Agarwal et al., 2010; Duchi et al., 2015; Shamir, 2013; Ghadimi and Lan, 2013; Nesterov and Spokoiny, 2017).

Our analysis is along the lines of works like (Ghadimi and Lan, 2013; Nesterov and Spokoiny, 2017) that also analyze the convergence to stationary points in zeroth order non-convex optimization. The general strategy is to first construct a smoothed version of the objective $J(\theta)$, denoted as $\hat{J}(\theta) = \mathbb{E}_{v \sim \mathbb{B}_d}[J(\theta + \delta v)]$, where $\mathbb{B}_d$ is the d-dimensional unit ball. We can then show that Algorithm 3 is essentially running SGD on the objective function $\hat{J}(\theta)$, which allows us to apply standard SGD analysis on $\hat{J}(\theta)$. Lastly we link the stationary point of the smoothed objective $\hat{J}(\theta)$ to that of the objective $J(\theta)$ using the assumptions on $J(\theta)$.

Theorem 4.1. *Consider running Algorithm 3 for T steps where the true objective $J(\theta)$ satisfies the assumptions stated above. Then we have,*

$$\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(\mathcal{Q}^{\frac{1}{2}} d T^{\frac{-1}{2}} + \mathcal{Q}^{\frac{1}{3}} d^{\frac{2}{3}} T^{\frac{-1}{3}} \sigma) \quad (5)$$

where $J(\theta) \leq \mathcal{Q}$ for all $\theta \in \Theta$ and σ^2 is the variance of the random noise η in Algorithm 3.

The above theorem gives us a convergence rate to a stationary point of policy search in parameter space. The role of variance of i.i.d noise in the noisy evaluations of the true objective is very important. Consider the case where there is little stochasticity in the environment dynamics, i.e. $\sigma \to 0$, then the first term in Equation 5 becomes dominant and we only need at most $\mathcal{O}(\frac{d^2\mathcal{Q}}{\epsilon^2})$ samples to reach a point θ where $\mathbb{E}\|\nabla_\theta J(\theta)\|_2^2 \leq \epsilon$. However, if there is a lot of stochasticity in the environment dynamics then the second term is dominant and we need at most $\mathcal{O}(\frac{d^2\mathcal{Q}\sigma^3}{\epsilon^3})$ samples. It is interesting to observe the direct impact that the stochasticity of environment dynamics has on convergence rate of policy search, which is also experimentally demonstrated in Sec. 5.2. Note that the convergence rate has no dependency on horizon length H because of the regularity assumption we used on total reward: J is always bounded by a constant $\mathcal{Q}$ that is independent of H. However, as we will see later, even under the regularity assumption convergence rate of action space exploration methods have an explicit dependence on H which will prove to be the primary reason why black-box parameter space policy search methods in (Mania et al., 2018) have been so effective when compared to action space methods.

4.2 Exploration in Action Space

Another way to optimize the objective defined in Section 2.1 is to optimize over the space of actions $\mathcal{A}$. From (Silver et al., 2014), we know that for $J(\theta) = \mathbb{E}_{s\sim\mu}[V^0_{\pi_\theta}(s)]$ we can express the gradient as

$$\nabla_\theta J(\theta) = \sum_{t=0}^{H-1} \mathbb{E}_{s_t \sim d^t_{\pi_\theta}}[\nabla_\theta \pi(\theta, s_t) \nabla_a Q^t_{\pi_\theta}(s_t, \pi(\theta, s_t))] \quad (6)$$

Observe that the first term in the above gradient $\nabla_\theta \pi(\theta, s)$ is the Jacobian of the policy, the local linear relationship between a small change in policy parameters θ and a small change in its output, i.e., actions. The second term $\nabla_a Q(s, a)$ is actually the improvement direction at state action pair (s, a), i.e., conditioned on state s, if we move action a an infinitesimally small step along the negative gradient $-\nabla_a Q(s, a)$, we decrease the cost-to-go $Q(s, a)$. Eqn 6 then leverages policy's Jacobian to transfer the improvement direction in action space to an improvement direction in parameter space.

We can compute Jacobian $\nabla_\theta \pi(\theta, s)$ exactly as we have knowledge of the policy function, i.e, we can leverage the first-order information of the parameterized policy. The second term $\nabla_a Q^t_{\pi_\theta}(s, \pi(\theta, s_t))$, however, is unknown as it depends on the dynamics and cost functions and needs to be estimated by interacting with the environment. We could employ a similar algorithm as Algorithm 3, shown in Algorithm 4, to obtain an estimate of the gradient $\nabla_a Q^t_{\pi_\theta}(s, \pi(\theta, s_t))$, i.e., a zeroth order estimation of $\nabla_a Q^t_{\pi_\theta}$, computed as $\frac{p\tilde{Q}_i}{\delta} u$, where $\tilde{Q}_i$ is an unbiased estimate of $Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u)$, with $u \sim \mathbb{S}_p$ (Line 7 & 9 in Alg. 4).

Another important difference from Algorithm 3 is the fact that we use a one-point estimate for the gradient g_i in Algorithm 4. We cannot employ the idea of two-

point estimate in random exploration in action space to reduce the variance of the estimate of $\nabla_a Q^t_{\pi_\theta}(s_t, a)$. This is due to the fact that environment is stochastic, and we cannot guarantee that we will reach the same state s_t at any two independent roll-ins with π_θ at time step t. Similar to Section 4.1, we will analyze the rate of convergence of Algorithm 4 to a stationary point of the objective $J(\theta)$. The following section will lay out the assumptions and present the convergence analysis.

Algorithm 4 Policy Search in Action Space

Input: Learning rate $\alpha \in \mathbb{R}^+$, standard deviation of exploration noise $\delta \in \mathbb{R}$, Horizon length H, Initial state distribution μ
Initialize parameters $\theta_1 \in \mathbb{R}^d$
for $i = 1$ to T **do**
  Sample $u \sim \mathbb{S}_p$, a p-dimensional unit sphere
  Sample uniformly $t \in \{0, \cdots, H-1\}$
  Execute policy $\pi(\theta_i, \cdot)$ until $t-1$ steps
  Execute perturbed action $a_t = \pi(\theta_i, s_t) + \delta u$ at timestep t and continue with policy $\pi(\theta_i, \cdot)$ until timestep $H-1$ to obtain an estimate $\tilde{Q}_i = Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u) + \tilde{\eta}_i$ where $\tilde{\eta}_i$ is zero mean random noise
  Compute policy Jacobian $\Psi_i = \nabla_\theta \pi(\theta_i, s_t)$
  Compute gradient estimate $g_i = H\Psi_i \frac{p\tilde{Q}_i}{\delta} u$
  Update $\theta_{i+1} = \theta_i - \alpha g_i$
end for

Assumptions and Analysis The assumptions for policy search in action space are similar to the assumptions in Section 4.1. We assume that $J(\theta)$ is differentiable with respect to θ over the entire domain. We also assume that $J(\theta)$ is G-lipschitz and L-smooth. In addition to these assumptions, we will assume that the policy function $\pi(\theta, s)$ is K-lipschitz in θ and the state-action value function $Q^t_{\pi_\theta}(s, a)$ is W-lipschitz and U-smooth in a. Finally, we assume that the state-action value function $Q(s, a)$ is differentiable with respect to a over the entire domain. Note that the Lipschitz assumptions above on $J(\theta)$, $Q^t_{\pi_\theta}(s, a)$, and $\pi(\theta, s)$ are also used in the analysis of Deterministic policy gradient (Silver et al., 2014). We need extra smoothness assumption to study the convergence of our algorithms.

Note that the gradient estimate g_i used in Algorithm 4 is a biased estimate of $\nabla_\theta J(\theta)$. We can show this by considering

$$\mathbb{E}_i[g_i] = \mathbb{E}_t \mathbb{E}_{s_t \sim d^t_{\pi_{\theta_i}}} \left[H \nabla_\theta \pi(\theta_i, s_t) \mathbb{E}_{u \sim \mathbb{S}_p} \left[\frac{p\tilde{Q}_i}{\delta} u \right] \right]$$

where $\mathbb{E}_i$ denotes expectation with respect to the randomness at iteration i. From (Flaxman et al., 2005), we have that $\mathbb{E}[\frac{p\tilde{Q}_i}{\delta} u] = \nabla_a \mathbb{E}_{v \sim \mathbb{B}_p}[Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v)]$ so we can rewrite the above equation as

$$\mathbb{E}[g_i] = \sum_{t=0}^{H-1} \mathbb{E}_{s_t \sim d^t_{\pi_{\theta_i}}} \mathbb{E}_{v \sim \mathbb{B}_p} [\nabla_\theta \pi(\theta_i, s_t) \nabla_a Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v)]$$

Comparing the above expression with equation 6, we can see that g_i is not an unbiased estimate of the gradient $\nabla_\theta J(\theta)$. We can also explicitly upper bound the variance of g_i by $\mathbb{E}_i \|g_i\|_2^2$. Note that in the limit when $\delta \to 0$, g_i becomes an unbiased estimate of $\nabla_\theta J(\theta)$, but the variance will approach to infinity. In our analysis, we explicitly tune δ to balance the bias and variance.

