Effects of Task Distributions on Policy Optimization

Bhairav Mehta
Mila, Universite de Montreal

Jonathan Plante
Mila

Abstract

Domain randomization is a popular technique for improving domain transfer, often used in a zero-shot setting when the target domain is unknown or cannot easily be used for training. In this work, we empirically examine the effects of domain randomization, and more generally, task distributions and curricula on policy optimization. We study the effects of curricula on optimization, using tools from perturbation analysis and continual learning. Our analysis provides empirical evidence on the importance of a correct curriculum in multi-task reinforcement learning, even when an explicit curriculum of tasks is not available.

1 Introduction

Recent trends in Deep Reinforcement Learning (DRL) exhibit a growing interest for zero-shot domain transfer, i.e. when a policy is learned in a source domain and is then tested *without finetuning* in an unseen target domain. Zero-shot transfer is particularly useful when the task in the target domain is inaccessible, complex, or expensive, such as gathering samples from a real-world robot. An ideal agent would learn to *generalize* by accomplishing the tasks without exploiting irrelevant features or deficiencies in the source domain (i.e., approximate physics in simulators), which can vary dramatically after transfer and consequently lead to failures. When an agent fails the transfer task, we say that the agent falls prey to the *domain adaptation* problem.

A promising route for zero-shot transfer has been *domain randomization* [Tobin et al., 2017]. The approach is simple: when training a policy in a simulator, *uniformly* randomize every parameter of the simulation environment (e.g. friction, motor torque, etc.) across predefined ranges. This induces a *randomization space*, and generates many superficially-similar MDPs for an agent to train on.

To build upon domain randomization, Mehta et al. [2019] hypothesized that not all generated MDPs are equally useful for learning and proposed *Active Domain Randomization (ADR)*, which poses the search for informative randomization parameters as a reinforcement learning problem. ADR scores the usefulness of various randomized environment instances by comparing trajectory rollouts from a randomized simulation instance to rollouts from a reference simulation instance. Intuitively, the regions of the randomization space that generate distinguishable rollouts are of more interest and should be focused on during the training period, as they correspond to more difficult environments. The work showed empirical performance improvements over Domain Randomization (DR).

This report focuses on trying to understand *why* ADR works in practice. More generally, we focus on the effect of task distributions and curricula on policy optimization. We are interested in why certain distributions of tasks (or, if that distribution evolves over time, *curricula* of tasks) induce varying generalization performances. We will study this by characterizing the optimization trajectory and the optima found by each strategy using empirical perturbation methods.

2 Related Work

2.1 Domain Randomization

Domain randomization (DR) is a technique to train policies completely in simulation and transfer them in a zero-shot manner to the real world, often on a robotic platform. While a solution to an

Preprint. Part of larger, continued research project with L. Paull, C. Pal.

empirical issue, domain randomization induces interesting learning dynamics, as the *sequence* of tasks can be seen as an overarching, non-stationary Markov Decision Process (MDP). Non-stationarity in reinforcement learning can introduce dynamics that are difficult to analyze. In our work, we study empirical properties that can characterize policy solutions under this training scheme.

2.2 Interference and Transfer in Multi-Task Learning

Multi-task learning deals with training a single agent to do multiple things, such as a robotic arm trained to pick up objects and open doors. Often in multi-task learning, we aim to train a *single* policy that can accomplish everything, rather than training multiple, separate policies for each task. When sharing parameters, or more commonly, using the same network across tasks, gradients from various sources can *interfere* or *transfer*, hindering or enabling learning across the curriculum. Recently, Riemer and Tesauro [2018] have shown that when distinct tasks with minimal gradient interference are trained on, the resulting agent transfers better in a multi-task learning scenario.

When two separate tasks have gradients pointing in opposite directions, they can cancel out, or combine (average) out to a third direction that is not optimal for either task. When taking the average gradient step, it has been empirically shown by Chaudhry et al. [2018] that when gradients do not interfere, agents improve their multi-task performance. Ideas have been proposed that these gradients *align* better [Park and Oliva, 2019], but in our setting, we focus on the sampled gradients from the tasks without any transformation.

