Understanding the impact of entropy on policy optimization

Zafarali Ahmed[▷][1]   Nicolas Le Roux[◁]
Mohammad Norouzi[◁]   Dale Schuurmans[◁]

[▷]Mila, McGill University   [◁]Google Brain

zafarali.ahmed@mail.mcgill.ca, {nlr, mnorouzi, schuurmans}@google.com

Abstract

Entropy regularization is commonly used to improve policy optimization in reinforcement learning. It is believed to help with *exploration* by encouraging the selection of more stochastic policies. In this work, we analyze this claim and, through new visualizations of the optimization landscape, we observe that incorporating entropy in policy optimization serves as a regularizer. We show that even with access to the exact gradient, policy optimization is difficult due to the geometry of the objective function. We qualitatively show that, in some environments, entropy regularization can make the optimization landscape smoother, thereby connecting local optima and enabling the use of larger learning rates. This paper presents new tools for understanding the underlying optimization landscape and highlights the challenge of designing general-purpose policy optimization algorithms in reinforcement learning.

Keywords: Policy gradient, Entropy regularization, Optimization landscape

1 Introduction

Policy optimization is a family of reinforcement learning (RL) algorithms aiming to directly optimize the parameters of a policy by maximizing discounted cumulative rewards. This often involves a difficult non-concave maximization problem, even when using a simple policy with a linear state-action mapping.

Contemporary policy optimization algorithms build upon the REINFORCE algorithm (Williams, 1992). These algorithms involve estimating a noisy gradient of the optimization objective using Monte-Carlo sampling to enable stochastic gradient ascent. As such high variance in gradient estimates is often seen as a major issue for which several solutions have been proposed (Konda & Tsitsiklis, 2000; Greensmith et al., 2004; Schulman et al., 2015b; Tucker et al., 2018). However, in this work we show that noisy estimates of the gradient are not necessarily the *main* issue: The optimization problem is difficult because of the geometry of the landscape. Given that "high variance" is often the reason given for the poor performance of policy optimization, it raises an important question:

How do we study the effects of different policy learning techniques on the underlying optimization problem?

An answer to this question would guide future research directions and drive the design of new policy optimization techniques. Our work makes progress toward this goal by taking a look at one such technique: *entropy regularization.*

In RL exploration is critical to finding good solutions during optimization: If the optimization procedure does not sample a large and diverse number of state-action pairs, it may converge to a poor solution. To prevent policies from becoming deterministic too quickly, researchers use entropy regularization (Williams & Peng, 1991; Mnih et al., 2016). Its success has sometimes been attributed to the fact that it "encourages exploration" (Mnih et al., 2016; Schulman et al., 2017a,b). Contrary to Q-learning (Watkins & Dayan, 1992) or Deterministic Policy Gradient (Silver et al., 2014) where the exploration is handled separately from the policy itself, direct policy optimization relies on the stochasticity of the policy being optimized for the exploration. However, policy optimization is a pure maximization problem that is frequently solved using stochastic estimates of the gradients. Hence, any strategy, such as entropy regularization, can only affect learning in one of two ways: either it reduces the noise in the gradient estimates or it changes the optimization landscape.

In this work we investigate some of these questions by controlling the stochasticity of policies and observing its effect on the geometry of the optimization landscape. Our work makes the following contributions:

- Experiments showing that the difficulty of policy optimization is more strongly linked to the geometry of the objective function than poor estimates of the gradient.

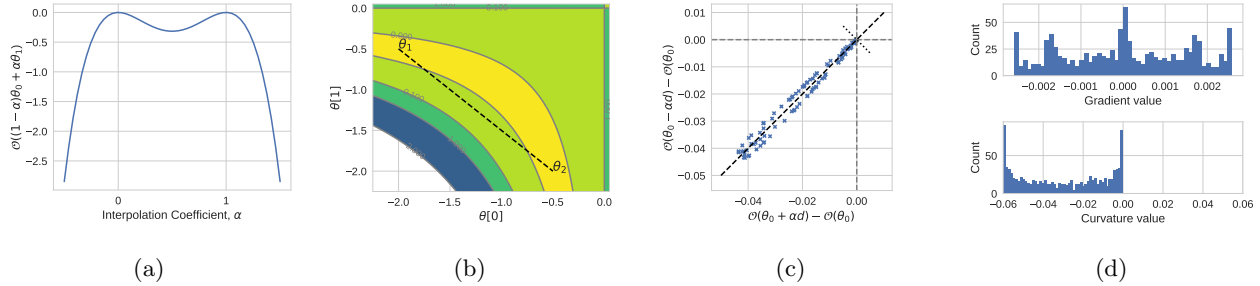[1]Work done during an internship at Google Brain

Figure 1: **Summary of the methods on** $\mathcal{O}(\theta) = -(1 - \theta[0]\theta[1])^2$ (a) A linear interpolation between two local optima, $\theta_0 = (-0.5, -2)$ and $\theta_1 = (-2, -0.5)$ suggests that these optima are isolated. (b) A contour plot of $\mathcal{O}$ shows that the linear interpolation (dashed) goes through a region of high $\mathcal{O}$, and θ_0 and θ_1 are not isolated but are connected by a low value of $\mathcal{O}$. (c) Random perturbations around θ_0 show that many directions lead to a decrease in $\mathcal{O}$ indicating a local optimum and some directions have near zero change indicating flatness. (d) Projecting the points from (c) onto the two axes (dotted) gives us density plots for the gradient and curvature. The values on the gradient spectra are close to zero, indicating it is a critical point. The curvature spectra shows some negative curvature (local optimum) and some zero curvature (flatness). See Section 2.1 for detailed explanation.

- A novel visualization of the objective that captures local information about gradient and curvature.
- Experiments showing that policies with higher entropy induce a smoother objective that connects local optima and enable the use of larger learning rates.

2 Approach

Any improvements due to entropy regularization can be attributed to at least two reasons: (1) better objective landscapes; and (2) better gradient estimates. In Section 2.1 we introduce tools used to investigate (1) in the context of general optimization problems. We will then explain the RL policy optimization problem, and entropy regularization in Section 2.2.

2.1 Understanding the landscape of objective functions

We explain our experimental techniques by considering the general optimization problem and motivating the relevance of studying objective landscapes. We are interested in finding parameters, $\theta \in \mathbb{R}^n$, that maximize an objective function, $\mathcal{O} : \mathbb{R}^n \to \mathbb{R}$, denoted $\theta^* = \arg\max_\theta \mathcal{O}(\theta)$. The optimization algorithm takes the form of gradient ascent: $\theta_{t+1} = \theta_t + \eta \nabla_\theta \mathcal{O}$, where η is the learning rate, $\nabla_\theta \mathcal{O}$ is the gradient of $\mathcal{O}$ and t is the iteration number.

Why should we study objective landscapes? The "difficulty" of this optimization problem is given by the properties of $\mathcal{O}$. For example, $\mathcal{O}$ might have kinks and valleys making it difficult to find good solutions from different initializations (Li et al., 2018b). Similarly, if $\mathcal{O}$ contains very flat regions, optimizers like gradient ascent can take a very long time to escape them (Dauphin et al., 2014). Alternatively, if the curvature of $\mathcal{O}$ changes rapidly with every θ_t, then it will be difficult to pick a proper η.

Therefore, understanding the geometry of the loss function is central to our investigation. In the subsequent subsections, we describe two effective techniques for visualization of the policy optimization landscapes.

2.1.1 Linear interpolations

One approach to visualize an objective function is to interpolate θ in the 1D subspace between two points θ_0 and θ_1 (Chapelle & Wu, 2010; Goodfellow et al., 2015) by evaluating the objective at $\mathcal{O}((1-\alpha)\theta_0 + \alpha\theta_1)$ for $0 \le \alpha \le 1$. Such visualizations can tell us about the existence of valleys or monotonically increasing paths of improvement between the parameters.

Though this technique provides interesting visualizations, our conclusions are limited to the 1D slice. We highlight this in an example $\mathcal{O}$ where a linear interpolation between two optima suggests that they are

isolated (Figure 1a) but are in fact connected by a manifold of equal value $\mathcal{O}$ (Figure 1b, Draxler et al. (2018)). Even though Goodfellow et al. (2015) and Li et al. (2018b) extend interpolations to 2D subspaces, conclusions are still limited to low dimensional projections. In short, we must be careful to conclude general properties about the landscape using this visualization. In the next section, we describe a new visualization technique that, together with linear interpolations, can serve as a powerful tool for landscape analysis.

2.1.2 Objective function geometry using random perturbations

To combat some of the limitations described in Section 2.1.1, we develop a new method to locally characterize the properties of $\mathcal{O}$. In particular, we use this technique to (1) classify points in the parameter space as local optimum, saddle point, or flat regions; and (2) measure curvature of the objective during optimization. To understand the local geometry of $\mathcal{O}$ around a point θ_0 we sample directions uniformly at random on the unit ball, $d = \frac{d'}{||d'||}$ where $d' \sim N(0, I)$. We then evaluate $\mathcal{O}$ at a pair of new points: $\theta^+ = \theta_0 + \alpha d$ and $\theta^- = \theta_0 - \alpha d$ to evaluate how $\mathcal{O}$ is changing along the sampled direction. After collecting multiple samples and calculating the change for each pair with respect to the initial point, $\Delta^{\mathcal{O}+} = \mathcal{O}(\theta^+) - \mathcal{O}(\theta_0)$ and $\Delta^{\mathcal{O}-} = \mathcal{O}(\theta^-) - \mathcal{O}(\theta_0)$, we can then classify a point θ_0 according to:

1. All perturbations have both $\{\Delta^{\mathcal{O}+}, \Delta^{\mathcal{O}-}\}$ negative implies θ_0 is likely a local maximum.
2. All perturbations have both $\{\Delta^{\mathcal{O}+}, \Delta^{\mathcal{O}-}\}$ positive implies θ_0 is likely a local minimum.
3. All perturbations have one positive and one negative value in $\{\Delta^{\mathcal{O}+}, \Delta^{\mathcal{O}-}\}$ implies that θ_0 is likely on a linear region.
4. Some perturbations have either both $\{\Delta^{\mathcal{O}+}, \Delta^{\mathcal{O}-}\}$ positive or both values negative implies θ_0 is a saddle point.
5. Some perturbations have both $\{\Delta^{\mathcal{O}+}, \Delta^{\mathcal{O}-}\}$ close to zero, implies θ_0 is likely on a flat region.

We also note that the proposed technique correctly recovers that most directions around the local optimum presented in Section 2.1.1 have a negative or no change in the objective (Figure 1c). In contrast, a strict local optimum has no directions with zero change (Figure S1). Figure S2 show this technique can detect saddle points and linear regions.