Theorem 4.2. *Consider running Algorithm 4 for T steps where the objective $J(\theta)$ satisfies the assumptions stated above. Then, we have*

$$\frac{1}{T} \sum_{i=1}^{T} \mathbb{E} \|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(T^{-\frac{1}{4}} H p^{\frac{1}{2}} (\mathcal{Q}^3 + \sigma^2 \mathcal{Q})^{\frac{1}{4}}) \quad (7)$$

where $J(\theta) \leq \mathcal{Q}$ for all $\theta \in \Theta$ and σ^2 is the variance of the random noise $\tilde{\eta}$ in Algorithm 4.

The above theorem gives us a convergence rate to a stationary point of $J(\theta)$ for policy search in action space. This means that to reach a point θ where $\mathbb{E}\|\nabla_\theta J(\theta)\|_2^2 \leq \epsilon$, policy search in action space needs at most $\mathcal{O}\left(\frac{p^2 H^4}{\epsilon^4}(\mathcal{Q}^3 + \sigma^2 \mathcal{Q})\right)$ samples. Interestingly, the convergence rate has a dependence on the horizon length H, unlike policy search in parameter space. Also, observe that the convergence rate has no dependence on the parameter dimensionality d as we have complete knowledge of the Jacobian of policy, and we have a dependence on stochasticity of the environment σ that slows down the convergence as the stochasticity increases, similar to policy search in parameter space.

5 Experiments

Given the analysis presented in the previous sections, we test the convergence properties of parameter and action space policy search approaches across several experiments: Contextual Bandit with rich observations, Linear Regression, RL benchmark tasks and Linear Quadratic Regulator (LQR). We use Augmented Random Search (ARS), from (Mania et al., 2018), as the policy search in parameter space method in our experiments as it has been empirically shown to be effective in RL tasks. For policy search in action space, we use either REINFORCE (Williams, 1992), or ExAct (Exploration in Action Space), the method described by Algorithm 4. In all the plots shown, solid lines represent the mean estimate over 10 random seeds and

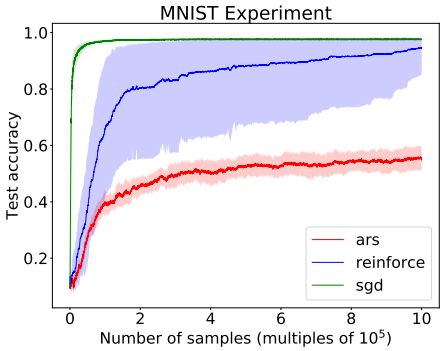


Figure 1: Mean test accuracy with standard error for different approaches against number of samples

shaded regions correspond to ± 1 standard error. The code for all our experiments can be found here[1][2].

5.1 One-Step Control

In these sets of experiments, we test the convergence rate of policy search methods for one time-step prediction. The objective is to minimize the instantaneous cost incurred. The motivation behind such experiments is that we want to understand the dependence of different policy search methods on parametric dimensionality d without the effect of horizon length H.

MNIST as a Contextual Bandit Our first set of experiments is the MNIST digit recognition task (LeCun et al., 1998). To formulate the task in an RL framework, we consider a sequential decision making problem where at each time-step the agent is given the features of the image and needs to predict one of ten actions (corresponding to digits). A reward of $+1$ is given for predicting the correct digit, and a reward of -1 for an incorrect prediction. With this reduction, the problem is essentially a Contextual Bandit Problem (Agarwal et al., 2014). We use a standard LeNet-style convolutional architecture, (LeCun et al., 1998), with $d = 21840$ trainable parameters. Figure 1 shows the learning curves for SGD under standard full-information supervised learning setting with cross entropy loss, REINFORCE and ARS. We can observe that in this setting where the parameter space dimensionality d significantly exceeds the action space dimensionality $p = 1$, policy search in action space outperforms parameter space methods.

Linear Regression with Partial Information These set of experiments are designed to understand how the sample complexity of different policy search methods vary as the parametric complexity is varied. More specifically, from our analysis in Section 3, we know that sample complexity of parameter space methods have a dependence on d, the parametric complexity, whereas action space methods have no dependence on d. We test this hypothesis in this experiment using artificial data with varying input dimensionality and output scalar values. Figure 2 shows the learning curves for standard full-information supervised learning approaches with full access to the square loss (SGD & Newton), REINFORCE, natural REINFORCE (Kakade, 2002), and ARS as we increase the input dimensionality, and hence parametric dimensionality d. Note that we have not included natural REINFORCE and Newton method in Figure 2c as extensive hyperparameter search for these methods is computationally expensive in such high dimensionality settings. The learning curves in Figure 2 match our expectations, and show that action space policy search methods do not degrade as parametric dimensionality increases whereas parameter space methods do. Moreover, action space methods lie between the curves of supervised learning and parameter space methods as they take advantage of the Jacobian of the policy and learn more quickly than parameter space methods.

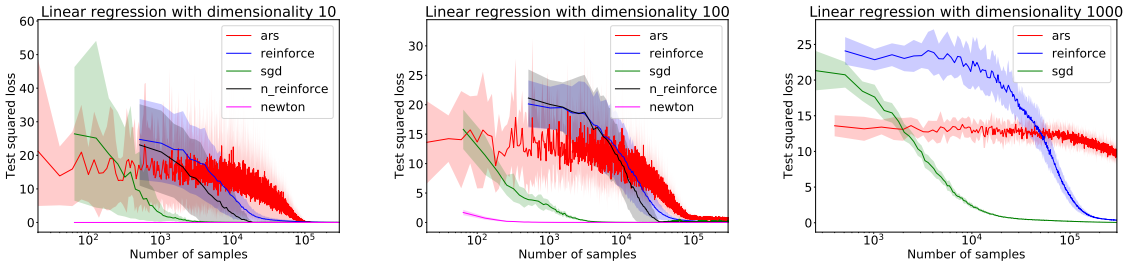
5.2 Multi-Step Control

The above experiments provide insights on the dependence of policy search methods on parametric dimensionality d. We now shift our focus on to the dependence on horizon length H. In this set of experiments, we extend the time horizon and test the convergence rate of policy search methods for multi-step control. The objective is to minimize the sum of costs incurred over a horizon H, i.e. $J(\theta) = \mathbb{E}[\sum_{t=1}^{T} c(s_t, a_t)]$. According to our analysis, we expect action space policy search methods to have a dependence on the horizon length H.

We test ARS and ExAct on two popular continuous control simulated benchmark tasks in OpenAI gym (Brockman et al., 2016a): Swimmer and HalfCheetah. We chose these two environments as they allow you to vary the horizon length H without terminating the task early. For both tasks, we use linear policies as they have been shown to be very effective in (Mania et al., 2018; Rajeswaran et al., 2017). Swimmer has an observation space dimensionality of $d = 8$ and a continuous action space of dimensionality $p = 2$. Similarly, for HalfCheetah $d = 17$ and $p = 6$. Figures 3a and 3b show the performance of both approaches in terms of the mean return $J(\theta)$ (expected sum of rewards) they obtain as the horizon length H varies. Note that both approaches are given access to the same number of samples $10^4 \times H$ from the environments for

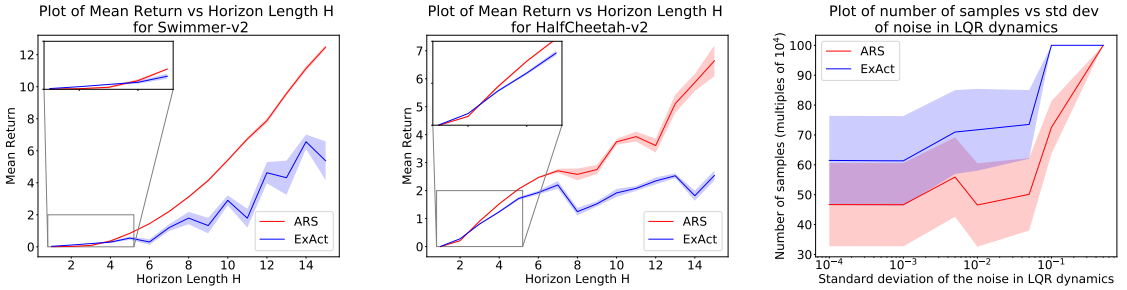
[1] https://github.com/LAIRLAB/contrasting_exploration_rl

[2] https://github.com/LAIRLAB/ARS-experiments



(a) $d = 10$ (b) $d = 100$ (c) $d = 1000$

Figure 2: Linear Regression Experiments with varying input dimensionality



(a) Swimmer (b) HalfCheetah (c) LQR

Figure 3: Multi-step Control. Figures 3a and 3b show performance of different methods as horizon length varies. Figure 3c shows number of samples needed to reach close to a stationary point as noise in dynamics varies

each horizon length H. In the regime of short horizon lengths, action space methods are better than parameter space methods as they do not have a dependence on parametric complexity d. However, as the horizon length increases, parameter space methods start outperforming action space methods handily as they do not have an explicit dependence on the horizon length, as pointed out by our analysis. We have observed the same trend of parameter space methods handily outperforming action space methods as far as $H = 200$ and expect this trend to continue beyond. This empirical insight combined with our analysis presented in Sections 4.1, 4.2 explains why ARS, a simple parameter space search method, outperformed state-of-the-art actor critic action space search methods in (Mania et al., 2018) on OpenAI gym benchmarks where the horizon length H is typically as high as 1000.

