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 amenable to massive parallelization.
2. Our random perturbations capture a lot of information about the local geometry of an objective function. In this work we discuss two possible summarizations that capture information about the gradient and Hessian. Other summarizations may exist that capture different geometrical and topological properties of the objective function around this point.

S.1.2. 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

$$\mathcal{O}(\theta_0 + \alpha d) = a^T(\theta_0 + \alpha d) + \frac{1}{2}(\theta_0 + \alpha d)^T H(\theta_0 + \alpha d) \tag{4}$$

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

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

Therefore:

$$\Delta^{\mathcal{O}+} = \mathcal{O}(\theta_0 + \alpha d) - \mathcal{O}(\theta_0) \tag{6}$$

$$= \alpha a^T d + \frac{\alpha^2}{2}d^T Hd + \alpha\theta_0^T Hd \tag{7}$$

and similarly,

$$\Delta^{\mathcal{O}-} = \mathcal{O}(\theta_0 - \alpha d) - \mathcal{O}(\theta_0) \tag{8}$$

$$= -\alpha a^T d + \frac{\alpha^2}{2}d^T Hd - \alpha\theta_0^T Hd \tag{9}$$

Now doing the projection onto the diagonal axes we get:

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

which gives us information about the Hessian in direction d and

$$\Delta^{\mathcal{O}+} - \Delta^{\mathcal{O}-} = 2\alpha a^T d + 2\alpha\theta_0^T Hd \tag{11}$$

$$= 2\alpha(a + \theta_0 H)^T d \tag{12}$$

$$= 2\alpha\nabla\mathcal{O}(\theta_0)^T d \tag{13}$$

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.1.3. Limitations

Consider $\mathcal{O}(\theta) = -\sum_{i=1}^{k_1}\theta_i^2 + -\sum_{i=k_1+1}^{k_1+k_2}(\theta_i - 2)^2$ where at $\theta = \vec{0}$ the function is locally optimal in k_1 directions but there are k_2 ascent directions that improve $\mathcal{O}$. To get an idea of the extent of this limitation, the loss function is perturbed based on directions given by a stochastic gradient[8] in contrast to random directions. When the total number of dimensions,

[8] Stochastic gradients are simulated by adding Gaussian noise with co-variance $\epsilon 2I_{k_1+k_2}$ to the true gradient (Mandt et al., 2017)

$k_1 + k_2$, is small, random perturbations can accurately capture all directions regardless of the relative magnitudes of k_1 and k_2 (Figure S3ab). When the number of dimensions, $k_1 + k_2$, is large and the number of improvement directions, $k_1 = k_2$, both methods discover the ascent directions (Figure S3c). However, when $k_1 \gg k_2$, both random perturbations and stochastic gradients miss the ascent directions unless noise is small (Figure S3d).

S.2. Derivation of Entropy-augmented exact policy gradient (Equation 3)

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:

$$V^\pi(s_0) = \sum_a \pi(a|s = s_0) Q^\pi(s_0, a) \tag{14}$$

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.

$$\begin{aligned} V^\pi(s_0) = \sum_a \pi(a|s = s_0) \big[& r(s_0, a) + \tau\mathbb{H}(\pi(\cdot|s_0)) \\ & + \gamma \sum_{s'} P(s'|s = s_0, a) V^\pi(s') \big] \end{aligned} \tag{15}$$

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:

$$\frac{d}{d\theta} V^\pi(s) = \frac{d}{d\theta} \sum_a \pi(a|s) Q^\pi(s, a) \tag{16}$$

By using the product rule we have that:

$$\frac{d}{d\theta} V^\pi(s) = \sum_a Q^\pi(s,a) \frac{d}{d\theta} \pi(a|s) + \sum_a \pi(a|s) \frac{d}{d\theta} Q^\pi(s,a) \tag{17}$$

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

$$\begin{aligned} \frac{d}{d\theta} Q^\pi(s,a) &= \frac{d}{d\theta} \left[r(s,a) + \tau\mathbb{H}(\pi(\cdot|s)) + \gamma \sum_{s'} P(s'|s,a) V^\pi(s') \right] \\ &= \frac{d}{d\theta} \tau\mathbb{H}(\pi(\cdot|s)) + \gamma \sum_{s'} P(s'|s,a) \frac{d}{d\theta} V^\pi(s') \end{aligned} \tag{18}$$

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

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

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:

$$\frac{d}{d\theta} V^\pi(s) = \sum_a \pi(a|s) \left[Q^\pi(s,a) \frac{d}{d\theta} \log \pi(a|s) + \frac{d}{d\theta} \tau\mathbb{H}(\pi(\cdot|s)) \right] + \sum_a \pi(a|s) \gamma \sum_{s'} P(s'|s,a) \frac{d}{d\theta} V^\pi(s') \tag{20}$$