Our method captures a lot of information about the local geometry. To summarize it, we can go one step further and disentangle information about gradient and curvature. If we assume $\mathcal{O}$ is locally quadratic, i.e., $\mathcal{O}(\theta) \approx a^T \theta + \frac{1}{2}\theta^T H \theta$, where $a^T \theta$ is a linear component and H is a symmetric matrix (i.e., Hessian), then:

$$
\begin{array}{rcl}
\Delta^{\mathcal{O}+} - \Delta^{\mathcal{O}-} & = & 2\alpha\nabla\mathcal{O}(\theta_0)^T d \ , \qquad\qquad\qquad\qquad\qquad (1) \\
\Delta^{\mathcal{O}+} + \Delta^{\mathcal{O}-} & = & \alpha^2 d^T H d^T \ . \qquad\qquad\qquad\qquad\qquad\quad (2)
\end{array}
$$

See the derivation in Appendix S.1.3. Therefore, projections of our scatter plots capture information about the components of the gradient and Hessian in the random direction d. By repeatedly sampling many directions we eventually recover how the gradient and curvature vary in many directions around θ_0. We can use a histogram to describe the density of these curvatures (Figure 1d). In particular, the maximum and minimum curvature values obtained from this technique are close to the maximum and minimum eigenvalues of H. This curvature spectrum is related to eigenvalue spectra which have been used before to analyze neural networks (Le Cun et al., 1991; Dauphin et al., 2014).

However, we stress that our method is more general than studying just the eigenvalues of H. A deeper discussion of benefits and limitations can be found in Appendix S.1.1 and S.1.2 respectively.

Our methods for understanding the objective function are summarized in Figure 1. Now that we have the tools to understand the objective function, we describe specific details about the RL optimization problem in the next section.

2.2 The policy optimization problem

In policy optimization, we aim to learn parameters, θ, of a policy, $\pi_\theta(a|s)$, such that when acting in an environment the sampled actions, a, maximize the discounted cumulative rewards, i.e., $\mathcal{O}_{ER}(\theta) = \mathbb{E}_{\pi_\theta}[\sum_{t=1}^{\infty} \gamma^t r_t]$, where γ is a discount factor. The gradient is given by the policy gradient theorem (Sutton et al., 2000) as:

$$
\nabla_\theta \mathcal{O}_{ER}(\theta) = \int_s d^{\pi_\theta}(s) \int_a \nabla_\theta \pi_\theta(a|s) Q^{\pi_\theta}(s, a) \mathrm{d}a \mathrm{d}s \qquad\qquad (3)
$$

3

where d^π is the stationary distribution of states and $Q^\pi(a_t, s_t)$ is the expected discounted sum of rewards starting state s_t, taking action a_t and then taking actions according to $a \sim \pi(\cdot|s)$.

One approach to prevent premature convergence to a deterministic policy is to induce *random* exploration using entropy regularization. One way to introduce entropy into the objective is to augment the rewards with entropy $r_t^\tau = r_t + \tau\mathbb{H}(\pi(\cdot|s_t))$ (Schulman et al., 2017a) and results in a slightly different gradient:

$$\nabla_\theta \mathcal{O}_{ENT}(\theta) = \int_s d^{\pi_\theta}(s) \int_a \pi(a|s)\Big[Q^{\tau,\pi_\theta}(s,a)\nabla_\theta \log \pi(a|s) + \tau\nabla_\theta\mathbb{H}(\pi(\cdot|s))\Big]\mathrm{d}a\mathrm{d}s \tag{4}$$

where $Q^{\tau,\pi_\theta}(s,a)$ is the expected discounted sum of entropy-augmented rewards starting at state s_t, taking action a_t and then taking actions according to $a \sim \pi(\cdot|s)$ (See derivation in Appendix S.2).

We use Equation 4 in a gradient ascent algorithm in Section 3.1 to show that even with access to the exact gradient, adding an entropy regularizer helps optimization. Since the end result of this method is to make policies more stochastic, in Section 3.2 we explicitly control entropy to investigate the impact of a more stochastic policy on the optimization landscape.

3 Results

3.1 Entropy helps even with access to the exact gradient

To emphasize that policy optimization is difficult even if we solved the "high variance" issue, we conduct experiments in a setting where the optimization procedure has access to the exact gradient. We will then link the poor optimization performance to visualizations of the objective function. Finally, we show how having an entropy augmented reward and, in general, a more stochastic policy changes this objective resulting in overall improvement in the solutions found.

3.1.1 Experimental Setup: Environments with no variance in the gradient estimate

Firstly, to investigate the claims of "high variance" we set our experiment in an environment where the gradient can be calculated exactly. In particular, we replace the integrals with summations and use environment dynamics to calculate Equation 4 exactly resulting in no sampling error. We chose a 5×5 Gridworld with two rewards: $R_{\mathrm{opt}} = 5$ at the bottom left corner and $R_{\mathrm{sub}} = 5\nu$ at the top right corner. Our agent starts in the top left corner and has four actions parameterized by categorical distribution $\pi(a|s_t) \propto \exp(\theta^T s_t)$. As such there are two locally optimal policies: go down, π_{opt} and go right, π_{sub}. To study entropy regularization we augment our rewards using $r_t^\tau = r_t + \tau\mathbb{H}(\pi(\cdot|s_t))$ as discussed in Section 2.2. We refer to the case where $\tau = 0$ as the *true objective*.

3.1.2 Are poor gradient estimates the main issue with policy optimization?

After running exact gradient ascent in the Gridworld starting from different random initializations of θ_0, we find that about 25% of policies that we converge to are sub-optimal: There is some inherent difficulty in the geometry of the optimization landscape independent of sampling noise. To get a better understanding of this landscape, we analyze two solutions that parameterize policies that are nearly deterministic for their respective rewards θ_{sub} and θ_{opt}. The objective function around θ_{sub} has a negligible gradient and small strictly negative curvature values indicating that the solution is a very flat local optimum (Figure 2a). On a more global scale, θ_{sub} and θ_{opt} are located in flat regions separated by a sharp valley of poor solutions (Figure 2b).

These results suggest that much of the difficulty in policy optimization comes from the flatness and valleys in the objective function and not poor gradient estimates. In the next sections, we investigate entropy regularization and link such improvements to the objective function.
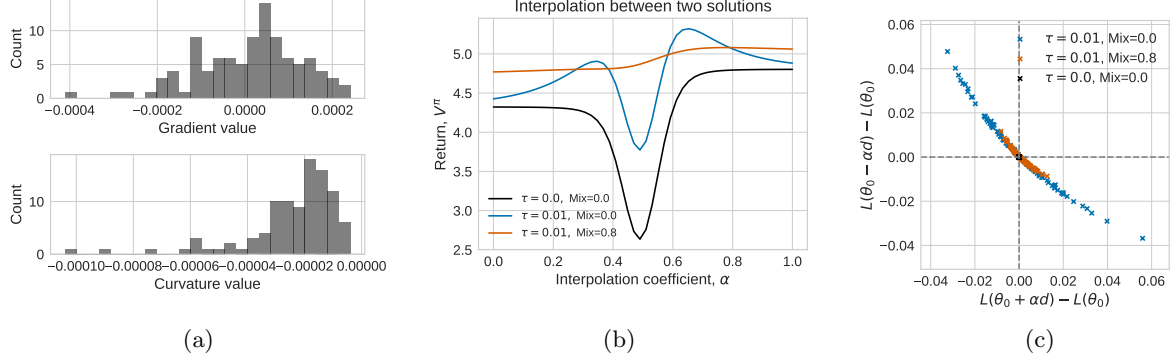
(a)   (b)   (c)

Figure 2: **Objective function geometry around solutions in the Gridworld.** (a) The gradient and curvature spectrum for a suboptimal solution θ_{sub} under the true objective shows that there is a negligible gradient and negative curvature values suggest that the solution is a local optimum and that there is no direction of improvement. (b) Linear interpolation between two solutions that find the suboptimal and optimal rewards are separated by a valley of poor solutions in the true objective (black). Under an entropy regularized objective with a stochastic policy these local optima become connected (orange) in this 1D slice. (c) Scatter plot for the change in objective for different random directions. With entropy and a more stochastic policy (blue and orange) many directions give positive improvement. See Section 3.1 for a more detailed explanation.

3.1.3   Why does using entropy regularization find better solutions?

Our problem setting is such that an RL practitioner would intuitively think of using entropy regularization to encourage the policy to "keep exploring" even after finding R_{sub}. Indeed, adding entropy ($\tau > 0$) along with a decay, reduces the proportion of sub-optimal solutions found by the optimization procedure to 0 (Figure S3)[2]. We explore reasons for the improved performance in this section.

Augmenting the objective with entropy results in a non-negligible gradient and directions of positive curvature (Figure 2c) at θ_{sub}: It is no longer a flat, locally optimal solution. However, it does not mean that we will reach the best possible policy. Specifically, a local optimum in the entropy augmented objective is not guaranteed to correspond to a local optimum in the true objective. Indeed, adding entropy on its own creates two new local optima in the objective (Figure 2b, S4) effectively encouraging randomness.

We argue that this objective will never be observed because the policies being interpolating are *nearly deterministic* which means that any interpolated policy will have two actions with almost zero probability. However, optimization occurs with a *stochastic* policy: To recreate this effect, we visualize the slice with a policy where the minimum probability of each action is given by a non-zero constant, $\frac{\mathrm{Mix}}{|A|}$[3].

We show that optimizing a stochastic policy with the entropy regularized landscape can connect the two local optima: A monotonically increasing path now appears in this augmented objective to a region with high value in the true objective (Figure 2b, S4). This means that if we knew a good direction a priori, a simple line search would have found a better solution.

This section provided a different and more accurate interpretation for entropy regularization by connecting it to changes in objective function geometry. Given that entropy regularization encourages our policies to be more stochastic, we now ask *What is it about stochastic policies that help learning?*. In the next two sections, we explore two possible reasons for this in a more realistic high dimensional continuous control problem.

3.2   More stochastic policies induce smoother objectives in some environments

In Section 3.1.3 we saw that entropy and, more generally, stochasticity can induce a "nicer" objective to optimize. Our second experimental setup allows us to answer questions about the optimization implications of stochastic policies. We show qualitatively that high entropy policies can speed up learning and improve

---

[2]We classify a policy as *optimal* if it achieves a return greater than that of the deterministic policy reaching R_{sub}.