Effect of environment stochasticity In this final set of experiments, we set out to understand the effect of stochasticity in environment dynamics on the performance of policy search methods. As our analysis in Sections 4.1 and 4.2 points out, the stochasticity of the environment plays an important role in controlling the variance of our gradient estimates in zeroth order optimization procedures. To empirically observe this, we use a stochastic LQR environment where we have access to the true cost function c and hence, can compute the gradient $\nabla_\theta J(\theta)$ exactly. Given access to such information, we vary the standard deviation σ of the noise in LQR dynamics and observe the number of samples needed for ARS to reach θ such that $\|\nabla_\theta J(\theta)\|_2^2 \leq 0.05$. Figure 3c presents the number of samples needed to reach close to a stationary point of $J(\theta)$ as the standard deviation of noise in LQR dynamics varies. Note that we limit the maximum number of samples to 10^6 for each run. The results match our expectations from the analysis, where we observed that as the stochasticity of the environment increases, convergence rate of policy search methods slows down.

6 Conclusion

Parameter space exploration via black-box optimization methods have often been shown to outperform sophisticated action space exploration approaches for the reinforcement learning problem. Our work highlights the major difference between parameter and action space exploration methods: the latter leverages Jacobian of the parameterized policy. This allows sample complexity of action space exploration methods to be independent of parameter space dimensionality and only dependent on the dimensionality of action space and horizon length. For domains where the action space dimensionality and horizon length are small

and the dimensionality of parameter space is large, we conclude that exploration in action space should be preferred. On the other hand, for long horizon control problems with low dimensional policy parameterization, exploration in parameter space will outperform exploration in action space.

Acknowledgements The authors would like to thank the anonymous reviewers for their useful comments, the entire LairLab for stimulating discussions and Ben Recht for his interesting blog posts.

References

Agarwal, A., Dekel, O., and Xiao, L. (2010). Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, pages 28–40. Citeseer.

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R. (2014). Taming the monster: A fast and simple algorithm for contextual bandits. In *International Conference on Machine Learning*, pages 1638–1646.

Bagnell, J. A., Kakade, S. M., Schneider, J. G., and Ng, A. Y. (2004). Policy search by dynamic programming. In *Advances in neural information processing systems*, pages 831–838.

Bagnell, J. A. and Schneider, J. G. (2001). Autonomous helicopter control using reinforcement learning policy search methods. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1615–1620. IEEE.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016a). Openai gym.

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

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. (2017a). Lower bounds for finding stationary points i. *arXiv preprint arXiv:1710.11606*.

Carmon, Y., Duchi, J. C., Hinder, O., and Sidford, A. (2017b). Lower bounds for finding stationary points ii: First-order methods. *arXiv preprint arXiv:1711.00841*.

Duchi, J. C., Jordan, M. I., Wainwright, M. J., and Wibisono, A. (2015). Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61(5):2788–2806.

Flaxman, A. D., Kalai, A. T., and McMahan, H. B. (2005). Online convex optimization in the bandit setting: gradient descent without a gradient. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 385–394. Society for Industrial and Applied Mathematics.

Ghadimi, S. and Lan, G. (2013). Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368.

Heidrich-Meisner, V. and Igel, C. (2008). Evolution strategies for direct policy search. In *International Conference on Parallel Problem Solving from Nature*, pages 428–437. Springer.

Kakade, S. (2002). A natural policy gradient. *NIPS*.

Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *ICML*.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kober, J., Bagnell, J. A., and Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Mania, H., Guy, A., and Recht, B. (2018). Simple random search provides a competitive approach to reinforcement learning. *arXiv preprint arXiv:1803.07055*.

Mannor, S., Rubinstein, R. Y., and Gat, Y. (2003). The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519.

Nesterov, Y. and Spokoiny, V. (2017). Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566.

Peters, J. and Schaal, S. (2008). Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697.

Rajeswaran, A., Lowrey, K., Todorov, E. V., and Kakade, S. M. (2017). Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561.

Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution strategies as a scal-

able alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. (2015). Trust region policy optimization. In *ICML*, pages 1889–1897.

Sehnke, F., Osendorfer, C., Rückstieß, T., Graves, A., Peters, J., and Schmidhuber, J. (2010). Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559.

Shamir, O. (2013). On the complexity of bandit and derivative-free stochastic convex optimization. In *Conference on Learning Theory*, pages 3–24.

Shamir, O. (2017). An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *Journal of Machine Learning Research*, 18(52):1–11.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge.

Szita, I. and Lörincz, A. (2006). Learning tetris using the noisy cross-entropy method. *Neural computation*, 18(12):2936–2941.

Tesch, M., Schneider, J., and Choset, H. (2011). Using response surfaces and expected improvement to optimize snake robot gait parameters. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1069–1074. IEEE.

Tu, S. and Recht, B. (2018). The gap between model-based and model-free methods on the linear quadratic regulator: An asymptotic viewpoint. *arXiv preprint arXiv:1812.03565*.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Zhao, T., Hachiya, H., Niu, G., and Sugiyama, M. (2011). Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems*, pages 262–270.

A Proof of Theorem 3.1

Proof of Theorem 3.1. To prove Eq. 3 for Alg. 1, we use the proof techniques from Flaxman et al. (2005). The proof is more simpler than the one in Flaxman et al. (2005) as we do not have to deal with shrinking and reshaping the predictor set Θ.

Denote $u \sim \mathbb{B}_b$ as uniformly sampling u from a b-dim unit ball, $u \sim \mathbb{S}_b$ as uniformly sampling u from the b-dim unit sphere, and $\delta \in (0,1)$. Consider the loss function $\hat{c}_i(w_i) = \mathbb{E}_{v\sim\mathbb{B}_b}[c_i(\theta_i + \delta v)]$, which is a smoothed version of $c_i(w_i)$. It is shown in Flaxman et al. (2005) that the gradient of $\hat{c}_i$ with respect to θ is:

$$\begin{aligned}
&\nabla_\theta \hat{c}_i(\theta)|_{\theta=\theta_i} \\
&= \frac{b}{\delta}\mathbb{E}_{u\sim\mathbb{S}_b}[c_i(\theta_i + \delta u)u] \\
&= \frac{b}{\delta}\mathbb{E}_{u\sim\mathbb{S}_b}[((\theta_i + \delta u)^T s_i - a_i)^2 u].
\end{aligned}$$

Hence, the descent direction we take in Alg. 1 is actually an unbiased estimate of $\nabla_\theta \hat{c}_i(\theta)|_{\theta=\theta_i}$. So Alg. 1 can be considered as running OGD with an unbiased estimate of gradient on the sequence of loss $\hat{c}_i(\theta_i)$. It is not hard to show that for an unbiased estimate of $\nabla_\theta \hat{c}_i(\theta)|_{\theta=\theta_i} = \frac{b}{\delta}((\theta_i + \delta u)^T s_i - a_i)^2 u$, the norm is bounded as $b(C^2 + C_s^2)/\delta$. Now we can directly applying Lemma 3.1 from Flaxman et al. (2005), to get:

$$\mathbb{E}\left[\sum_{i=1}^T \hat{c}_i(\theta_i)\right] - \min_{\theta^\star\in\Theta}\sum_{i=1}^T \hat{c}_i(\theta^\star) \leq \frac{C_\theta b(C^2 + C_s^2)}{\delta}\sqrt{T}. \tag{8}$$

We can bound the difference between $\hat{c}_i(\theta)$ and $c_i(\theta)$ using the Lipschitiz continuous property of c_i:

$$\begin{aligned}
|\hat{c}_i(\theta) - c_i(\theta)| &= |\mathbb{E}_{v\sim\mathbb{B}_b}[c_i(\theta + \delta v) - c_i(\theta)]| \\
&\leq \mathbb{E}_{v\sim\mathbb{B}_b}[|c_i(\theta + \delta v) - c_i(\theta)|] \leq L\delta.
\end{aligned} \tag{9}$$

Substitute the above inequality back to Eq. 8, rearrange terms, we get:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^T c_i(\theta_i)\right] - \min_{\theta^\star\in\Theta}\sum_{i=1}^T c_i(w^\star) \\
&\leq \frac{C_\theta b(C^2 + C_s^2)}{\delta}\sqrt{T} + 2LT\delta.
\end{aligned} \tag{10}$$

By setting $\delta = T^{-0.25}\sqrt{\frac{C_\theta b(C^2+C_s^2)}{2L}}$, we get:

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i=1}^T c_i(\theta_i)\right] - \min_{w^\star\in\Theta}\sum_{i=1}^T c_i(w^\star) \\
&\leq \sqrt{C_\theta b(C^2 + C_s^2)L}T^{3/4}.
\end{aligned}$$

To prove Eq. 4 for Alg. 4, we follow the similar strategy in the proof of Alg. 1.