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:

$$g^\pi = r^\pi + \gamma P^\pi g^\pi \tag{21}$$

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

$$g^\pi = \frac{r^\pi}{(I - \gamma P^\pi)} \tag{22}$$

Written explicitly this is:

$$\frac{dV^{\pi}(s)}{d\theta} = \sum_{t} \gamma^{t} P(s_t = s|s_0) \sum_{a} \pi(a|s) \left[Q^{\pi}(s,a) \frac{d}{d\theta} \log \pi(a|s) + \frac{d}{d\theta} \tau \mathbb{H}(\pi(\cdot|s)) \right] \tag{23}$$

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

$$e_{s0}^{T}(I - \gamma P^{\pi})^{-1} \sum_{a} \pi(a|s) \left[Q^{\pi}(s,a) \frac{d}{d\theta} \log \pi(a|s) + \frac{d}{d\theta} \tau \mathbb{H}(\pi(\cdot|s)) \right] \tag{24}$$

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

$$e_{s0}^{T}(I - \gamma P^{\pi})^{-1} \sum_{a} STOP(\pi(a|s)) \left[\log \pi(a|s) STOP(Q^{\pi}(s,a)) + \tau \mathbb{H}(\pi(\cdot|s)) \right] \tag{25}$$

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 by the policy gradient theorem,

$$\nabla_{\theta} \mathcal{O}_{ER}(\theta) = \int_{s} d^{\pi_\theta}(s) \int_{a} \nabla_{\theta} \pi_{\theta}(a|s) Q^{\pi_\theta}(s,a) \mathrm{d}a \mathrm{d}s \tag{26}$$

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:

$$\nabla \mathcal{O}_{ER}(\theta) \approx \frac{1}{N} \sum_{n} \sum_{s_t^n, a_t^n \sim \pi} \nabla \log \pi(a_t^n|s_t^n) G_t \tag{27}$$

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 and to account of the confounding variance reduction effect of σ in the case of Gaussian policies (Zhao et al., 2011).

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)[9].

S.3.2. REINFORCEMENT LEARNING EXPERIMENTS

Our Gridworld is implemented in the easyMDP package[10] which provides access to quantities needed to calculate the analytic gradient. The experiments are reproduced in a Colab with embedded instructions[11].

Our high dimensional experiments used the Hopper-v1, Walker2d-v1 and HalfCheetah-v1 continuous control environments from OpenAI Gym (Brockman et al., 2016) based on Mujoco (Todorov et al., 2012). The REINFORCE algorithm is implemented in Tensorflow Eager[12][13]. Learning curves are generated based on the deterministic evaluation $\sigma = 0$ to ensure policies trained using different standard deviations can be compared. Evaluation rollouts are independent of trajectories collected during training (Khetarpal et al., 2018).

To do thorough 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

[9] Landscape analysis demo: https://goo.gl/nXEDXJ
[10] easyMDP https://github.com/zafarali/emdp
[11] Exact policy gradient experiments: https://goo.gl/D3g4vE.
[12] Algorithm: https://goo.gl/ZbtLLV.
[13] Launcher script: https://goo.gl/dMgkZm.

parameters[14]. Different standard deviations can be given to investigate the objective function for policies with different amounts of entropy.

Similarly, we also provide the script that does random perturbations experiment around one parameter[15]. 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 used ≈ 1000 evaluations per parameter vector. Each evaluation used 512 rollouts.

We also provide a small library to create plots that can easily be imported into a Colab[16].

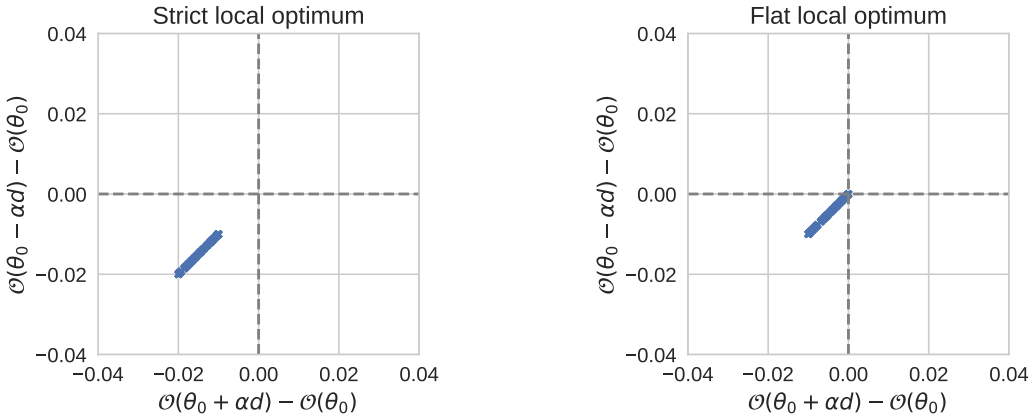
S.4. Limitations of the Analysis

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.