[3]This is done by setting the policy being evaluated to $(1 - \mathrm{Mix})\pi_{(1-\alpha)\theta_0 + \alpha\theta_1}(a|s) + \frac{\mathrm{Mix}}{|A|}$.
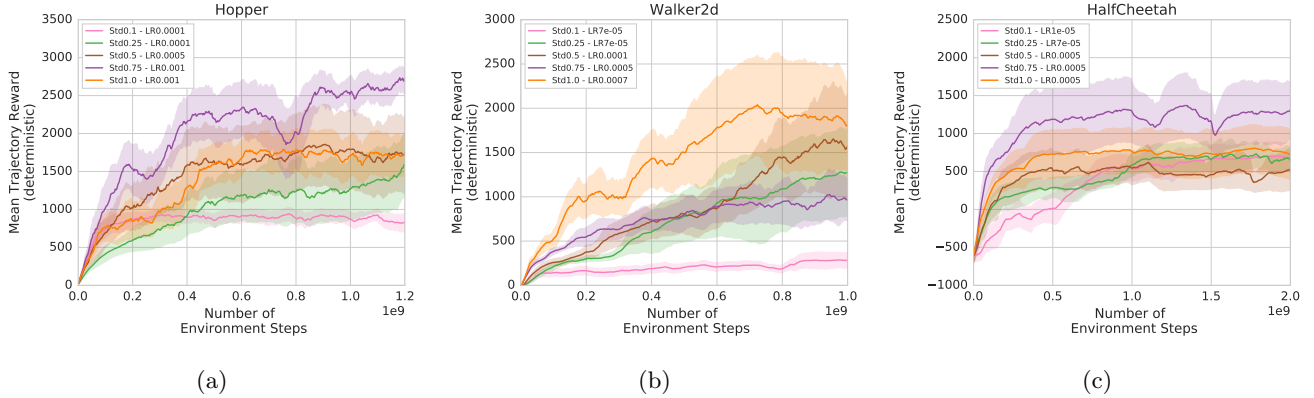
Figure 3: **Learning curves for policies with varying levels of σ for different continuous control tasks (a) Hopper-v1, (b) Walker2d-v1 and (c) HalfCheetah-v1.** In all environments using a high entropy policy results in faster learning. These effects are less apparent in HalfCheetah. In Hopper and Walker high entropy policies also find better final solutions. Learning rates are shown in the legends. Solid curve represents the average of 5 random seeds. Shaded region represents half a standard deviation for readability. Individual learning curves are shown in Figure S5 for Hopper, Figure S6 for Walker and Figure S7 for HalfCheetah. See Section 3.2.3 and 3.2.4 for reasons for improvement and discrepancies.

the final solutions found. We empirically investigate some reasons for these improvements and show that both these observations are environment-specific highlighting the challenges of policy optimization.

3.2.1 Experimental Setup: Environments that allow explicit study of policy entropy

Continuous control tasks from the MuJoCo simulator facilitate studying the impact of entropy because they require policies to be parameterized by Gaussian distributions (Todorov et al. (2012); We use Hopper, Walker2d and HalfCheetah from Brockman et al. (2016)). Specifically, the entropy of a Gaussian distribution depends only on σ, and thus we can control σ explicitly to study varying levels of entropy. To keep analysis as simple as possible, we parameterize the mean by $\theta^T s_t$ which is known to result in good performance (Rajeswaran et al., 2017). Since we do not have access to transition and reward dynamics, we cannot calculate Equation 3 exactly and use the REINFORCE estimator (Williams (1992), Appendix S.2.1). We use the techniques described in Section 2.1 to analyze θ under different values of σ. Specifically, we obtain a value of θ by optimizing a particular value of σ and to understand how objective landscapes change, reevaluate θ under a different value of σ. We consider $\sigma = 0$ to be the policy we are interested in and refer to the objective calculated as the *true objective*.

3.2.2 What is the effect of entropy on learning dynamics?

We first show that optimizing a more stochastic policy can result in faster learning in more complicated environments and better final policies in some.

In Hopper and Walker high entropy policies ($\sigma > 0.1$) quickly learn a better policy than low entropy policies ($\sigma = 0.1$) (Figure 3ab). In HalfCheetah, even though high entropy policies learn quicker (Figure 3c) the differences are less apparent and are more strongly influenced by the initialization seed (Figure S7). In both Hopper and Walker2d, the mean reward of final policies found by optimizing high entropy policies is $2 - 8$ times larger than a policy with $\sigma = 0.1$. Whereas in HalfCheetah, all policies converge to a similar final reward commonly observed in the literature (Schulman et al., 2017b).

Though statistical power to make fine-scale conclusions is limited, the qualitative trend holds: More stochastic policies perform better in terms of speed of learning and, in some environments, final policy learned. In the next two sections we investigate some reasons for these performance improvements as well as the discrepancy with HalfCheetah.
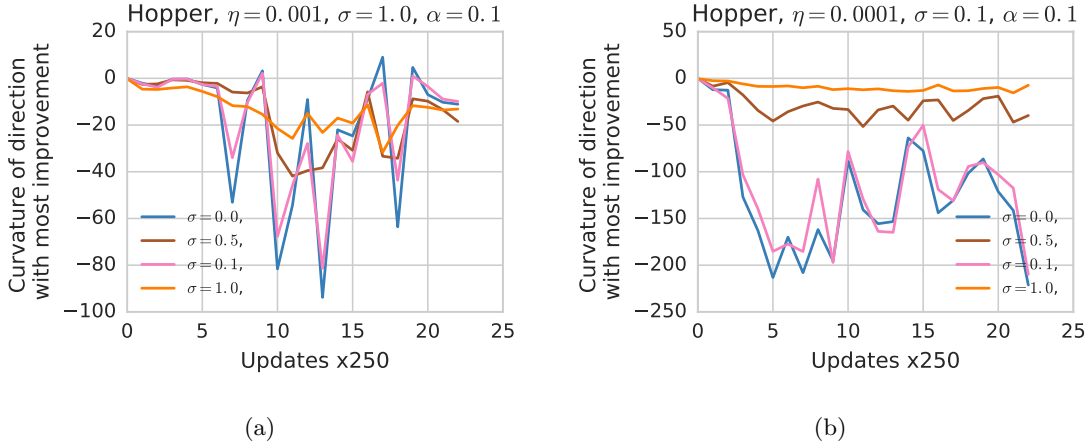
Figure 4: **The curvature for the direction of most improvement fluctuates rapidly for optimization objectives with low entropy ($\sigma \in \{0.0, 0.1\}$) compared to those of high entropy ($\sigma \in \{0.5, 1.0\}$).** (a) Optimizing with a high entropy policy ($\sigma = 1.0$) and plotting the corresponding curvature value for lower entropy policies. (b) Optimizing a low entropy policy ($\sigma = 0.1$) and plotting the curvature value higher entropy policies. Both Figures show that the curvature fluctuates more rapidly for low entropy policies regardless of which policy is actually being optimized. See Figure S8 for different values of α. See Figure S9 for the same analysis on Walker2d and Figure S10 for HalfCheetah.

3.2.3 Why do high entropy policies learn quickly?

We first focus on the speed of learning: A hint for the answer comes from our hyperparameter search over constant learning rates. In Hopper and Walker, the learning rate increases consistently with entropy: The learning rate for $\sigma = 1$ is 10 times larger than for $\sigma = 0.1$. This suggests that objective functions induced by optimizing high entropy policies are easier to optimize with a constant learning rate. Specifically, it suggests that the curvature in the direction of our updates does not change much. If the curvature fluctuated rapidly, a larger learning rate would overshoot when updating θ. Therefore, we claim that the curvature in directions of improvement of high entropy policies change less often making it more amenable to constant η optimization.

To investigate, we calculate the curvature of the objective during the first few thousand iterations of the optimization procedure. In particular, we record the curvature in a direction of improvement[4]. As expected, curvature values fluctuate with a large amplitude for low values of σ (Figure 4, S8, S9). In this setting, selecting a large and constant η might be more difficult compared to an objective induced by a policy with a larger σ. In contrast, the magnitude of fluctuations are only marginally affected by increasing σ in HalfCheetah (Figure S10) which might explain why using a more stochastic policy in this environment does not facilitate the use of larger learning rates.

In this section, we showed that fluctuations in the curvature of objectives decrease for more stochastic policies in some environments. The implications for these are two-fold: (1) It provides evidence for why high entropy policies facilitate the use of a larger learning rate; and (2) The impact of entropy can be highly environment specific. In the next section, we shift our focus to investigate the reasons for improved quality of *final* policies found when optimizing high entropy policies.

3.2.4 Can high entropy policies induce an objective with fewer local optima?

In this section, we improve our understanding of which values of θ are reachable at the end of optimization. We are trying to understand *Why do high entropy policies learn better final solutions?* Specifically, we attempt to classify the local geometry of parameters and investigate the effect of making the policy more stochastic. We will then argue that high entropy policies induce a more connected landscape.

To measure local optimality of solutions, we calculate the proportion of directions with negative curvature

---

[4]We selected the direction of improvement closest to the 90-th percentile which would be robust to outliers.

around final parameters of Hopper. Final solutions for $\sigma = 0.1$ have roughly 3 times more directions with negative curvature than $\sigma = 1.0$. This suggests that final solutions found when optimizing a high entropy policy lie in regions that are flatter and some directions *might* lead to improvement. To understand if a more stochastic policy can facilitate an improvement from a poor solution, we visualize the local objective for increasing values of σ. Figure 5 shows this analysis for one such solution. For deterministic policies there is a large gradient (Figure 5a) and 84% of directions have a detectable[5] negative curvature (Figure 5b) with the rest having near-zero curvature: The solution is likely near a local optimum. When the policy is made more stochastic, the gradient remains large but the number of directions with negative curvature reduces dramatically suggesting that the solution is in a linear region. However, just because there might be fewer directions with negative curvature, it does not imply that any of them reach good final policies.

To verify that there exists *at least* one path of improvement to a good solution, we linearly interpolate between this solution and parameters for a good final policy obtained by optimizing $\sigma = 1.0$ starting from the same random initialization (Figure 5c). Surprisingly, even though this is just one very specific direction, we find that in the high entropy objective there exists a monotonically increasing path to a better solution: If the optimizer knew the direction in advance, a simple line search would have improved upon a bad solution when using a high entropy policy. This finding extends to other pairs of parameters (Figure S13 for Hopper and Figure S14 for Walker2d) but not all (Figure S15) indicating that *some* slices of the objective function may become easier to optimize and find better solutions.

Our observations do not extend to HalfCheetah, where we were unable to find such pairs (Figure S15c) and specifically, objectives around final solutions did not change much for different values of σ (Figure S16). These observations suggest that the objective landscape is not affected significantly by changing how stochastic a policy is and explains the marginal influence of entropy in finding better solutions in this environment.