Denote $\epsilon \sim [-1,1]$ as uniformly sampling ϵ from the interval $[-1,1]$, $e \sim \{-1,1\}$ as uniformly sampling e from the set containing -1 and 1. Consider the loss function $\tilde{c}_i(\theta) = \mathbb{E}_{\epsilon\sim[-1,1]}[(\theta^T s_i + \delta\epsilon - a_i)^2]$. One can show that the gradient of $\tilde{c}_i(\theta)$ with respect to θ is:

$$\nabla_\theta \tilde{c}_i(\theta) = \frac{1}{\delta}\mathbb{E}_{e\sim\{-1,1\}}[e(\theta^\top s_i + \delta e - a_i)^2 s_i]. \tag{11}$$

As we can see that the descent direction we take in Alg. 4 is actually an unbiased estimate of $\nabla_\theta \tilde{c}_i(\theta)|_{\theta=\theta_i}$. Hence Alg. 4 can be considered as running OGD with unbiased estimates of gradients on the sequence of loss functions $\tilde{c}_i(\theta)$. For an unbiased estimate of the gradient, $\frac{1}{\delta}e(\theta_i^\top s_i + \delta e - a_i)^2 s_i$, its norm is bounded as $(C^2+1)C_s/\delta$. Note

that different from Alg. 1, here the maximum norm of the unbiased gradient *is independent of feature dimension* b. Now we apply Lemma 3.1 from Flaxman et al. (2005) on $\tilde{c}_i$, to get:

$$\mathbb{E}\left[\sum_{i=1}^{T}\tilde{c}_i(\theta_i)\right] - \min_{\theta^\star\in\Theta}\sum_{i=1}^{T}\tilde{c}_i(\theta^*) \leq \frac{C_\theta(C^2+1)C_s}{\delta}\sqrt{T}. \tag{12}$$

Again we can bound the difference between $\tilde{c}_i(\theta)$ and $c_i(\theta)$ for any θ using the fact that $(\hat{a}_i - a_i)^2$ is Lipschitz continuous with respect to prediction $\hat{a}_i$ with Lipschitz constant C:

$$\begin{aligned}|\tilde{c}_i(\theta) - c_i(\theta)| &= |\mathbb{E}_{\epsilon\sim[-1,1]}[(\theta^\top s_i + \delta\epsilon - a_i)^2 - (\theta^\top s_i - a_i)^2]| \\ &\leq \mathbb{E}_{\epsilon\sim[-1,-1]}[C\delta|\epsilon|] \leq C\delta.\end{aligned} \tag{13}$$

Substitute the above inequality back to Eq. 12, rearrange terms:

$$\begin{aligned}&\mathbb{E}\left[\sum_{i=1}^{T}\tilde{c}_i(\theta_i)\right] - \min_{\theta^\star\in\Theta}\sum_{i=1}^{T}\tilde{c}_i(\theta^*) \\ &\leq \frac{C_\theta(C^2+1)C_s}{\delta}\sqrt{T} + 2C\delta T.\end{aligned}$$

Set $\delta = T^{-0.25}\sqrt{\frac{C_\theta(C^2+1)C_s}{2C}}$, we get:

$$\begin{aligned}&\mathbb{E}\left[\sum_{i=1}^{T}\tilde{c}_i(\theta_i)\right] - \min_{\theta^*\in\Theta}\sum_{i=1}^{T}\tilde{c}_i(\theta^*) \\ &\leq \sqrt{C_\theta(C^2+1)C_sC}T^{3/4}.\end{aligned}$$

□

B Proof of Theorem 4.1

We first present some useful lemmas below.

Consider the smoothed objective given by $\hat{J}(\theta) = \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta+\delta v)]$ where $\mathbb{B}_d$ is the unit ball in d dimensions and δ is a positive constant. Using the assumptions stated in Section 4.1, we obtain the following useful lemma:

Lemma B.1. *If the objective $J(\theta)$ satisfies the assumptions in Section 4.1 and the smoothed objective $\hat{J}(\theta)$ is as given above, then we have that*

1. *$\hat{J}(\theta)$ is also G-Lipschitz and L-smooth*
2. *For all $\theta\in\mathbb{R}^d$, $\|\nabla_\theta J(\theta) - \nabla_\theta\hat{J}(\theta)\| \leq L\delta$*

Proof of Lemma B.1. Consider for any $\theta_1, \theta_2 \in \mathbb{R}^d$,

$$\begin{aligned}|\hat{J}(\theta_1) - \hat{J}(\theta_2)| &= |\mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta_1+\delta v) - J(\theta_2+\delta v)]| \\ &\leq \mathbb{E}_{v\sim\mathbb{B}_d}[|J(\theta_1+\delta v) - J(\theta_2+\delta v)|] \\ &\leq \mathbb{E}_{v\sim\mathbb{B}_d}[G\|\theta_1-\theta_2\|] \\ &= G\|\theta_1-\theta_2\|\end{aligned}$$

The above inequalities are due to the fact that expectation of absolute value is greater than absolute value of expectation, and the G-lipschitz assumption on $J(\theta)$. Thus, the smoothened loss function $\hat{J}(\theta)$ is also G-lipschitz. Similarly consider,

$$\begin{aligned}&\|\nabla_\theta\hat{J}(\theta_1) - \nabla_\theta\hat{J}(\theta_2)\| \\ &\quad= \|\nabla_\theta\mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta_1+\delta v)] - \nabla_\theta\mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta_2+\delta v)]\|\end{aligned}$$

$$
\begin{aligned}
&= \|\mathbb{E}_{v\sim\mathbb{B}_d}[\nabla_\theta J(\theta_1+\delta v) - \nabla_\theta J(\theta_2+\delta v)]\| \\
&\leq \mathbb{E}_{v\sim\mathbb{B}_d}[\|\nabla_\theta J(\theta_1+\delta v) - \nabla_\theta J(\theta_2+\delta v)\|] \\
&\leq \mathbb{E}_{v\sim\mathbb{B}_d}[L\|\theta_1-\theta_2\|] \\
&= L\|\theta_1-\theta_2\|
\end{aligned}
$$

The above inequalities are due to the fact that expectation of norm is greater than norm of expectation, and the L-smoothness assumption on $J(\theta_1)$. We interchange the expectation and derivative using the assumptions on $J(\theta_1)$ and the dominated convergence theorem. Thus, the smoothened loss function $\hat{J}(\theta_1)$ is also L-smooth.

We know,

$$
\begin{aligned}
\nabla_\theta \hat{J}(\theta) &= \nabla_\theta \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta+\delta v)] \\
&= \mathbb{E}_{v\sim\mathbb{B}_d}[\nabla_\theta J(\theta+\delta v)]
\end{aligned}
$$

Note that the expectation and derivative can be interchanged using the dominated convergence theorem. Hence, we have

$$
\begin{aligned}
\|\nabla_\theta \hat{J}(\theta) - \nabla_\theta J(\theta)\| &= \|\mathbb{E}_{u\sim\mathbb{B}_d}[\nabla_\theta J(\theta+\delta v)] - \nabla_\theta J(\theta)\| \\
&\leq \mathbb{E}_{u\sim\mathbb{B}_d}\|\nabla_\theta J(\theta+\delta v) - \nabla_\theta J(\theta)\| \\
&\leq \mathbb{E}_{u\sim\mathbb{B}_d}[L||\delta v||] \\
&\leq L\delta
\end{aligned}
$$

□

The above lemma will be very useful later when we try to relate the convergence rate for the smoothed objective and the true objective. It is shown in (Flaxman et al., 2005; Agarwal et al., 2010) that the gradient estimate g_i is an unbiased estimator of the gradient $\nabla_\theta \hat{J}(\theta_i)$. Hence, Algorithm 3 is performing SGD on the smoothed objective $\hat{J}(\theta)$. Using this insight, we can use the convergence rate of SGD for nonconvex functions to stationary points from (Ghadimi and Lan, 2013) which is given as follows

Lemma B.2 ((Ghadimi and Lan, 2013))**.** *Consider running SGD on the objective $\hat{J}(\theta)$ that is L-smooth and G-Lipschitz for T steps. Fix initial solution θ_0 and denote $\Delta_0 = \hat{J}(\theta_0) - \hat{J}(\theta^*)$ where θ^* is the point at which $\hat{J}(\theta)$ attains global minimum. Also, assume that the gradient estimate g_i is unbiased and has a bounded variance, i.e. for all i, $\mathbb{E}_i[\|g_i - \nabla_\theta \hat{J}(\theta_i)\|_2^2] \leq V \in \mathbb{R}^+$ where $\mathbb{E}_i$ denotes expectation with randomness only at iteration i conditioned on history upto iteration $i-1$. Then we have,*

$$
\frac{1}{T}\sum_{i=1}^{T} \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \frac{2\sqrt{2\Delta_0 L(V+G^2)}}{\sqrt{T}} \tag{14}
$$

For completeness, we include a proof of the above lemma below.