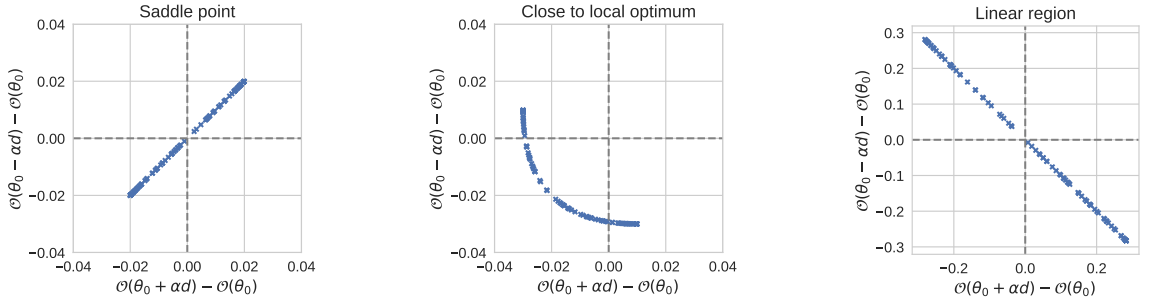
2. Recent new work (Ilyas et al., 2018) has shown that in the sample size we have used to visualize the landscape, kinks and bumps are expected but get smoother with larger sample sizes. Though our batch size is higher than most RL methods but not as high as (Ilyas et al., 2018), it captures what day-to-day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

[14]Interpolation experiments: https://goo.gl/CGVPvG
[15]Random perturbation experiments: https://goo.gl/vY7gYK
[16]Analysis tools: https://goo.gl/DMbkZA



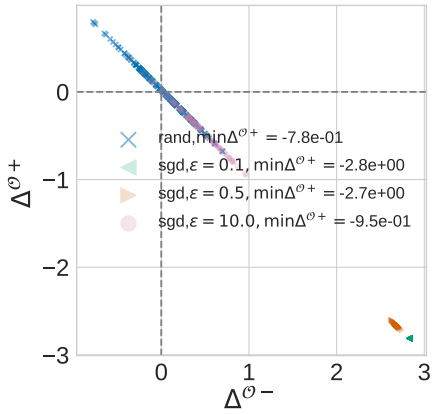
(a) $-x^2 + 0y^2$ at $(x, y) = (0, 0)$ (b) $-x^2 - 2y^2$ at $(x, y) = (0, 0)$

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.

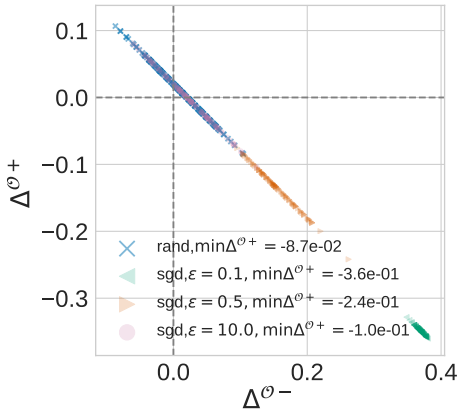


(a) $-x^2 + y^2$ at $(x, y) = (0, 0)$ (b) $-x^2 - y^2$ at $(x, y) = (0.1, 0.0)$ (c) $-2x + 2y$ at $(x, y) = (0, 0)$

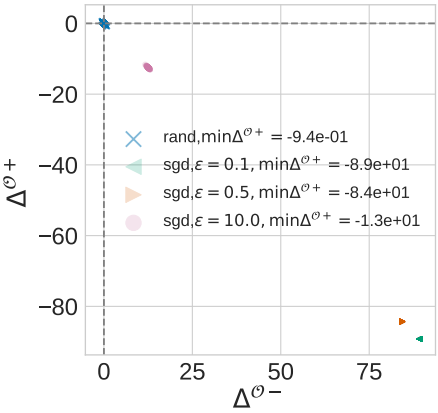
Figure S2. Example visualizations of the random perturbation method of saddle points, linear regions in simple loss functions.



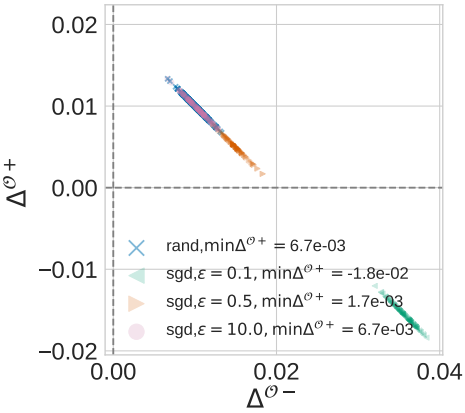
(a) $k_1 = 50, k_2 = 50$



(b) $k_1 = 99, k_2 = 1$

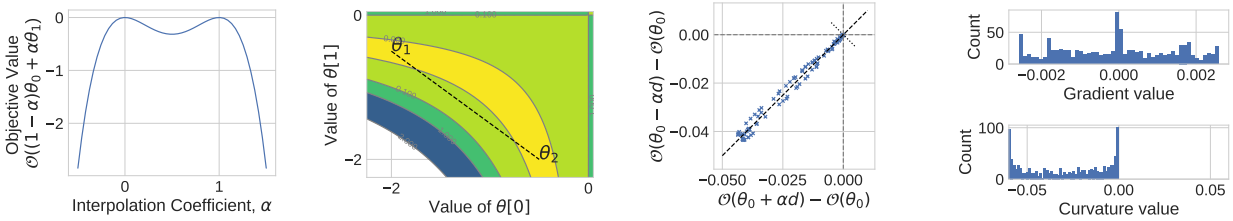


(c) $k_1 = 50000, k_2 = 50000$



(d) $k_1 = 99999, k_2 = 1$

Figure S3. **Assessing the limitations of random perturbations.** We repeat the sampling procedure using directions given by stochastic gradients rather than random directions. For small models, $k_1 + k_2 = 100$, our method can correctly recover all directions of descent. For larger models, $k_1 + k_2 = 10000$, we find that when the number of descent directions, k_2, is small both methods will miss this direction. It is only if the gradient noise ϵ is small will it capture the descent direction. We numerically verify that running stochastic gradient descent from this point does not lead to a solution in a reasonable number of iterations.