As seen before the impact of entropy on the objective function seems to be environment specific. However, in environments where the objective functions are affected by having a more stochastic policy, we have evidence that they can reduce at least a few local optima by connecting different regions of parameter space.
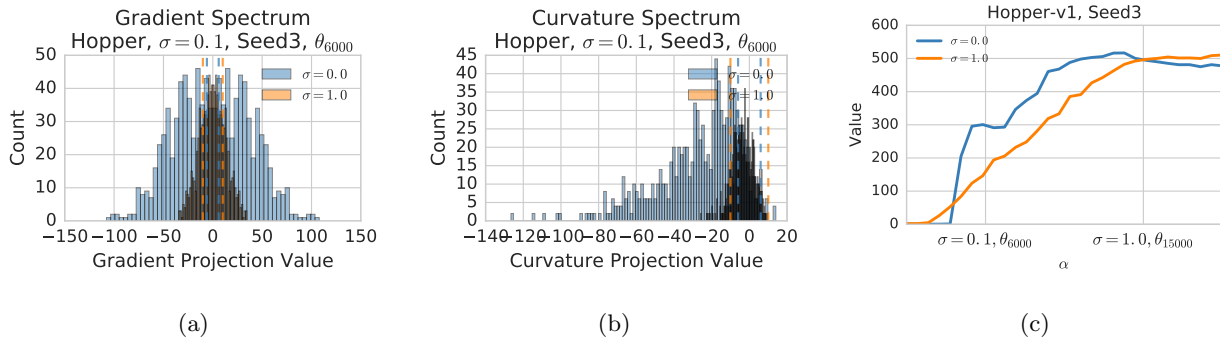

(a)

(b)

(c)

Figure 5: **Analyzing locally optimal solutions in objectives given by different amounts of entropy** (a) For both $\sigma = 0$ and $\sigma = 1.0$ there is a non-zero gradient. (b) For $\sigma = 0.0$, 85% of curvature values are negative. When σ is increased to 1, nearly all curvature values are within sampling noise (indicated by dashed horizontal lines). (c) A linear interpolation shows that a monotonically increasing path to a better solution exists from the poor parameter vector. See Figures S13 and S14 for a different seed and in Walker respectively. See Figure S15 for negative examples.

4   Related Work

There have been many visualization techniques for objective functions proposed in the last few years (Goodfellow et al., 2015; Li et al., 2018b; Draxler et al., 2018). Many of these project the high dimensional objective into one or two useful dimensions. Our work reuses linear interpolations and introduces a new technique that can characterize the local objective around parameters without losing information due to the projection.

---

[5]Taking into account noise in the sampling process.

Our technique is closely related to Li et al. (2018b) who used random directions for 2D interpolations, except that we interpolate in many more directions to summarize how the objective function changes locally.

Understanding the impact of entropy on the policy optimization problem was first studied by Williams & Peng (1991). A penalty similar to entropy regularization has also been explored in the context of deep learning. Chaudhari et al. (2017) show that such a penalty induces objectives with higher β-smoothness and complement our smoothness results. Recent work by Neu et al. (2017) has shown the equivalence between the type of entropy used and the corresponding dual optimization algorithm.

Our work is most closely related to concurrent work by Ilyas et al. (2018) who show that, in deep RL, gradient estimates can be uncorrelated with the true gradient but optimization can still perform well. This observation complements our work in saying that high variance is not the biggest issue we have to tackle. The authors use 2D interpolations to show that an ascent in surrogate objectives used in PPO did not necessarily correspond to an ascent in the true objective. Our results provide a potential explanation to this phenomenon: surrogate objectives can connect regions of parameter space where the true objective might decrease. In summary, Ilyas et al. (2018) share similar motivations to our work in appealing to the community to study the policy optimization problem more closely. Rajeswaran et al. (2017); Henderson et al. (2018) also had similar goals.

5   Discussion and Future Directions

The difficulty of policy optimization. Our work aims to redirect some research focus from the high variance issue to the study of better optimization techniques. In particular, if we were able to perfectly estimate the gradient, policy optimization would still be difficult due to the geometry of the objective function used in RL.

Specifically, our experiments bring to light two issues unique to policy optimization. Firstly, given that we are optimizing probability distributions, many reparameterizations can result in the same distribution. This results in objective functions that are especially susceptible to having flat regions and difficult geometries (Figure 2b, S14c). There are a few solutions to this issue: As pointed out in Kakade (2001), methods based on the natural gradient are well equipped to deal with plateaus induced by probability distributions. Alternatively, given that using natural policy gradient inspired methods like TRPO and surrogate objective methods like PPO avoids the poor solution in HalfCheetah suggests that these techniques are well motivated in RL (Schulman et al., 2015a, 2017b; Rajeswaran et al., 2017). Such improvements are orthogonal to the noisy gradient problem and suggest that making policy optimization easier is a fruitful line of research.

Secondly, the landscape we are optimizing is problem dependent and is particularly surprising in our work. Given that the mechanics of many MuJoCo tasks are very similar, our observations on Hopper and HalfCheetah are vastly different. This presents a challenge for both studying and designing optimization techniques. For example, if our analysis was restricted to just Hopper and Walker, our conclusions with respect to entropy would have been different. The MuJoCo environments considered here are deterministic: An interesting and important extension would be to investigate other sources of noise and in general answering *What aspects of the environment induce difficult objectives?*

Our proposed method will likely be useful in answering at least a few such questions. In particular, this work only exploited a small fraction of information from the local geometry captured by the random perturbations. A thorough exposition of this technique in studying the geometry and topology of optimization landscapes will be the subject of future work.

Random exploration and objective function geometry. Our learning curves are not surprising under the mainstream interpretation of entropy regularization: A small value of σ will not induce a policy that adequately "explores" the whole range of available actions. However, our results on HalfCheetah tell a different story: All values of σ converged to policy with similar final reward (Figure 3c) indicating that the mechanism through which entropy regularization impacts learning is not necessarily through "exploration". Our combined results from curvature analysis and linear interpolations (Figure 4 and 5) have shown explicitly that the geometry of the objective function is linked to the entropy of the policy being optimized. Thus using a more stochastic policy, induced by entropy regularization, facilitates the use of a larger learning rate and when the entropy is increased at a local optimum, provides more directions of improvement.

Our results should hold for any *random* exploration technique. Investigating how other *directed* explo-

ration techniques impact the landscape will be useful to inform the design of new exploration techniques. We expect that the ultimate impact of exploration in RL is to make the objective function smoother and easier to optimize using gradient ascent algorithms.

Finally, our experimental results make one suggestion: Smoothing can help learning. Therefore, *How can we leverage these observations to make new algorithms?* Perhaps some work should be directed on alternate smoothing techniques: Santurkar et al. (2018) suggests that techniques like batch normalization also smooth the objective function and might be able to replicate some of the performance benefits. In the context of RL, Q-value smoothing has been explored in Nachum et al. (2018); Fujimoto et al. (2018) that resulted in performance gains for an off-policy policy optimization algorithm.

In summary, our work has provided a new tool for and highlighted the importance of studying the underlying optimization landscape in direct policy optimization. We have shown that these optimization landscapes are highly problem dependent making it challenging to come up with general purpose optimization algorithms. We show that optimizing policies with more entropy results in a smoother objective function that can be optimized with a larger learning rate. Finally, we identify a myriad of future work that might be of interest to the community with significant impact.

Acknowledgements

References

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Olivier Chapelle and Mingrui Wu. Gradient descent optimization of smoothed information retrieval metrics. *Information retrieval*, 13(3):216–235, 2010.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *International Conference on Learning Representations*, 2017.

Po-Wei Chou, Daniel Maturana, and Sebastian Scherer. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International Conference on Machine Learning*, pp. 834–843, 2017.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred A Hamprecht. Essentially no barriers in neural network energy landscape. *International Conference on Machine Learning*, 2018.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *International Conference on Machine Learning*, 2018.

Yasuhiro Fujita and Shin-ichi Maeda. Clipped action policy gradient. *International Conference on Machine Learning*, 2018.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *International Conference on Learning Representations*, 2015.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

Peter Henderson, Joshua Romoff, and Joelle Pineau. Where did my optimum go?: An empirical analysis of gradient descent optimization in policy gradient methods. *European Workshop on Reinforcement Learning*, 2018.

Andrew Ilyas, Logan Engstrom, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Are deep policy gradient algorithms truly policy gradient algorithms? *arXiv preprint arXiv:1811.02553*, 2018.

Sham Kakade. A natural policy gradient. In *Neural Information Processing Systems*, volume 14, pp. 1531–1538, 2001.

Khimya Khetarpal, Zafarali Ahmed, Andre Cianflone, Riashat Islam, and Joelle Pineau. Re-evaluate: Reproducibility in evaluating reinforcement learning algorithms. *2nd Reproducibility in Machine Learning Workshop at ICML 2018*, 2018.

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

Yann Le Cun, Ido Kanter, and Sara A Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters*, 66(18):2396, 1991.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. *International Conference on Learning Representations (ICLR)*, 2018a.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *International Conference on Learning Representations, Workshop Track*, 2018b.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Ofir Nachum, Mohammad Norouzi, George Tucker, and Dale Schuurmans. Smoothed action value functions for learning gaussian policies. *International Conference on Machine Learning*, 2018.

Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

Aravind Rajeswaran, Kendall Lowrey, Emanuel V Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pp. 6550–6561, 2017.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization?(no, it is not about internal covariate shift). *arXiv preprint arXiv:1805.11604*, 2018.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. *arXiv preprint arXiv:1704.06440*, 2017a.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017b.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012.

George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. *International Conference on Learning Representations (Workshop Track)*, 2018.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

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

Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

S   Appendix

S.1   More details about visualizing objective functions using random perturbations

We introduced a novel technique for visualizing objective functions by using random perturbations. Understanding the benefits and limitations is key to knowing when this method will be useful.

S.1.1   Benefits of random perturbations

1. Since our technique is only bounded by the number of samples we wish to obtain, it allows us to scale beyond regimes where computing eigenvalues of H might be computationally expensive. In particular, our method does not require computing any gradients and is ammenable to massive parallelization.

2. Our random perturbations capture a lot of information about the local geometry of an objective function. Though in this work we discuss two possible summarizations, other summarizations may exist that capture different geometrical and topological properties of the objective function around this point.

S.1.2   Limitations of random perturbations

In this section we discuss limitations of the random perturbation methods.