Proof of Lemma B.2. Denote $\xi_i = g_i - \nabla_\theta \hat{J}(\theta_i)$. Note that $\mathbb{E}_i[\xi_i] = 0$ since the stochastic gradient g_i is unbiased. From $\theta_{i+1} = \theta_i - \alpha g_i$, we have:

$$
\begin{aligned}
\hat{J}(\theta_{i+1}) &= \hat{J}(\theta_i - \alpha g_i) \\
&\leq \hat{J}(\theta_i) - \nabla_\theta \hat{J}(\theta_i)^\top(\alpha g_i) + \frac{L\alpha^2}{2}\|g_i\|_2^2 \\
&= \hat{J}(\theta_i) - \alpha\nabla_\theta \hat{J}(\theta_i)^\top g_i + \frac{L\alpha^2}{2}\|\xi_i + \nabla_\theta \hat{J}(\theta_i)\|_2^2 \\
&= \hat{J}(\theta_i) - \alpha\nabla_\theta \hat{J}(\theta_i)^\top g_i + \frac{L\alpha^2}{2}(\|\xi_i\|_2^2 \\
&\quad + 2\xi_i^\top \nabla_\theta \hat{J}(\theta_i) + \|\nabla_\theta \hat{J}(\theta_i)\|_2^2)
\end{aligned}
$$

The first inequality above is obtained since the loss function $\hat{J}(\theta)$ is L-smooth. Adding $\mathbb{E}_i$ on both sides and using the fact that $\mathbb{E}_i[\xi_i] = 0$, we have:

$$
\mathbb{E}_i[\hat{J}(\theta_{i+1})] = \hat{J}(\theta_i) - \alpha\|\nabla_\theta \hat{J}(\theta_i)\|_2^2
$$

$$
\begin{aligned}
&+ \frac{L\alpha^2}{2}\left(\mathbb{E}_i[\|\xi_i\|_2^2] + \|\nabla_\theta \hat{J}(\theta_i)\|_2^2\right) \\
&\leq \hat{J}(\theta_i) - \alpha\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \\
&\quad + \frac{L\alpha^2}{2}\left(\mathbb{E}_i[\|\xi_i\|_2^2] + G^2\right)
\end{aligned}
$$

where the inequality is due to the lipschitz assumption. Rearranging terms, we get:

$$
\begin{aligned}
\alpha\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 &= \hat{J}(\theta_i) - \mathbb{E}_i[\hat{J}(\theta_{i+1})] \\
&\quad + \frac{L\alpha^2}{2}(\mathbb{E}_i[\|\xi_i\|_2^2] + G^2) \\
&\leq \hat{J}(\theta_i) - \mathbb{E}_i[\hat{J}(\theta_{i+1})] + \frac{L\alpha^2}{2}(V + G^2)
\end{aligned}
$$

Sum over from time step 1 to T, we get:

$$
\begin{aligned}
\alpha\sum_{t=1}^{T}\mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 &\leq \mathbb{E}[\hat{J}(\theta_0) - \hat{J}(\theta_T)] \\
&\quad + \frac{LT\alpha^2}{2}(V + G^2)
\end{aligned}
$$

Divide α on both sides, we get:

$$
\begin{aligned}
\sum_{t=1}^{T}\mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 &\leq \frac{1}{\alpha}\mathbb{E}[\hat{J}(\theta_0) - \hat{J}(\theta_T)] + LT\alpha(V + G^2) \\
&\leq \frac{1}{\alpha}\mathbb{E}[\hat{J}(\theta_0) - \hat{J}(\theta^*)] + LT\alpha(V + G^2) \\
&= \frac{1}{\alpha}\Delta_0 + LT\alpha(V + G^2) \\
&\leq \sqrt{\frac{\Delta_0 LT(V + G^2)}{2}} + \sqrt{2\Delta_0 LT(V + G^2)} \\
&\leq 2\sqrt{2\Delta_0 LT(V + G^2)}
\end{aligned}
$$

with $\alpha = \sqrt{\frac{2\Delta_0}{LT(V+G^2)}}$. Hence, we have:

$$
\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \leq \frac{2\sqrt{2\Delta_0 L(V + G^2)}}{\sqrt{T}}
$$

□

The above lemma is useful as it gives us the following result:

$$
\begin{aligned}
\min_{1\leq i\leq T}\mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 &\leq \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2 \\
&\leq \frac{2\sqrt{2\Delta_0 L(V + G^2)}}{\sqrt{T}} \qquad (15)
\end{aligned}
$$

since the minimum is always less than the average. We have then that using SGD to minimize a nonconvex objective finds a θ_i that is 'almost' a stationary point in bounded number of steps provided the stochastic gradient estimate has bounded variance.

We now show that the gradient estimate g_i used in Algorithm 3 indeed has a bounded variance. Observe that the estimate g_i in the algorithm is a two-point estimate, which should have substantially less variance than one-point estimates (Agarwal et al., 2010). However, the two evaluations, resulting in J_i^+ and J_i^-, have different independent noise. This is due to the fact that in policy search, stochasticity arises from the environment and cannot be controlled and we cannot obtain the significant variance reduction that is typical of two-point estimators. The following lemma quantifies the bound on the variance of gradient estimate g_i:

Lemma B.3. *Consider a smoothed objective $\hat{J}(\theta) = \mathbb{E}_{v\sim\mathbb{B}_d}[J(\theta+\delta v)]$ where $\mathbb{B}_d$ is the unit ball in d dimensions, $\delta > 0$ is a scalar and the true objective $J(\theta)$ is G-lipschitz. Given gradient estimate $g_i = \frac{d(J_i^+ - J_i^-)}{2\delta}u$ where u is sampled uniformly from a unit sphere $\mathbb{S}_d$ in d dimensions, $J_i^+ = J(\theta_i + \delta u) + \eta_i^+$ and $J_i^- = J(\theta - \delta u) + \eta_i^-$ for zero mean random i.i.d noises η_i^+, η_i^-, we have*

$$\mathbb{E}_i[\|g_i - \nabla_\theta \hat{J}(\theta_i)\|_2^2] \leq 2d^2G^2 + 2\frac{d^2\sigma^2}{\delta^2} \tag{16}$$

where σ^2 is the variance of the random noise η.

Proof of Lemma B.3. From Shamir (2017), we know that g_i is an unbiased estimate of the gradient of $\hat{J}(\theta_i)$, i.e. $\mathbb{E}_{u_i\sim\mathbb{S}_d}[g_i] = \nabla\hat{J}(\theta_i)$. Thus, we have

$$\begin{aligned}
&\mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i - \nabla\hat{J}(\theta_i)\|^2 \\
&\quad= \mathbb{E}_{u_i\sim\mathbb{S}_d}[\|g_i\|^2 + \|\nabla\hat{J}(\theta)_i\|^2 - 2g_i^T\nabla\hat{J}(\theta_i)] \\
&\quad= \mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i\|^2 + \|\nabla\hat{J}(\theta_i)\|^2 - 2\|\nabla\hat{J}(\theta_i)\|^2 \\
&\quad= \mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i\|^2 - \|\nabla\hat{J}(\theta_i)\|^2 \\
&\quad\leq \mathbb{E}_{u_i\sim\mathbb{S}_d}\|g_i\|^2 \\
&\quad= \frac{d^2}{4\delta^2}\mathbb{E}_{u_i\sim\mathbb{S}_d}\|(J(\theta_i+\delta u_i) - J(\theta_i - \delta u_i) \\
&\qquad+ (\eta_i^+ - \eta_i^-))u_i\|^2 \\
&\quad\leq \frac{d^2}{2\delta^2}[\mathbb{E}_{u_i\sim\mathbb{S}_d}\|(J(\theta_i+\delta u_i) - J(\theta_i - \delta u_i)u_i\|_2^2 \\
&\qquad+ \mathbb{E}_{u_i\sim\mathbb{S}_d}\|(\eta_i^+ - \eta_i^-))u_i\|^2] \\
&\quad\leq \frac{d^2}{2\delta^2}[\mathbb{E}_{u_i\sim\mathbb{S}_d}4G^2\delta^2\|u_i\|^2 + 4\mathbb{E}_{u_i\sim\mathbb{S}_d}\|\eta_i^+\|_2^2\|u_i\|_2^2] \\
&\quad= 2d^2G^2 + 2\frac{d^2\sigma^2}{\delta^2}
\end{aligned}$$

where the second inequality is true as $\|a+b\|_2^2 \leq 2(\|a\|_2^2 + \|b\|_2^2)$ and the last inequality is due to the Lipschitz assumption on $J(\theta)$. □

We are ready to prove Theorem 4.1.