(a) Linear Interpolation (b) 2D Plot of $\mathcal{O}$ (c) $\{\Delta^{\mathcal{O}^+}, \Delta^{\mathcal{O}^-}\}$ at θ_0 (d) Projection of (c)

Figure S4. **Summary of the methods on** $\mathcal{O}(\theta) = -(1-\theta[0]\theta[1])^2$ (a) A linear interpolation between two local maxima, $\theta_0 = (-0.5, -2)$ and $\theta_1 = (-2, -0.5)$ suggests that these maxima 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 maxima 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 maximum) and some zero curvature (flatness). See Section 2.1 for detailed explanation.

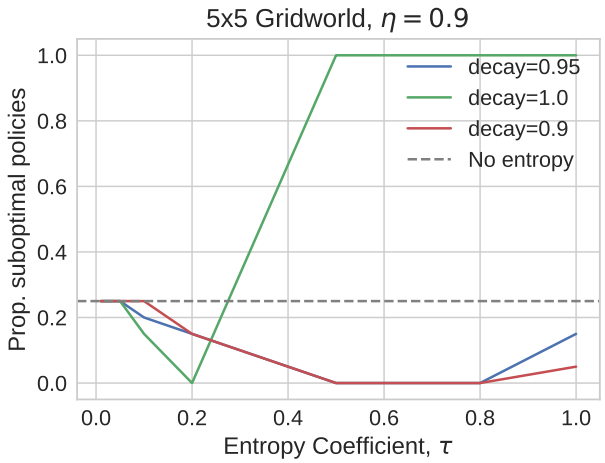
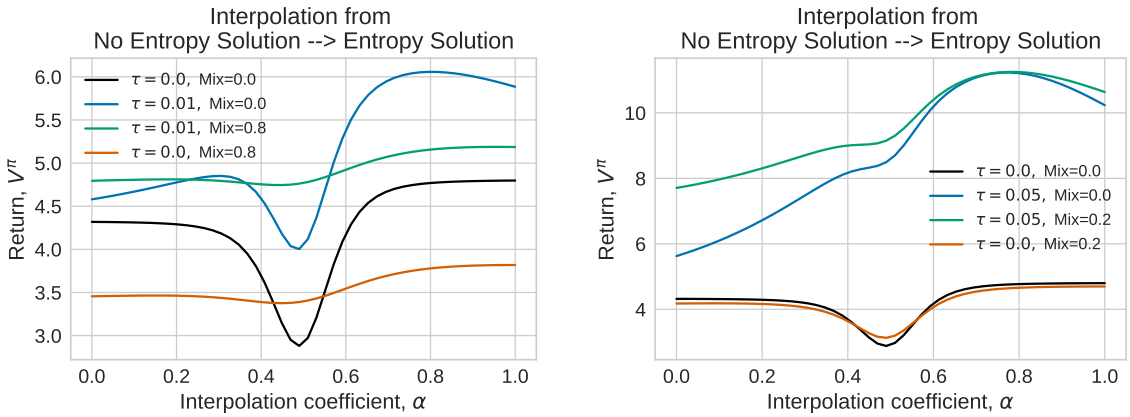


Figure S5. Proportion of sub-optimal solutions found for different entropy coefficients τ and decay factors. Using an entropy coefficient helps learn the optimal policy.



(a) (b)

Figure S6. 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.