1. Near solutions in high dimensional parameter spaces, the number of directions with positive curvature are few and we might miss them because of the stochastic nature of the algorithm. In our work this does not play a big role since our policies have less than 200 parameters. Further work is needed to determine how many samples are needed to capture curvature in all directions. A potentially interesting extension will be to understand the geometry in the intrinsic subspace from Li et al. (2018a).
2. If α is set too large, we might detect false positive local optima since we are perturbing the parameters too far. In our work we were careful to pick the smallest α that showed deviation from noise.

S.1.3 Derivation for Equation 1

Here we derive the form for projections onto the two diagonal axes $x = y$ and $x = -y$. Assume $\mathcal{O}(\theta) \approx a^T\theta + \frac{1}{2}\theta^T H\theta$. Now

$$\mathcal{O}(\theta_0 + \alpha d) = a^T(\theta_0 + \alpha d) + \frac{1}{2}(\theta_0 + \alpha d)^T H(\theta_0 + \alpha d) \tag{5}$$

$$= a^T\theta_0 + \alpha a^T d + \frac{\alpha}{2}[\theta_0^T H d + d^T H\theta_0] + \frac{\alpha^2}{2}d^T H d + \frac{1}{2}\alpha\theta_0^T H\theta_0 \tag{6}$$

$$= \mathcal{O}(\theta_0) + \alpha a^T d + \frac{\alpha^2}{2}d^T H d + \alpha\theta_0^T H d \tag{7}$$

Therefore:

$$\Delta^{\mathcal{O}+} = \mathcal{O}(\theta_0 + \alpha d) - \mathcal{O}(\theta_0) = \alpha a^T d + \frac{\alpha^2}{2}d^T H d + \alpha\theta_0^T H d \tag{8}$$

and similarly,

$$\Delta^{\mathcal{O}-} = \mathcal{O}(\theta_0 - \alpha d) - \mathcal{O}(\theta_0) = -\alpha a^T d + \frac{\alpha^2}{2}d^T H d - \alpha\theta_0^T H d \tag{9}$$

Now doing the projection onto the diagonal axes we get:

$$\Delta^{\mathcal{O}+} + \Delta^{\mathcal{O}-} = \alpha^2 d^T H d \tag{10}$$

which gives us information about the Hessian in direction d and

$$\Delta^{\mathcal{O}+} - \Delta^{\mathcal{O}-} = 2\alpha a^T d + 2\alpha\theta_0^T H d = 2\alpha(a + \theta_0 H)^T d = 2\alpha\nabla\mathcal{O}(\theta_0)^T d \tag{11}$$

which gives us information about the gradient in that direction.

By repeating this procedure and obtaining many samples, and can thus get an understanding of how $\mathcal{O}$ changes in many directions around θ_0.

S.2 Derivation of Entropy-augmented exact policy gradient (Equation 4)

In this section we derive the exact gradient updates used in Section 3.1 for the entropy regularized objective. This derivation differs from but has the same solution as Sutton et al. (2000) when $\tau = 0$. Recall that the objective function is given by:

$$V^\pi(s_0) = \sum_a \pi(a|s = s_0)Q^\pi(s_0, a) \tag{12}$$

where $V^\pi(s_0)$ is the expected discounted sum of rewards from the starting state. We can substitute the definition of $Q^\pi(s, a) = \left[r(s, a) + \tau\mathbb{H}(\pi(\cdot|s)) + \gamma\sum_{s'} P(s'|s = s_0, a)V^\pi(s')\right]$ to obtain a recursive formulation of the objective.

$$V^\pi(s_0) = \sum_a \pi(a|s = s_0)\left[r(s_0, a) + \tau\mathbb{H}(\pi(\cdot|s_0)) + \gamma\sum_{s'} P(s'|s = s_0, a)V^\pi(s')\right] \tag{13}$$

If our policy π is parameterized by θ we can take the gradient of this objective function so that we can use it in a gradient ascent algorithm:

$$\frac{d}{d\theta}V^\pi(s) = \frac{d}{d\theta}\sum_a \pi(a|s)Q^\pi(s, a) \tag{14}$$

By using the product rule we have that:

$$\frac{d}{d\theta}V^\pi(s) = \sum_a Q^\pi(s, a)\frac{d}{d\theta}\pi(a|s) + \sum_a \pi(a|s)\frac{d}{d\theta}Q^\pi(s, a) \tag{15}$$

13

We can now focus on the term $\frac{dQ^\pi(s,a)}{d\theta}$:

$$\frac{d}{d\theta}Q^\pi(s,a) \quad = \quad \frac{d}{d\theta}\Big[r(s,a) + \tau\mathbb{H}(\pi(\cdot|s)) + \gamma\sum_{s'}P(s'|s,a)V^\pi(s')\Big] \tag{16}$$

$$= \quad \frac{d}{d\theta}\tau\mathbb{H}(\pi(\cdot|s)) + \gamma\sum_{s'}P(s'|s,a)\frac{d}{d\theta}V^\pi(s') \tag{17}$$

We can substitute the last equation in our result from the product rule expansion:

$$\frac{d}{d\theta}V^\pi(s) = \sum_a Q^\pi(s,a)\frac{d}{d\theta}\pi(a|s) + \sum_a \pi(a|s)\Big[\frac{d}{d\theta}\tau\mathbb{H}(\pi(\cdot|s)) + \gamma\sum_{s'}P(s'|s,a)\frac{d}{d\theta}V^\pi(s')\Big] \tag{18}$$

We can use the fact that $\frac{d}{d\theta}\pi(a|s) = \pi(a|s)\frac{d}{d\theta}\log\pi(a|s)$ to simplify some terms:

$$\frac{d}{d\theta}V^\pi(s) = \sum_a \pi(a|s)\Big[Q^\pi(s,a)\frac{d}{d\theta}\log\pi(a|s) + \frac{d}{d\theta}\tau\mathbb{H}(\pi(\cdot|s))\Big] + \sum_a\pi(a|s)\gamma\sum_{s'}P(s'|s,a)\frac{d}{d\theta}V^\pi(s') \tag{19}$$

We can now consider the term $Q^\pi(s,a)\frac{d}{d\theta}\log\pi(a|s) + \frac{d}{d\theta}\tau\mathbb{H}(\pi(\cdot|s))$ as a "cumulant" or augmented reward $\hat{r}(s,a)$. Let us define $r^\pi(s) = \sum_a \pi(a|s)\hat{r}(a,s)$ and r^π the vector form containing the values $r^\pi(s)$ and g^π the vector form of $\frac{d}{d\theta}V^\pi$ for each state. We also define $P^\pi(s',s) = \sum_a \pi(a|s)\sum'_s p(s'|s,a)$ as the transition matrix representing the probability of going from $s \to s'$. If we write everything in matrix form we get that:

$$g^\pi = r^\pi + \gamma P^\pi g^\pi \tag{20}$$

This is a Bellman equation and we can solve it using the matrix inverse:

$$g^\pi = \frac{r^\pi}{(I - \gamma P^\pi)} \tag{21}$$

In non-matrix form this is:

$$\frac{dV^\pi(s)}{d\theta} = \sum_t \gamma^t P(s_t = s|s_0)\sum_a \pi(a|s)\Big[Q^\pi(s,a)\frac{d}{d\theta}\log\pi(a|s) + \frac{d}{d\theta}\tau\mathbb{H}(\pi(\cdot|s))\Big] \tag{22}$$

To get the correct loss, we extract the term corresponding to s_0:

$$e_{s0}^T(I - \gamma P^\pi)^{-1}\sum_a \pi(a|s)\Big[Q^\pi(s,a)\frac{d}{d\theta}\log\pi(a|s) + \frac{d}{d\theta}\tau\mathbb{H}(\pi(\cdot|s))\Big] \tag{23}$$

We make this loss suitable for automatic differentiation by placing a "stop gradient" in the appropriate locations:

$$e_{s0}^T(I - \gamma P^\pi)^{-1}\sum_a STOP(\pi(a|s))\Big[\log\pi(a|s)STOP(Q^\pi(s,a)) + \tau\mathbb{H}(\pi(\cdot|s))\Big] \tag{24}$$

The code that implements the above loss is provided here: https://goo.gl/D3g4vE

S.2.1 REINFORCE gradient estimator

In most environments, we do not have access to the exact transition and reward dynamics needed to calculate $d^\pi(s)$. Therefore, the gradient of $\mathcal{O}_{ER}$ given in Equation 3 cannot be evaluated directly. The REINFORCE (Williams, 1992) estimator is derived by considering the fact that $\nabla_\theta\pi_\theta(s|a) = \pi_\theta(s|a)\nabla_\theta\log\pi_\theta(s|a)$, allowing us to estimate $\nabla\mathcal{O}_{ER}$ using Monte-Carlo samples:

$$\nabla\mathcal{O}_{ER}(\theta) \approx \frac{1}{N}\sum_n \sum_{s_t^n,a_t^n\sim\pi} \nabla\log\pi(a_t^n|s_t^n)G_t \tag{25}$$

where G_t is the Monte-Carlo estimate for $Q^\pi(a_t, s_t)$. We use $N = 128$ and the batch average baseline to reduce variance in the estimator.

S.3   Open source implementation details and reproducibility instructions

S.3.1   Objective function analysis demonstration

We provide a demonstration of our random perturbation method (Section 2.1.2) in a Colab notebook using toy landscapes as well as FashionMNIST (Xiao et al., 2017)[6].

S.3.2   Reinforcement Learning Experiments

Our Gridworld is implemented in the easyMDP package[7] which provides access to quantities needed to calculate the analytic gradient. The experiments are reproduced in a Colab with embedded instructions[8]

Our continuous control experiments use the REINFORCE algorithm implemented in Tensorflow Eager[9][10]. Evaluations are conducted independently of trajectories collected during training (Khetarpal et al., 2018). Learning curves are generated based on the deterministic evaluation to ensure policies trained using different standard deviations can be compared.

To do objective function analysis, it is necessary to store the parameters of the model every few updates. Once the optimization is complete and parameters have been obtained we provide a script that does linear interpolations between two parameters[11]. Different standard deviations can be given to investigate the objective function under different stochasticities.

Similarly, we also provide the script that does random perturbations experiment around one parameter[12]. To scale up and collect a large number of samples, we recommend running this script multiple times in parallel as evaluations in random directions can be done independently of one another.

We also provide scripts to create plots which can easily be imported into a Colab[13].