Proof of Theorem 4.1. Fix initial solution θ_0 and denote $\Delta_0 = \hat{J}(\theta_0) - \hat{J}(\theta^*)$ where $\hat{J}(\theta)$ is the smoothed objective and θ^* is the point at which $\hat{J}(\theta)$ attains global minimum. Since the gradient estimate g_i used in Algorithm 3 is an unbiased estimate of the gradient $\nabla_\theta\hat{J}(\theta_i)$, we know that Algorithm 3 performs SGD on the smoothed objective. Moreover, from Lemma B.3, we know that the variance of the gradient estimate g_i is bounded. Hence, we can use Lemma B.2 on the smoothed objective $\hat{J}(\theta)$ to get

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta\hat{J}(\theta_i)\|_2^2 \leq \frac{2\sqrt{2\Delta_0 L(V+G^2)}}{\sqrt{T}} \tag{17}$$

where $V \leq 2d^2G^2 + 2\frac{d^2\sigma^2}{\delta^2}$ (from Lemma B.3). We can relate $\nabla_\theta\hat{J}(\theta)$ and $\nabla_\theta J(\theta)$ - the quantity that we ultimately care about, as follows:

$$\begin{aligned}
&\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \\
&\quad= \frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i) - \nabla_\theta\hat{J}(\theta_i) + \nabla_\theta\hat{J}(\theta_i)\|_2^2
\end{aligned}$$

$$\leq \frac{2}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i) - \nabla_\theta \hat{J}(\theta_i)\|_2^2 + \mathbb{E}\|\nabla_\theta \hat{J}(\theta_i)\|_2^2$$

We can use Lemma B.1 to bound the first term and Equation 17 to bound the second term. Thus, we have

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \frac{2}{T}[TL^2\delta^2 + 2\sqrt{2\Delta_0 L(V+G^2)T}]$$

Substituting the bound for V from Lemma B.3, using the inequality $\sqrt{a+b} \leq \sqrt{a}+\sqrt{b}$ for $a, b \in \mathbb{R}^+$, optimizing over δ, and using $\Delta_0 \leq \mathcal{Q}$ we get

$$\frac{1}{T}\sum_{i=1}^{T}\mathbb{E}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(\mathcal{Q}^{\frac{1}{2}} d T^{\frac{-1}{2}} + \mathcal{Q}^{\frac{1}{3}} d^{\frac{2}{3}} T^{\frac{-1}{3}} \sigma)$$

□

C Proof of Theorem

The bound on the bias of the gradient estimate is given by the following lemma:

Lemma C.1. *If the assumptions in Section 4.2 are satisfied, then for the gradient estimate g_i used in Algorithm 4 and the gradient of the objective $J(\theta)$ given in equation 6, we have*

$$\|\mathbb{E}[g_i] - \nabla_\theta J(\theta_i)\| \leq KUH\delta \tag{18}$$

Proof of Lemma C.1. To prove that the bias is bounded, let's consider for any i

$$\begin{aligned}
&\|\mathbb{E}[g_i] - \nabla_\theta J(\theta_i)\|_2 \\
&= \| \sum_{t=0}^{H-1} \mathbb{E}_{s_t\sim d^t_{\pi_{\theta_i}}}[\nabla_\theta \pi(\theta_i, s_t) \\
&\nabla_a (\mathbb{E}_{v\sim\mathbb{B}_p} Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v) - Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t)))]\|_2 \\
&\leq \sum_{t=0}^{H-1} \mathbb{E}_{s_t\sim d^t_{\pi_{\theta_i}}, v\sim\mathbb{B}_p} \|\nabla_\theta \pi(\theta_i, s_t)\|_2 \\
&\quad \|[\nabla_a Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta v) - \nabla_a Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t))]\|_2 \\
&\leq \sum_{t=0}^{H-1} KU\delta \mathbb{E}_{v\sim\mathbb{B}_p}\|v\|_2 \\
&\leq KUH\delta
\end{aligned}$$

The first inequality above is obtained by using the fact that $\|\mathbb{E}[X]\|_2 \leq \mathbb{E}\|X\|_2$, and the second inequality using the K-lipschitz assumption on $\pi(\theta, s)$ and U-smooth assumption on $Q^t_{\pi_\theta}(s, a)$ in a. Also, observe that we interchanged the derivative and expectation above by using the assumptions on $Q^t_{\pi_\theta}$ as stated in Section 4.2. □

We will now show that the gradient estimate g_i used in Algorithm 4 has a bounded variance. Note that the gradient estimate constructed in Algorithm 4 is a one-point estimate, unlike policy search in parameter space where we had a two-point estimate. Thus, the variance would be higher and the bound on the variance of such a one-point estimate is given below

Lemma C.2. *Given a gradient estimate g_i as shown in Algorithm 4, the variance of the estimate can be bounded as*

$$\mathbb{E}\|g_i - \mathbb{E}[g_i]\|_2^2 \leq \frac{2H^2p^2K^2}{\delta^2}((\mathcal{Q} + W\delta)^2 + \sigma^2) \tag{19}$$

where σ^2 is the variance of the random noise $\tilde{\eta}$.

Proof of Lemma C.2. To bound the variance of the gradient estimate g_i in Algorithm 4, lets consider

$$\begin{aligned}
&\mathbb{E}_i\|g_i - \mathbb{E}[g_i]\|_2^2 = \mathbb{E}_i\|g_i\|_2^2 - \|\mathbb{E}_i[g_i]\|_2^2 \leq \mathbb{E}_i\|g_i\|_2^2 \\
&= \frac{H^2p^2}{\delta^2}\mathbb{E}_i\|\nabla_\theta\pi(\theta_i, s_t)(Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u) + \tilde{\eta}_i)u\|_2^2 \\
&\leq \frac{K^2p^2H^2}{\delta^2}\mathbb{E}_i\|Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t) + \delta u)u + \tilde{\eta}_i u\|_2^2
\end{aligned}$$

where $\mathbb{E}_i$ denotes expectation with respect to the randomness at iteration i and the inequality is obtained using K-lipschitz assumption on $\pi(\theta, s)$. Note that we can express $Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t)+\delta u) \leq Q^t_{\pi_{\theta_i}}(s_t, \pi(\theta_i, s_t)) + W\delta\|u\|_2 \leq \mathcal{Q}+W\delta$ where we used the W-lipschitz assumption on $Q^t_{\pi_\theta}(s, a)$ in a and that it is bounded everywhere by constant $\mathcal{Q}$. Thus, we have

$$\begin{aligned}
&\mathbb{E}_i\|g_i - \mathbb{E}[g_i]\|_2^2 \\
&\leq \frac{K^2p^2H^2}{\delta^2}\mathbb{E}_i\|(\mathcal{Q} + W\delta)u + \tilde{\eta}_i u\|_2^2 \\
&\leq \frac{2K^2p^2H^2}{\delta^2}(\mathbb{E}_i\|(\mathcal{Q} + W\delta)u\|_2^2 + \mathbb{E}_i\|\tilde{\eta}_i u\|_2^2 \\
&\leq \frac{2K^2p^2H^2}{\delta^2}((\mathcal{Q} + W\delta)^2 + \sigma^2)
\end{aligned}$$

□

We are now ready to prove theorem 4.2

Proof of Theorem 4.2. Fix initial solution θ_0 and denote $\Delta_0 = J(\theta_0) - J(\theta^*)$ where θ^* is the point at which $J(\theta)$ attains global minimum. Denote $\xi_i = g_i - \mathbb{E}_i[g_i]$ and $\beta_i = \mathbb{E}_i[g_i] - \nabla_\theta J(\theta_i)$. From Lemma C.1, we know $\|\beta_i\| \leq KUH\delta$ and from lemma C.2, we know $\mathbb{E}\|\xi_i\|_2^2 = V \leq \frac{2K^2p^2H^2}{\delta^2}((\mathcal{Q} + W\delta)^2 + \sigma^2)$ and $\mathbb{E}_i[\xi_i] = 0$ from definition. From $\theta_{i+1} = \theta_i - \alpha g_i$ we have:

$$\begin{aligned}
J(\theta_{i+1}) &= J(\theta_i - \alpha g_i) \\
&\leq J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T g_i + \frac{L\alpha^2}{2}\|g_i\|_2^2 \\
&= J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T g_i + \frac{L\alpha^2}{2}\|\xi_i + \mathbb{E}_i[g_i]\|_2^2 \\
&= J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T g_i \\
&\quad + \frac{L\alpha^2}{2}(\|\mathbb{E}_i[g_i]\|_2^2 + \|\xi_i\|_2^2 + 2\mathbb{E}_i[g_i]^T\xi_i)
\end{aligned}$$

Taking expectation on both sides with respect to randomness at iteration i, we have

$$\begin{aligned}
\mathbb{E}_i[J(\theta_{i+1})] &= J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T\mathbb{E}_i[g_i] \\
&\quad + \frac{L\alpha^2}{2}(\|\mathbb{E}_i[g_i]\|_2^2 + \mathbb{E}_i\|\xi_i\|_2^2 + 2\mathbb{E}_i[g_i]^T\mathbb{E}_i[\xi_i]) \\
&\leq J(\theta_i) - \alpha\nabla_\theta J(\theta_i)^T(\beta_i + \nabla_\theta J(\theta_i)) \\
&\quad + \frac{L\alpha^2}{2}(\|\beta_i + \nabla_\theta J(\theta_i)\|_2^2 + V) \\
&= J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(\|\nabla_\theta J(\theta_i)\|_2^2 + V + \|\beta_i\|_2^2) \\
&\quad + (L\alpha^2 - \alpha)\nabla_\theta J(\theta_i)^T\beta_i \\
&\leq J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) \\
&\quad + (L\alpha^2 - \alpha)\nabla_\theta J(\theta_i)^T\beta_i
\end{aligned}$$

$$
\begin{aligned}
&\leq J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) \\
&\quad + (L\alpha^2 + \alpha)\|\nabla_\theta J(\theta_i)\|\|\beta_i\| \\
&\leq J(\theta_i) - \alpha\|\nabla_\theta J(\theta_i)\|_2^2 + \frac{L\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) \\
&\quad + (L\alpha^2 + \alpha)GKUH\delta
\end{aligned}
$$