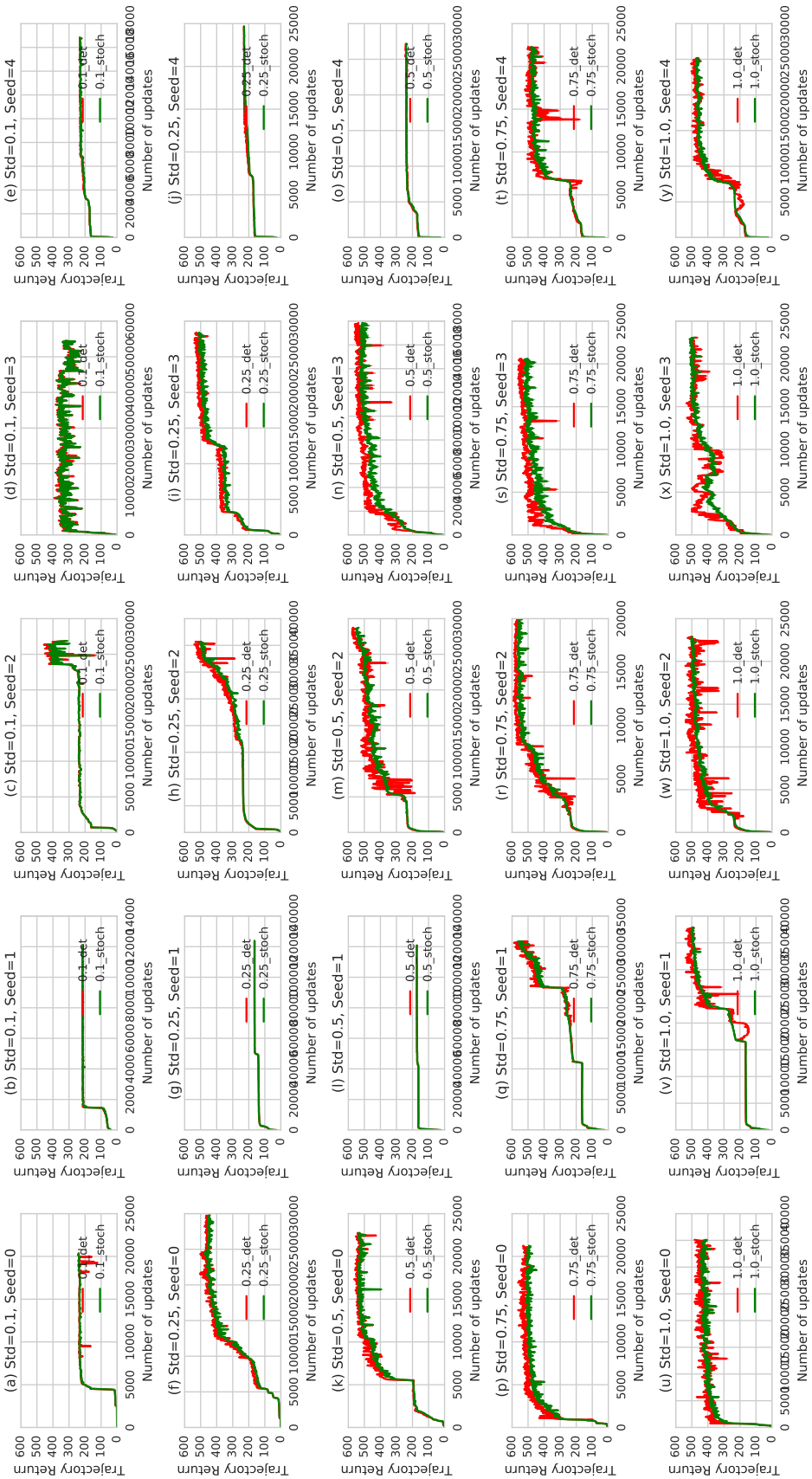


Figure S7. Individual learning curves for Hopper

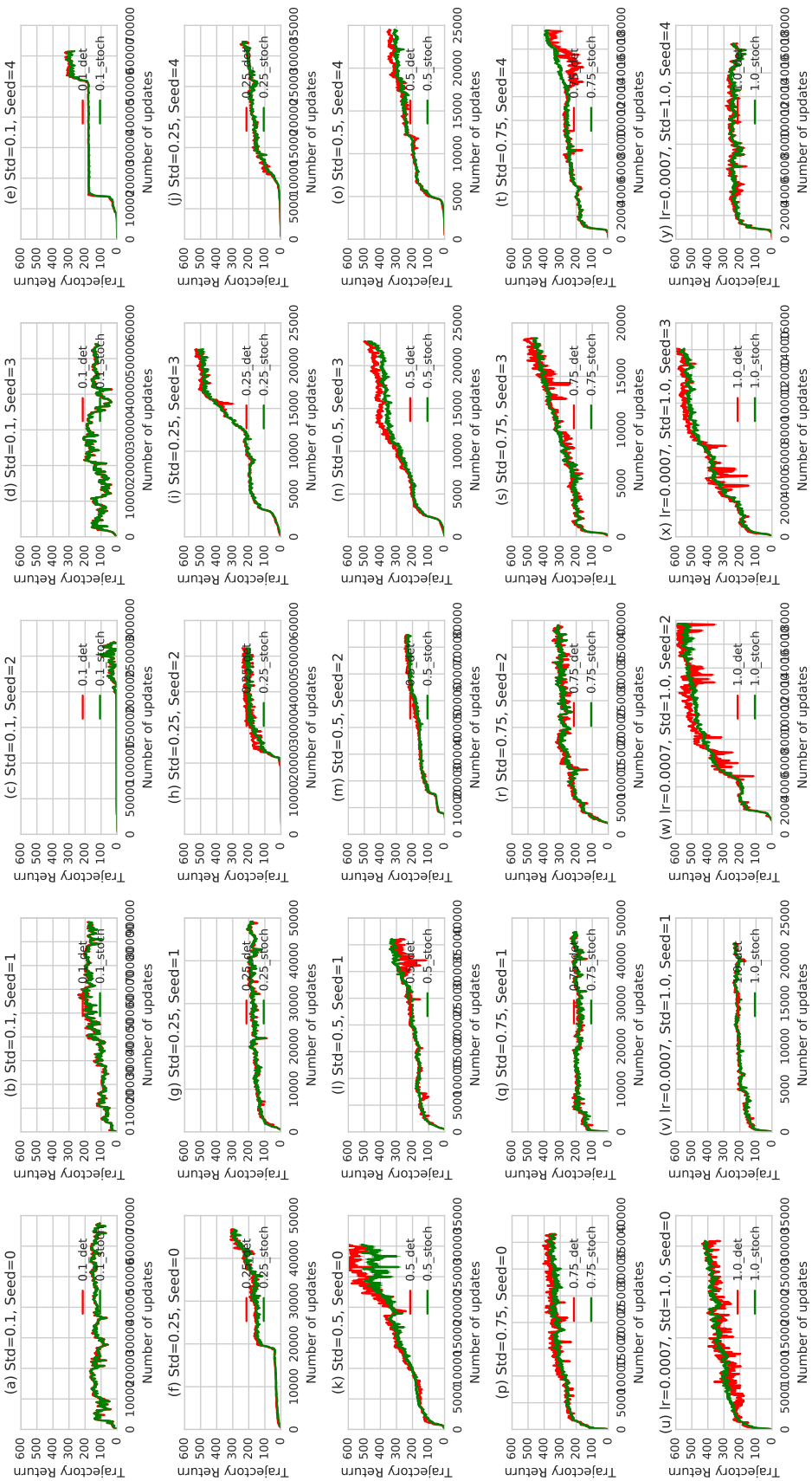