S.4   Limitations of the Analysis

In this section we describe some limitations of our work:

1. The high entropy policies we train are especially susceptible to over-reliance on the fact that actions are clipped before being executed in the environment. This phenomenon has been documented before in Chou et al. (2017); Fujita & Maeda (2018). Beta policies and TanhGaussian policies are occasionally used to deal with the boundaries naturally. In this work we chose to use the simplest formulation possible: the Gaussian policy. In the viewpoint of the optimization problem it still maximizes the objective. Since all relevant continuous control environments use clipping, we were careful to ensure our policies were not completely clipped in this work and that σ was always smaller than the length of the window of values that would not be clipped. We do not expect clipping to have a significant impact on our observations with respect to smoothing behaviours of high entropy policies.

2. Recent new work (Ilyas et al., 2018) has shown that in the sample size we have used to visualize the landscape, kinks and bumps are expected but get smoother with larger sample sizes. Since our batch size is higher than most RL methods but not as high as Ilyas et al. (2018), we argue that it captures what day-to-day algorithms face. We were careful to ensure our evaluations has a small standard error.

---

[6]Landscape analysis demo: https://goo.gl/nXEDXJ

[7]easyMDP https://github.com/zafarali/emdp

[8]Exact policy gradient experiments: https://goo.gl/D3g4vE.

[9]Algorithm: https://goo.gl/ZbtLLV.

[10]Launcher script:https://goo.gl/dMgkZm.

[11]Interpolation experiments: https://goo.gl/CGVPvG

[12]Random perturbation experiments: https://goo.gl/vY7gYK
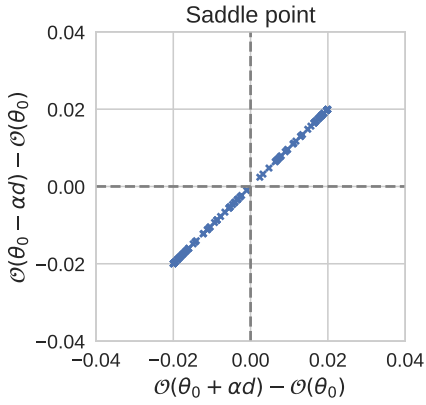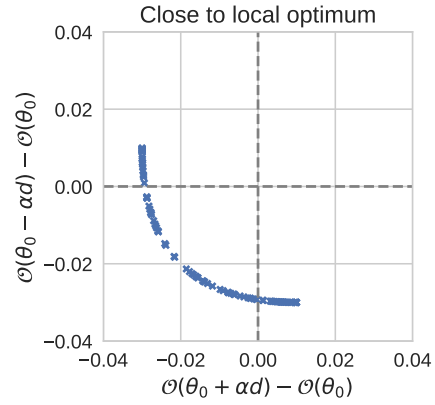
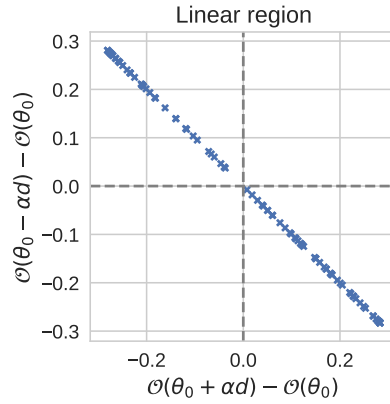[13]Analysis tools: https://goo.gl/DMbkZA
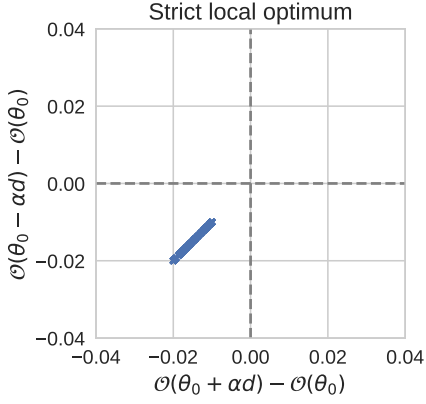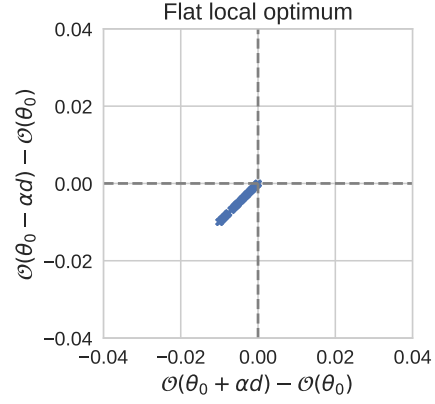
Figure S1: **Example visualizations of the random perturbation method of local optima in simple loss functions.** Scatter plots can distinguish between strict local optimum (where all directions are negative and have negative curvature) with a flat optimum (where some directions might have 0 curvature.

Figure S2: **Example visualizations of the random perturbation method of saddle points, linear regions in simple loss functions.**
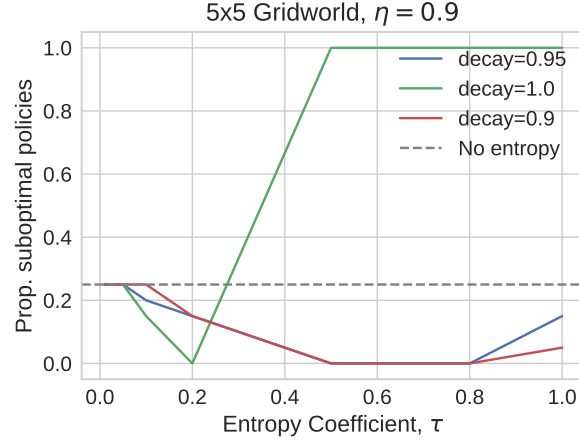
Figure S3: **Proportion of sub-optimal solutions found for different entropy coefficients τ and decay factors.** Using an entropy coefficient helps learn the optimal policy.

(a)

(b)
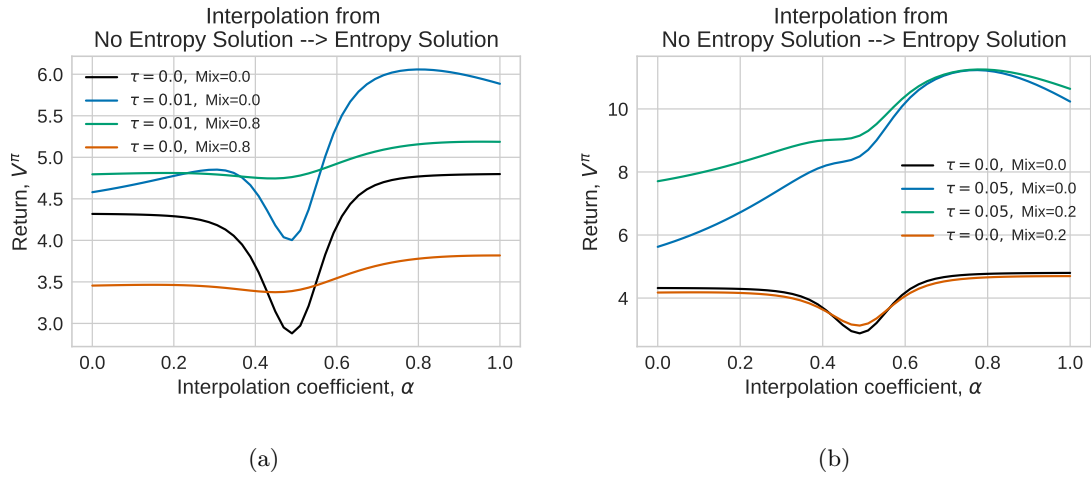
Figure S4: Visualizing the objective function for different combinations of τ and minimum policy entropy (mix) in the interpolation between solutions found when optimizing with and without entropy entropy regularization.
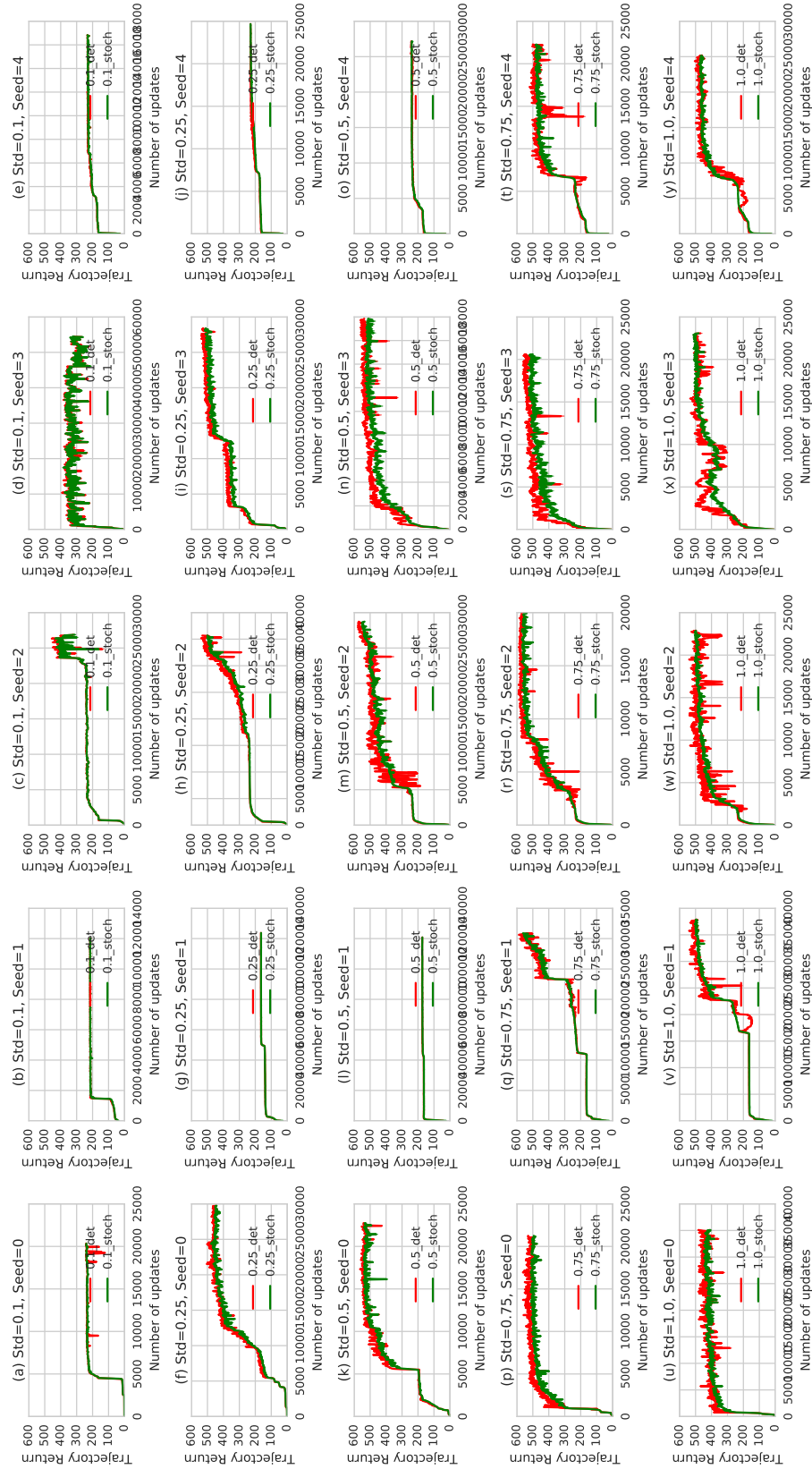
Figure S5: **Individual learning curves for Hopper**
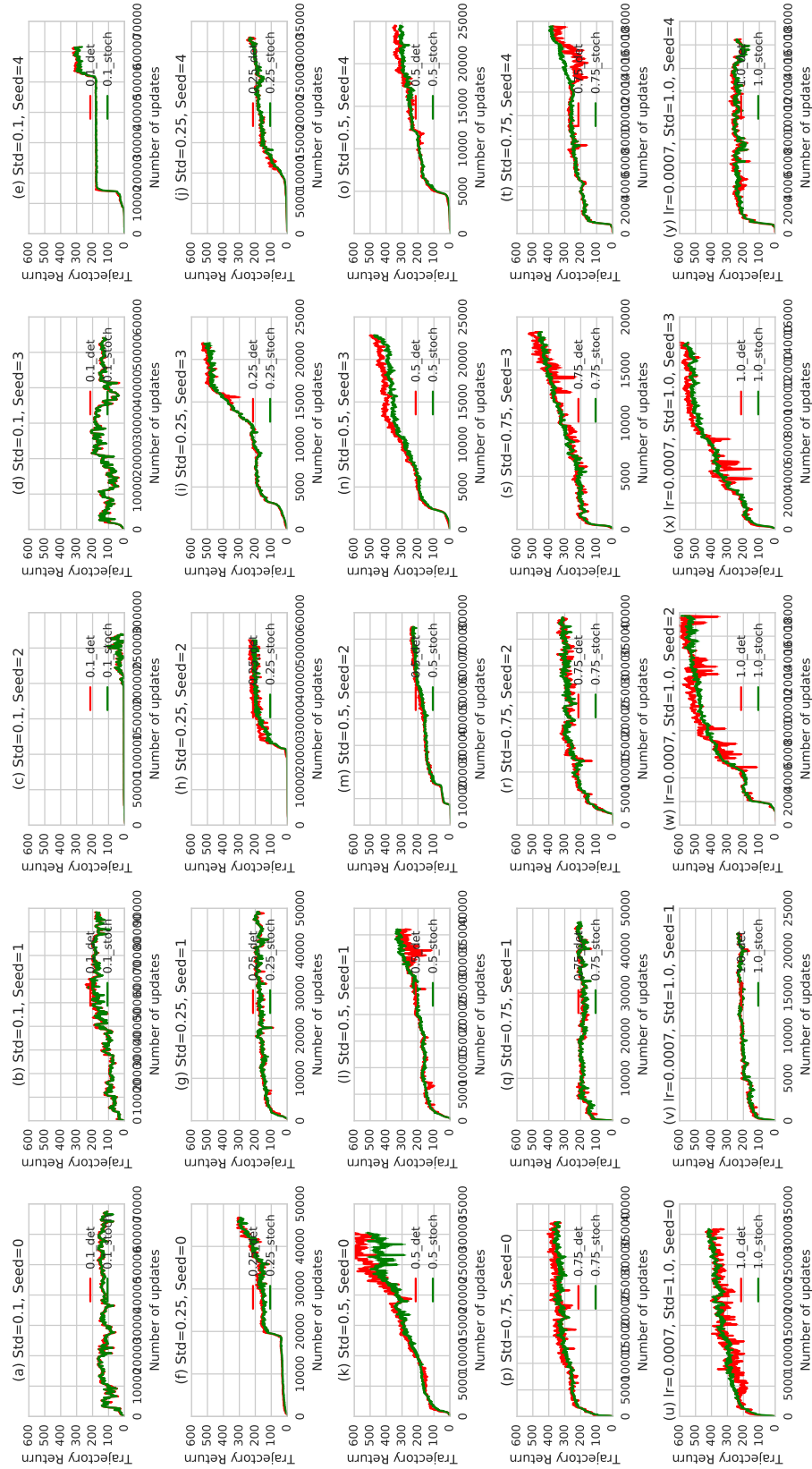
Figure S6: **Individual learning curves for Walker**