Rearranging terms and summing over timestep 1 to T, we get

$$
\begin{aligned}
\alpha\sum_{i=1}^{T}\|\nabla_\theta J(\theta_i)\|_2^2 &\leq J(\theta_0) - \mathbb{E}_T[J(\theta_T)] \\
&\quad + \frac{LT\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) + (L\alpha^2 + \alpha)GKUHT\delta \\
&\leq \Delta_0 + \frac{LT\alpha^2}{2}(G^2 + V + K^2H^2U^2\delta^2) \\
&\quad + (L\alpha^2 + \alpha)GKUHT\delta \\
\sum_{i=1}^{T}\|\nabla_\theta J(\theta_i)\|_2^2 &\leq \frac{\Delta_0}{\alpha} + \frac{LT\alpha}{2}(G^2 + V + K^2H^2U^2\delta^2) \\
&\quad + (L\alpha + 1)GKUHT\delta \\
&\leq \frac{\Delta_0}{\alpha} + \frac{LT\alpha}{2}(G^2 + K^2H^2U^2\delta^2 + 2GKUH\delta) \\
&\quad + GKUHT\delta + \frac{LT\alpha}{2}V \\
&\leq \frac{\Delta_0}{\alpha} + \frac{LT\alpha}{2}(G + KHU\delta)^2 \\
&\quad + GKUHT\delta + \frac{LT\alpha K^2p^2H^2}{\delta^2}((\mathcal{Q} + W\delta)^2 + \sigma^2) \\
&\leq \frac{\Delta_0}{\alpha} + LT\alpha(G^2 + K^2H^2U^2\delta^2) \\
&\quad + GKUHT\delta + 2\frac{LT\alpha K^2p^2H^2}{\delta^2}(\mathcal{Q}^2 + W^2\delta^2 + \sigma^2)
\end{aligned}
$$

Using $\Delta_0 \leq \mathcal{Q}$ and optimizing over α and δ, we get $\alpha = \mathcal{O}(\mathcal{Q}^{\frac{3}{4}}T^{-\frac{3}{4}}H^{-1}p^{-\frac{1}{2}}(\mathcal{Q}^2+\sigma^2)^{-\frac{1}{4}})$ and $\delta = \mathcal{O}(T^{-\frac{1}{4}}p^{\frac{1}{2}}(\mathcal{Q}^2+\sigma^2)^{\frac{1}{4}})$. This gives us

$$
\frac{1}{T}\sum_{i=1}^{T}\|\nabla_\theta J(\theta_i)\|_2^2 \leq \mathcal{O}(T^{-\frac{1}{4}}Hp^{\frac{1}{2}}(\mathcal{Q}^3 + \sigma^2\mathcal{Q})^{\frac{1}{4}}) \tag{20}
$$

□

D Implementation Details

D.1 One-step Control Experiments

D.1.1 Tuning Hyperparameters for ARS

We tune the hyperparameters for ARS (Mania et al., 2018) in both MNIST and linear regression experiments, by choosing a candidate set of values for each hyperparameter: stepsize, number of directions sampled, number of top directions chosen and the perturbation length along each direction. The candidate hyperparameter values are shown in Table 1.

We use the hyperparameters shown in Table 2 chosen through this tuning for each of the experiments in this work. The hyperparameters are chosen by averaging the test squared loss across three random seeds (different from the 10 random seeds used in actual experiments) and chosing the setting that has the least mean test squared loss after 100000 samples.

Hyperparameter	Candidate Values
Stepsize	$0.001, 0.005, 0.01, 0.02, 0.03$
# Directions	$10, 50, 100, 200, 500$
# Top Directions	$5, 10, 50, 100, 200$
Perturbation	$0.001, 0.005, 0.01, 0.02, 0.03$

Table 1: Candidate hyperparameters used for tuning in ARS experiments

Experiment	Stepsize	# Dir.	# Top Dir.	Perturbation
MNIST	0.02	50	20	0.03
LR $d = 10$	0.03	10	10	0.03
LR $d = 100$	0.03	10	10	0.02
LR $d = 1000$	0.03	200	200	0.03

Table 2: Hyperparameters chosen for ARS in each experiment. LR is short-hand for Linear Regression.

D.1.2 MNIST Experiments

The CNN architecture used is as shown in Figure 4[3]. The total number of parameters in this model is $d = 21840$. For supervised learning, we use a cross-entropy loss on the softmax output with respect to the true label. To train this model, we use a batch size of 64 and a stochastic gradient descent (SGD) optimizer with learning rate of 0.01 and a momentum factor of 0.5. We evaluate the test accuracy of the model over all the 10000 images in the MNIST test dataset.

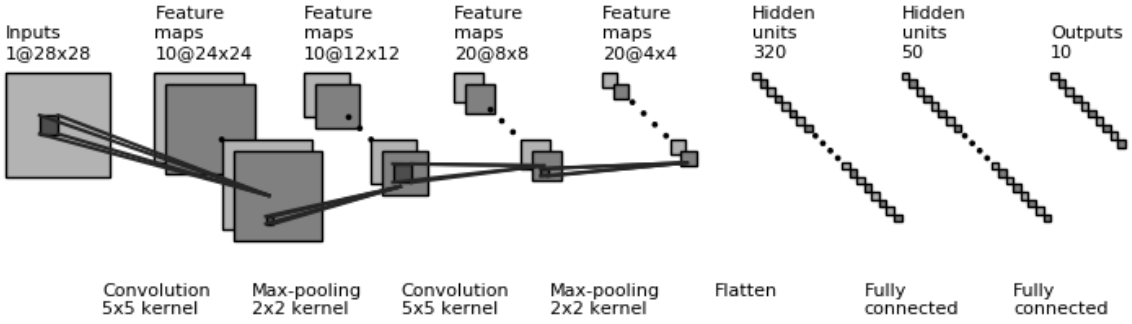


Figure 4: CNN architecture used for the MNIST experiments

For REINFORCE, we use the same architecture as before. We train the model by sampling from the categorical distribution parameterized by the softmax output of the model and then computing a ± 1 reward based on whether the model predicted the correct label. The loss function is the REINFORCE loss function given by,

$$J(\theta) = \frac{1}{N} \sum_{i=1}^{N} r_i \log(\mathbb{P}(\hat{y}_i | x_i, \theta)) \tag{21}$$

where θ is the parameters of the model, r_i is the reward obtained for example i, $\hat{y}_i$ is the predicted label for example i and x_i is the input feature vector for example i. The reward r_i is given by $r_i = 2 * \mathbb{I}[\hat{y}_i = y_i] - 1$, where $\mathbb{I}$ is the $0 - 1$ indicator function and y_i is the true label for example i.

For ARS, we use the same architecture and reward function as before. The hyperparameters used are shown in Table 2 and we closely follow the algorithm outlined in (Mania et al., 2018).

[3]This figure is generated by adapting the code from `https://github.com/gwding/draw_convnet`

Experiment	Learning Rate	Batch size
MNIST	0.001	512
LR $d = 10$	0.08	512
LR $d = 100$	0.03	512
LR $d = 1000$	0.01	512

Table 3: Learning rate and batch size used for REINFORCE experiments. We use an ADAM (Kingma and Ba, 2014) optimizer for these experiments.

Experiment	Learning Rate	Batch size
LR $d = 10$	2.0	512
LR $d = 100$	2.0	512

Table 4: Learning rate and batch size used for Natural REINFORCE experiments. Note that we decay the learning rate after each batch by $\sqrt{T}$ where T is the number of batches seen.

D.1.3 Linear Regression Experiments

We generate training and test data for the linear regression experiments as follows: we sampled a random $d+1$ dimensional vector w where d is the input dimensionality. We also sampled a random $d \times d$ covariance matrix C. The training and test dataset consists of $d+1$ vectors x whose first element is always 1 (for the bias term) and the rest of the d terms are sampled from a multivariate normal distribution with mean $\mathbf{0}$ and covariance matrix C. The target vectors y are computed as $y = w^T x + \epsilon$ where ϵ is sampled from a univariate normal distribution with mean 0 and standard deviation 0.001.