Figure S8. Individual learning curves for Walker

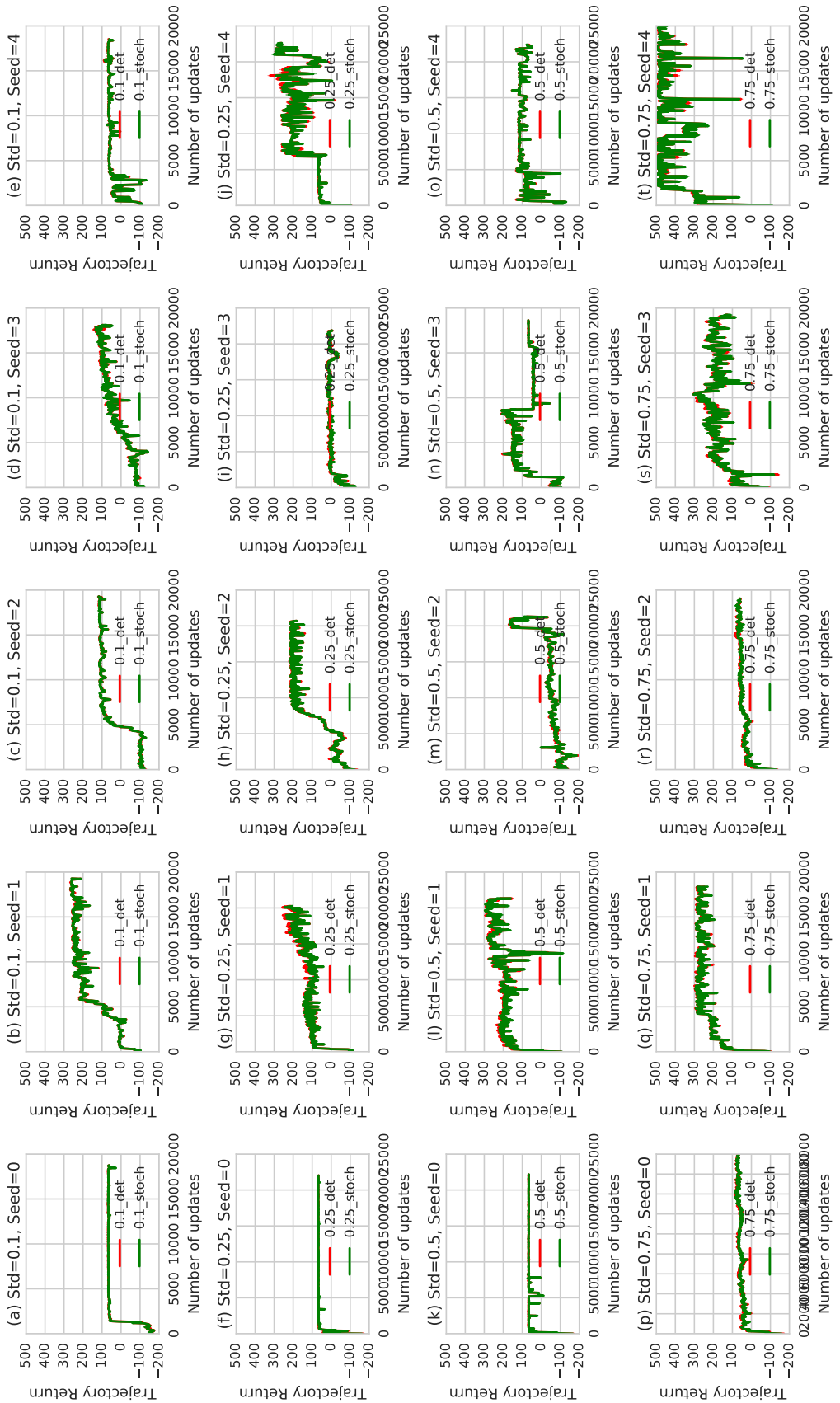
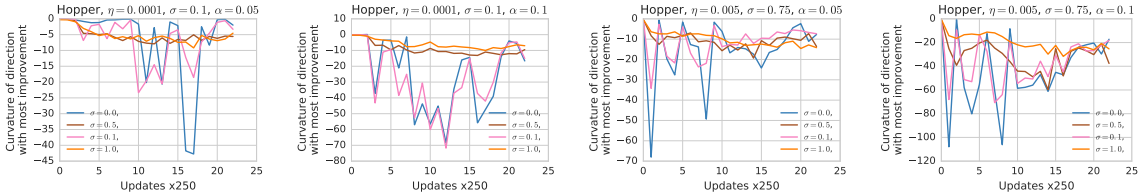
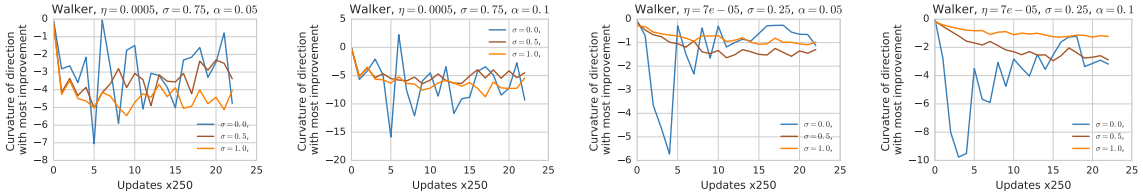