19
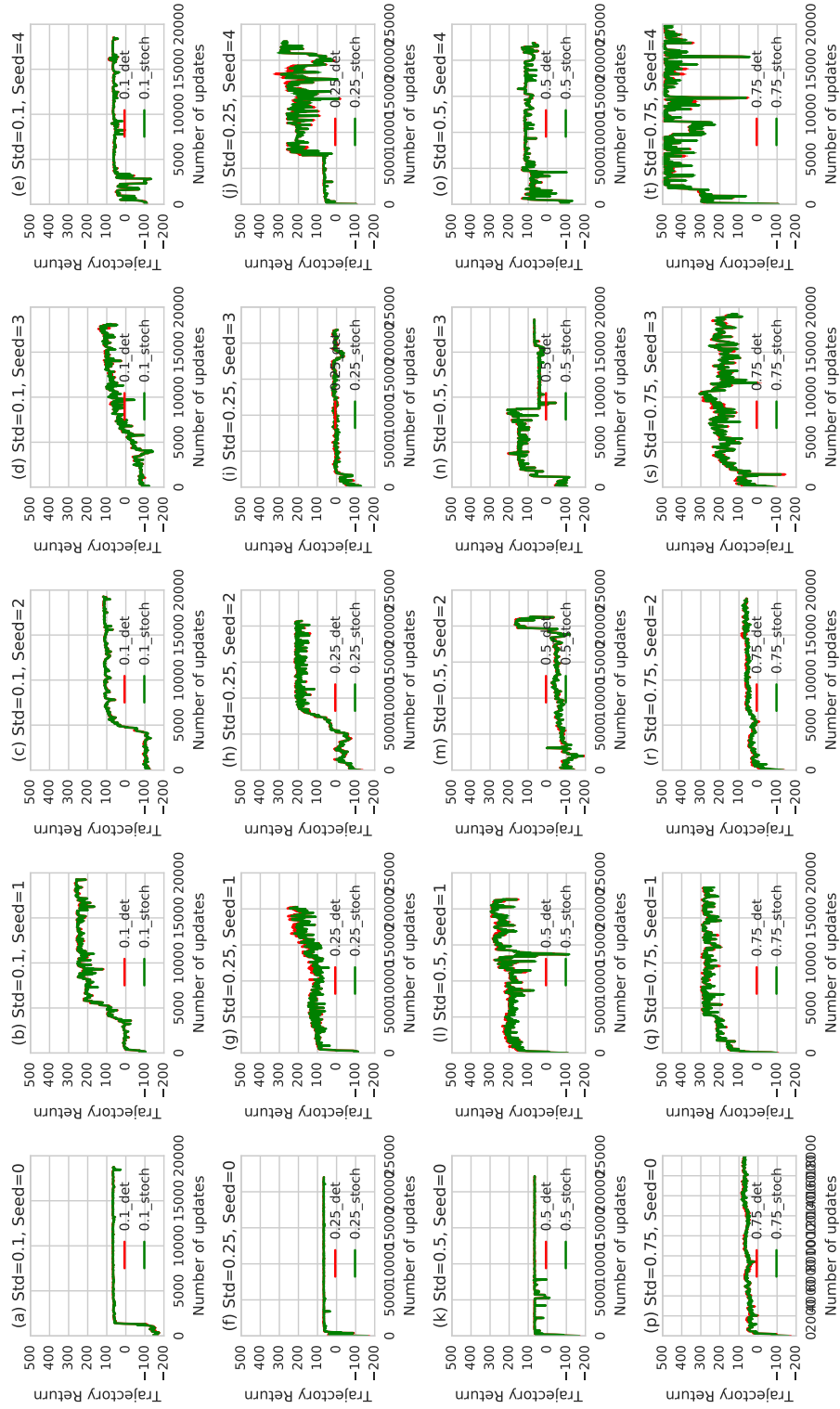
Figure S7: **Individual learning curves for HalfCheetah**
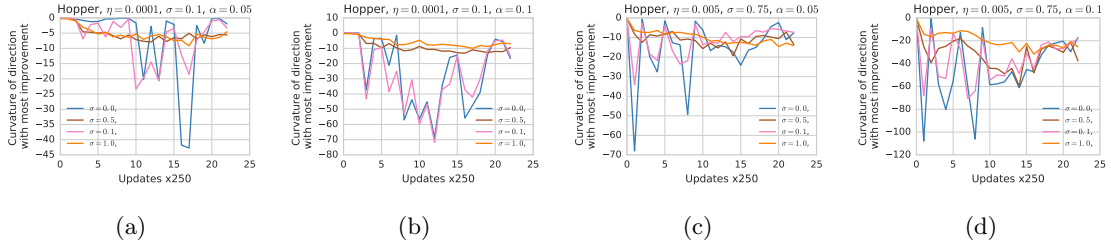
Figure S8: Curvature for the direction with the most improvement during the optimization for different seeds and standard deviations in Hopper.
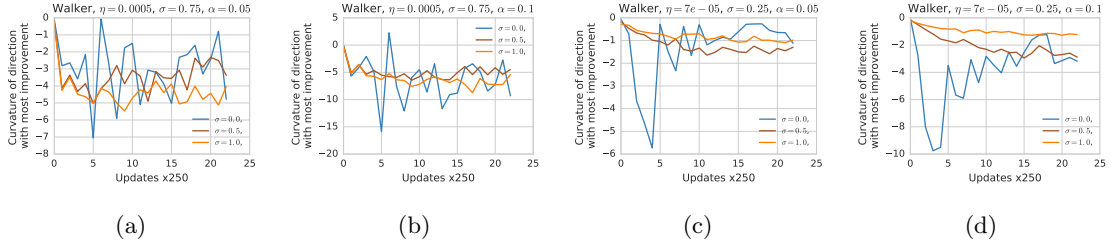
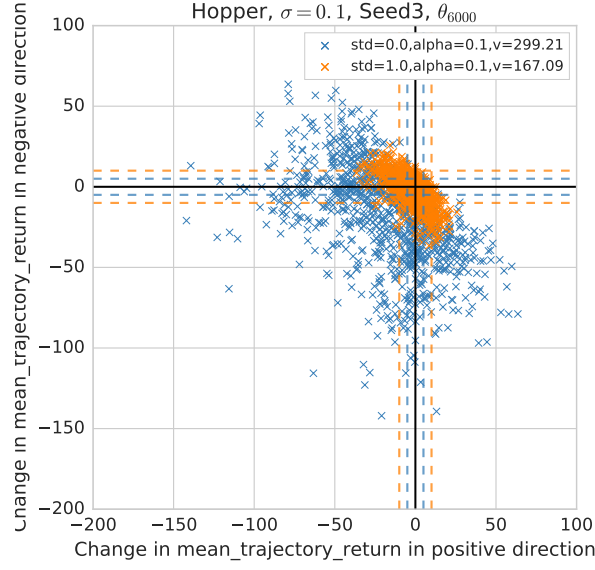Figure S9: Curvature for the direction with the most improvement during the optimization for different seeds and standard deviations in Walker2d.
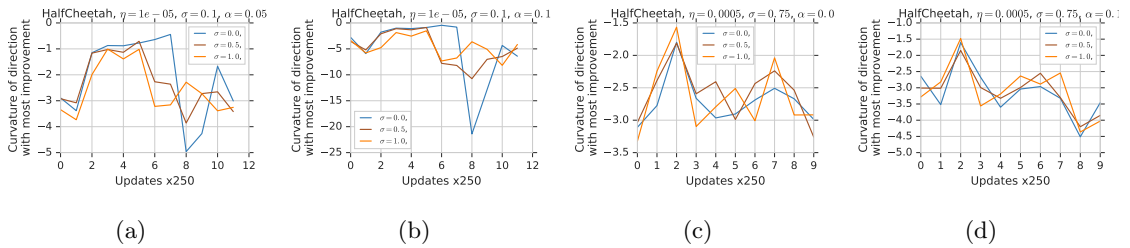
Figure S10: Curvature for the direction with the most improvement during the optimization for different seeds and standard deviations in HalfCheetah.