We implemented both SGD and Newton Descent on the mean squared loss, for the supervised learning experiments. For SGD, we used a learning rate of 0.1 for $d = 10, 100$ and a learning rate of 0.01 for $d = 1000$, and a batch size of 64. For Newton Descent, we also used a batch size of 64. To frame it as a one-step MDP, we define a reward function r which is equal to the negative of mean squared loss. Both REINFORCE and ARS use this reward function. To compute the REINFORCE loss, we take the prediction of the model $\hat{w}^T x$, add a mean 0 standard deviation $\beta = 0.5$ Gaussian noise to it, and compute the reward (negative mean squared loss) for the noise added prediction. The REINFORCE loss function is then given by

$$J(w) = \frac{1}{N} \sum_{i=1}^{N} r_i \frac{-(y_i - \hat{w}^T x_i)^2}{2\beta^2} \tag{22}$$

where $r_i = -(y_i - \hat{y}_i)^2$, $\hat{y}_i$ is the noise added prediction and $\hat{w}^T x_i$ is the prediction by the model. We use an Adam optimizer with learning rate and batch size as shown in Table 3. For the natural REINFORCE experiments, we estimate the fisher information matrix and compute the descent direction by solving the linear system of equations $Fx = g$ where F is the fisher information matrix and g is the REINFORCE gradient. We use SGD with a $O(1/\sqrt{T})$ learning rate, where T is the number of batches seen, and batch size as shown in Table 4.

For ARS, we closely follow the algorithm outlined in (Mania et al., 2018).

D.2 Multi-step Control Experiments

D.2.1 Tuning Hyperparameters for ARS

We tune the hyperparameters for ARS (Mania et al., 2018) in both mujoco and LQR experiments, similar to the one-step control experiments. The candidate hyperparameter values are shown in Tables 5 and 6. We have observed that using all the directions in ARS is always preferable under the low horizon settings that we explore. Hence, we do not conduct a hyperparameter search over the number of top directions and instead keep it the same as the number of directions.

We use the hyperparameters shown in Tables 7 and 8 chosen through tuning for each of the multi-step experiments. The hyperparameters are chosen by averaging the total reward obtained across three random seeds (different from the 10 random seeds used in experiments presented in Figures 3a, 3b, 3c) and chosing the setting that has the highest total reward after 10000 episodes of training..

Hyperparameter	Swimmer-v2	HalfCheetah-v2
Stepsize	$0.03, 0.05, 0.08, 0.1, 0.15$	$0.001, 0.003, 0.005, 0.008, 0.01$
# Directions	$5, 10, 20$	$5, 10, 20$
Perturbation	$0.05, 0.1, 0.15, 0.2$	$0.01, 0.03, 0.05, 0.08$

Table 5: Candidate hyperparameters used for tuning in ARS experiments

Hyperparameter	LQR
Stepsize	$0.0001, 0.0003, 0.0005, 0.0008, 0.001, 0.003, 0.005, 0.008, 0.01$
# Directions	10
Perturbation	$0.01, 0.05, 0.1$

Table 6: Candidate hyperparameters used for tuning in ARS experiments

D.2.2 Tuning Hyperparameters for ExAct

We tune the hyperparameters for ExAct (Algorithm 4) in both mujoco and LQR experiments, similar to ARS. The candidate hyperparameter values are shown in Tables 9 and 10. Similar to ARS, we do not conduct a hyperparameter search over the number of top directions and instead keep it the same as the number of directions.

Hyperparameter	Swimmer-v2	HalfCheetah-v2
Stepsize	$0.005, 0.008, 0.01, 0.015, 0.02, 0.025, 0.03$	$0.0001, 0.0003, 0.0005, 0.0008, 0.001, 0.002, 0.003$
# Directions	$5, 10, 20$	$5, 10, 20$
Perturbation	$0.15, 0.2, 0.3, 0.5$	$0.15, 0.2, 0.3, 0.5$

Table 9: Candidate hyperparameters used for tuning in ExAct experiments

We use the hyperparameters shown in Tables 11 and 12 chosen through tuning for each of the multi-step experiments, similar to ARS.

D.2.3 Mujoco Experiments

For all the mujoco experiments, both ARS and ExAct use a linear policy with the same number of parameters as the dimensionality of the state space. The hyperparameters for both algorithms are chosen as described above. Each algorithm is run on both environments (Swimmer-v2 and HalfCheetah-v2) for 10000 episodes of training across 10 random seeds (different from the ones used for tuning). This is repeated for each horizon value $H \in \{1, 2, \cdots, 15\}$. In each experiment, we record the mean evaluation return obtained after training and plot the results in Figures 3a, 3b. For more details on the environments used, we refer the reader to (Brockman et al., 2016b).

D.2.4 LQR Experiments

In the LQR experiments, we constructed a linear dynamical system $x_{t+1} = Ax_t + Bu_t + \xi_t$ where $x_t \in \mathbb{R}^{100}$, $A \in \mathbb{R}^{100\times100}$, $B \in \mathbb{R}^{100}$, $u_t \in \mathbb{R}$ and the noise $\xi_t \sim \mathcal{N}(0_{100}, cI_{100\times100})$ with a small constant $c \in \mathbb{R}^+$. We explicitly make sure that the maximum eigenvalue of A is less than 1 to avoid instability. We fix a quadratic cost function $c(x, u) = x^TQx + uRu$, where $Q = 10^{-3}I_{100\times100}$ and $R = 1$. The hyperparameters chosen for both algorithms are chosen as described above.

For each algorithm, we run it for noise covariance values $c \in \{10^{-4}, 5\times10^{-4}, 10^{-3}, 5\times10^{-3}, 10^{-2}, 5\times10^{-2}, 10^{-1}, 5\times10^{-1}\}$ until we reach a stationary point where $\|\nabla_\theta J(\theta)\|_2^2 \leq 0.05$. The number of interactions with the environment allowed is capped at 10^6 steps for each run. This is repeated across 10 random seeds (different from the ones used for tuning). The number of interactions needed to reach the stationary point as the noise covariance is increased is recorded and shown in Figure 3c.

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.15	5	0.2
$H = 2$	0.08	5	0.2
$H = 3$	0.15	5	0.2
$H = 4$	0.08	5	0.2
$H = 5$	0.05	5	0.2
$H = 6$	0.08	5	0.2
$H = 7$	0.08	5	0.2
$H = 8$	0.08	5	0.2
$H = 9$	0.1	5	0.2
$H = 10$	0.08	5	0.2
$H = 11$	0.08	5	0.2
$H = 12$	0.1	5	0.2
$H = 13$	0.08	5	0.2
$H = 14$	0.08	5	0.2
$H = 15$	0.08	10	0.2

Table 7: Hyperparameters chosen for multi-step experiments for ARS in Swimmer-v2

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.001	20	0.08
$H = 2$	0.008	5	0.08
$H = 3$	0.008	10	0.08
$H = 4$	0.003	5	0.05
$H = 5$	0.003	5	0.05
$H = 6$	0.003	10	0.05
$H = 7$	0.008	20	0.05
$H = 8$	0.008	5	0.05
$H = 9$	0.01	20	0.03
$H = 10$	0.005	10	0.03
$H = 11$	0.008	20	0.03
$H = 12$	0.005	5	0.05
$H = 13$	0.008	20	0.03
$H = 14$	0.01	10	0.03
$H = 15$	0.008	20	0.03

Table 8: Hyperparameters chosen for multi-step experiments for ARS in HalfCheetah-v2

Hyperparameter	LQR
Stepsize	$0.0001, 0.0003, 0.0005, 0.0008, 0.001, 0.003, 0.005, 0.008, 0.01$
# Directions	10
Perturbation	$0.01, 0.05, 0.1$

Table 10: Candidate hyperparameters used for tuning in ExAct experiments

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.02	5	0.2
$H = 2$	0.02	5	0.2
$H = 3$	0.015	10	0.2
$H = 4$	0.015	10	0.2
$H = 5$	0.01	10	0.2
$H = 6$	0.015	10	0.2
$H = 7$	0.01	20	0.2
$H = 8$	0.015	20	0.2
$H = 9$	0.02	20	0.2
$H = 10$	0.008	5	0.2
$H = 11$	0.02	5	0.15
$H = 12$	0.02	20	0.2
$H = 13$	0.015	5	0.15
$H = 14$	0.02	10	0.15
$H = 15$	0.01	5	0.1

Table 11: Hyperparameters chosen for multi-step experiments for ExAct in Swimmer-v2

Horizon	Stepsize	# Directions	Perturbation
$H = 1$	0.0001	20	0.2
$H = 2$	0.001	5	0.2
$H = 3$	0.001	5	0.2
$H = 4$	0.001	5	0.2
$H = 5$	0.001	10	0.2
$H = 6$	0.001	5	0.2
$H = 7$	0.001	10	0.2
$H = 8$	0.001	5	0.2
$H = 9$	0.001	5	0.2
$H = 10$	0.001	5	0.2
$H = 11$	0.0008	5	0.15
$H = 12$	0.001	5	0.2
$H = 13$	0.001	10	0.2
$H = 14$	0.001	5	0.2
$H = 15$	0.0008	10	0.2

Table 12: Hyperparameters chosen for multi-step experiments for ExAct in HalfCheetah-v2