Figure S9. Individual learning curves for HalfCheetah



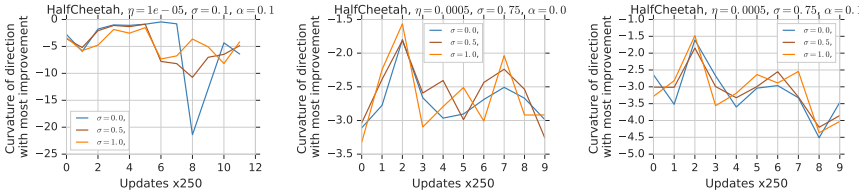
(a) (b) (c) (d)

Figure S10. Curvature for the direction with the most improvement during the optimization for different seeds and standard deviations in Hopper.



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

Figure S11. Curvature for the direction with the most improvement during the optimization for different seeds and standard deviations in Walker2d.



(a) (b) (c)

Figure S12. Curvature for the direction with the most improvement during the optimization for different seeds and standard deviations in HalfCheetah.

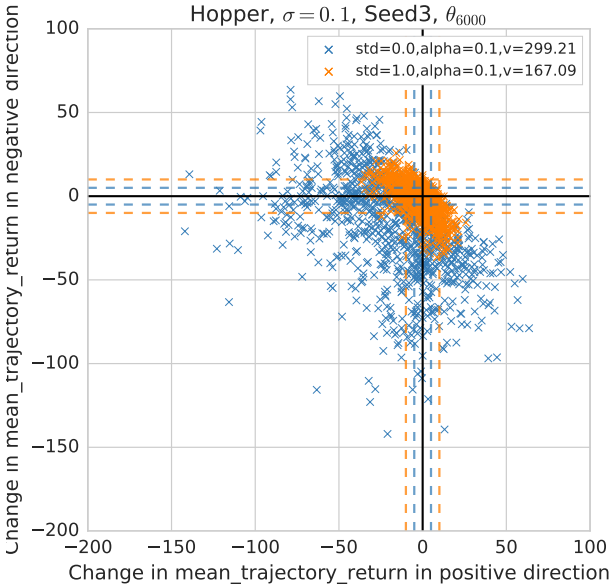
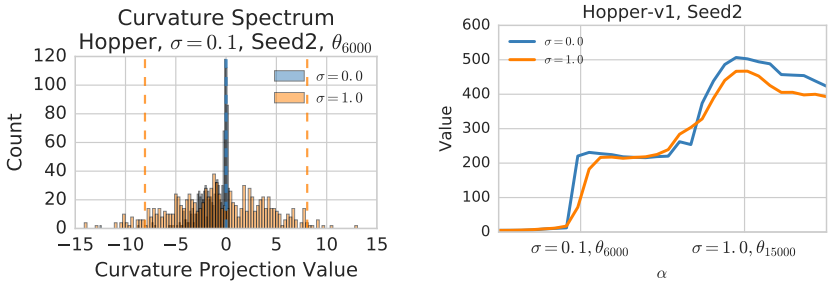
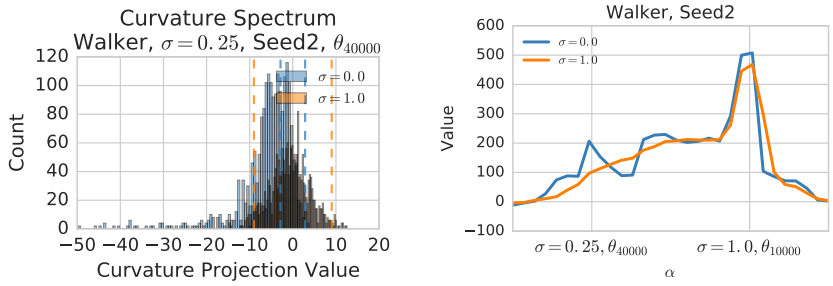