Figure S11: **Scatter plot for randomly drawn directions for the local optimum shown in Figure 5** For $\sigma = 0$, θ_{6000} is close to a flat local optimum. Most directions are negative and they all have near zero or negative curvature. For $\sigma = 1$, there are fewer negative directions, and more importantly less negative curvature: Indicating an almost linear region. This linear region can be seen in the 1D interpolation (Figure 5c)
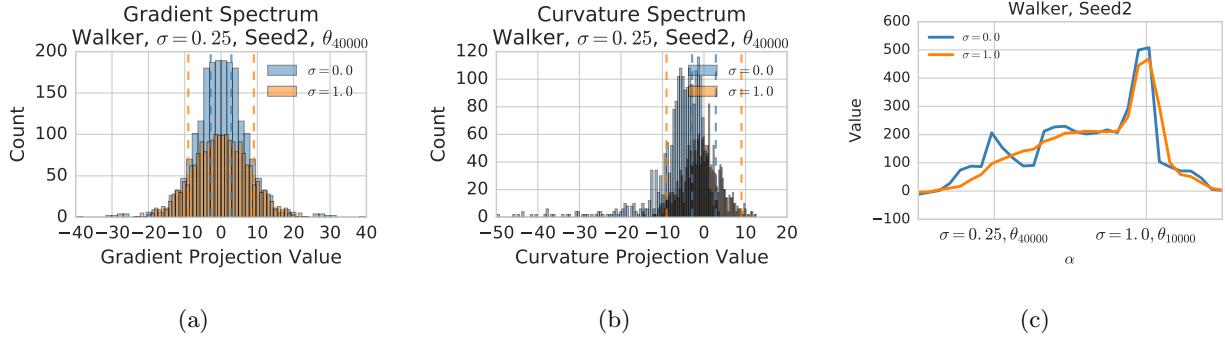

| (a) | (b) | (c) |

Figure S12: Curvature and gradient spectra for two solutions in Walker2d. (a) For both values of σ, the gradient is non-negligible. (b) For $\sigma = 0$ most directions have a significant negative curvature indicating local optimum. For $\sigma = 1$ all curvature values are within the noise threshold. Putting these results together suggests that the solution is at a local optima when $\sigma = 0$ but in a linear region when $\sigma = 1$. (c) A monotonically increasing path to a better solution exists if we knew the direction to a solution a-priori.
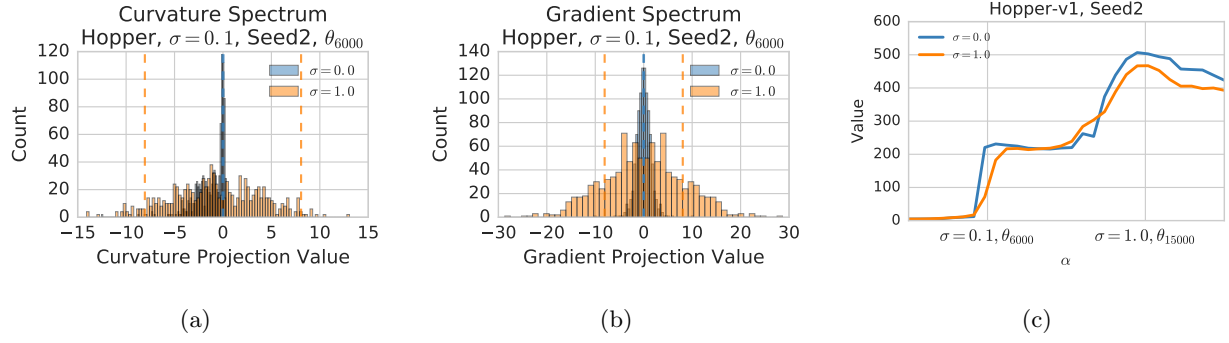
(a)
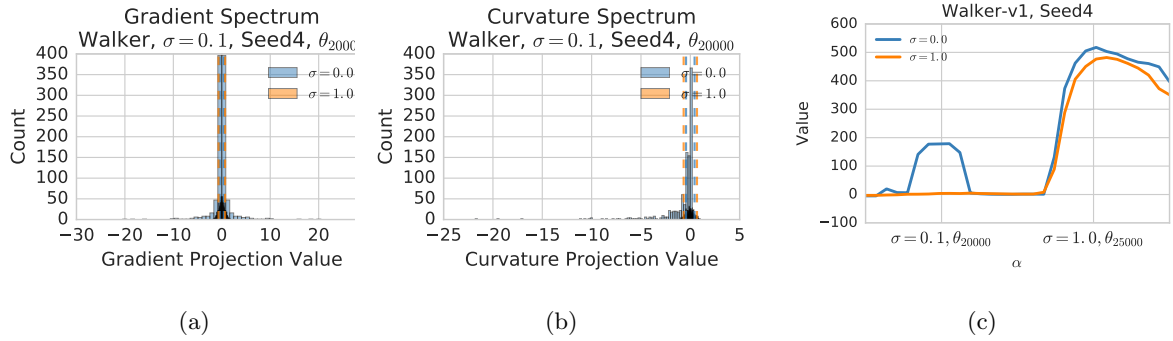
(b)

(c)

Figure S13

(a)

(b)

(c)

Figure S14: Curvature and gradient spectra for two solutions in Walker2d. (a) For $\sigma = 0$ there is a non-negligible gradient which becomes indistinguishable from noise when $\sigma = 1$. (b) For $\sigma = 0$ the local objective has mostly negative curvature, but when increased to $\sigma = 1$ there is no curvature. Putting these two results together, having no gradient and no curvature means that the solution ends up being in a flat region. (c) A linear interpolation confirms this observation in one dimension.
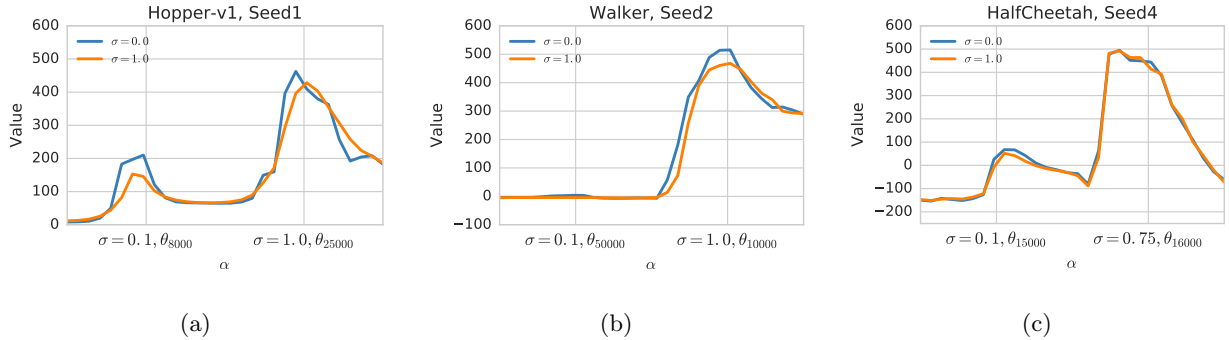

(a)

(b)

(c)

Figure S15: **Negative examples for linear interpolations between solutions.** Interpolations between these solutions do not show a monotonically increasing path under the high entropy objective suggesting that though high entropy objectives might connect some local optima, they do not connect all.
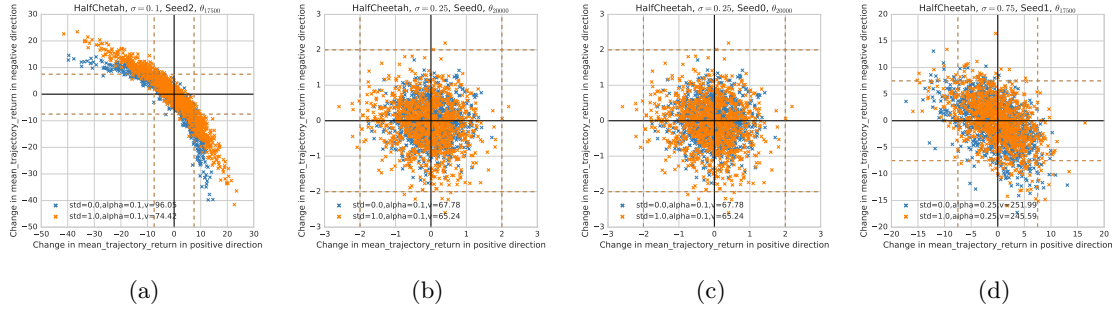
23

(a) (b) (c) (d)

Figure S16: Local objective functions do not change much in HalfCheetah when σ is increased.