Figure S13. Scatter plot for randomly drawn directions for the solution shown in Figure 7. For $\sigma = 0$, most directions are negative and they all have near zero or negative curvature. For $\sigma = 1$, there are fewer negative directions, and more importantly less negative curvature: Indicating an almost linear region. This linear region can be seen in the 1D interpolation (Figure 7c)



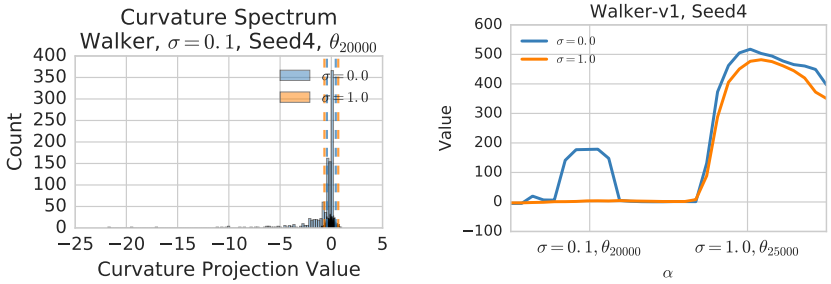
(a) (b)

Figure S14. Curvature spectra and linear interpolation for a solution in Hopper. (a) For $\sigma = 0$ most directions have a significant negative curvature implying that this solution is near a local optimum. For $\sigma = 1$ all curvature values are indistinguishable from noise. (b) An increasing path to a better solution exists but might be non-trivial for a line search to follow.



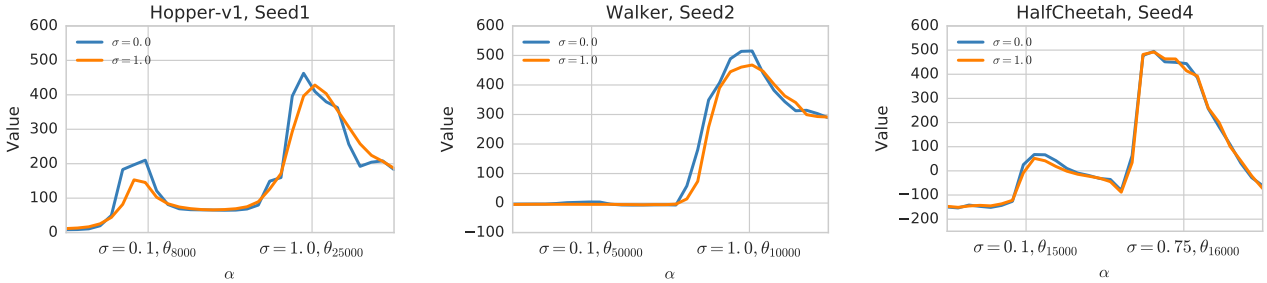
(a) (b)

Figure S15. Curvature spectra and linear interpolation for solutions in Walker2d. (a) For $\sigma = 0$ most directions have a significant negative curvature implying that this solution is near a local optimum. For $\sigma = 1$ all curvature values are indistinguishable from noise. (c) A monotonically increasing path to a better solution exists if we knew the direction to a solution a-priori.



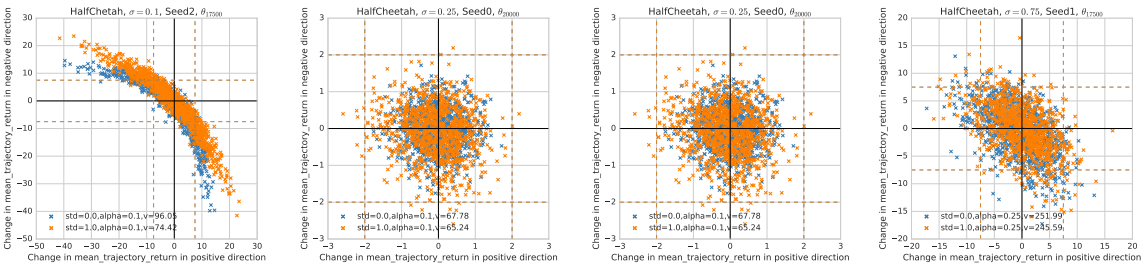
(a) (b)

Figure S16. Curvature spectra and linear interpolation for solutions in Walker2d. (a) 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 means that the solution ends up being in a flat region. (b) A linear interpolation confirms this observation in one dimension.



(a) (b) (c)

Figure S17. 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.



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

Figure S18. Local objective functions do not change much in HalfCheetah when σ is increased.