We can quantify *transfer* and *interference* of gradients using the cosine similarity of gradients from two tasks, T_a and T_b, as follows:

$$\rho_{ab}(\theta) = \frac{\langle \nabla J_{T_a}(\theta), \nabla J_{T_b}(\theta) \rangle}{||\nabla J_{T_a}(\theta)|| ||\nabla J_{T_b}(\theta)||} \tag{1}$$

where cosine similarity $\rho(\theta)$ is bounded between $+1$ (full transfer) and -1 (full interference).

2.3 Perturbation Analysis in Reinforcement Learning

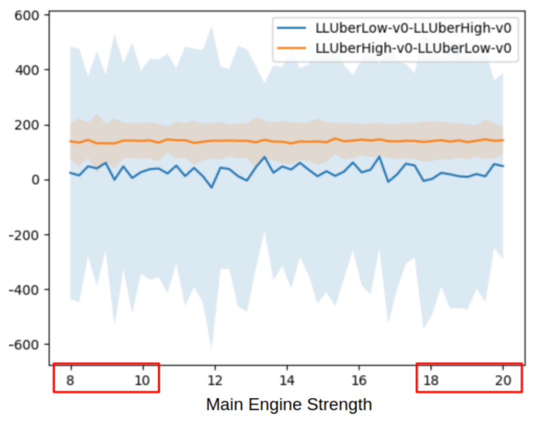
In policy optimization, we traditionally consider a Reinforcement Learning (RL) framework [Sutton and Barto, 2018] where some task T is defined by a MDP consisting of a state space S, action space A, state transition function $P : S \times A \mapsto S$, reward function $R : S \times A \mapsto \mathbb{R}$, and discount factor $\gamma \in (0, 1)$. The goal for an agent trying to solve T is to learn a policy π with parameters θ that maximizes the expected total discounted reward.

After training the policy in this maximization setting, ideally, any solution found by policy optimization would be some kind of local maximum. However, under a non-stationary MDP, standard policy optimization algorithms are no longer guaranteed to converge to any type of optima. In addition, when working with function approximators in high dimensions, optimization analysis of solutions is very difficult.

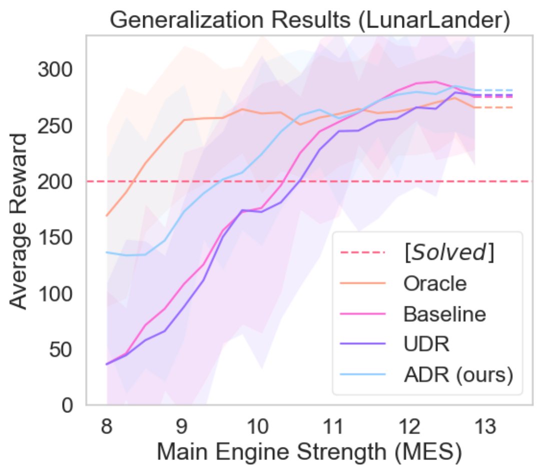
Recently, Ahmed et al. [2018] showed that with linear policies, we can analyze policy optimization solutions using pertubative methods. Briefly, they approximate the solution neighborhood as a local quadratic, and estimate the curvature of the neighborhood by perturbing the solution with a minor amount of noise. This allows to estimate the policy quality from the perturbations' effects. While the method produces a high variance estimate (we characterize the perturbation using accumulated reward as a metric), with enough noise samples, meaningful conclusions regarding solutions found with policy optimization can still be drawn.

3 Motivating Example

Before we can make wide claims about the effect of domain randomization, and more generally, curricula, on policy optimization, we need to ensure that the choice of curricula can truly impact optimization in a significant way. DR, as discussed, randomizes various parameters of the simulator, generating a multitude of MDPs for an agent to train on. We focus on a toy environment: `LunarLander-v2`, where the task is to ground a lander in a designated zone, and reward is based on the quality of the landing (fuel used, impact velocity, etc). `LunarLander-v2` has one main axis of



(a) Effects of curriculum order on agent performance. A wrong curriculum, shown in blue, can induce high variance, and overall poor performance.



(b) Generalization for various agents who saw different MES randomization ranges. Crossing the *Solved* line "earlier" on x-axis means more environments solved (better generalization).

Figure 1: Generalization curves from the two types of experiments.

randomization that we vary: the main engine strength (MES). Different values of MES change the MDP, and can be seen as different tasks. The MES, shown on the X-axis of Figure 1a, controls the strength of the landing engine for the agent, crucial for the task performance. When we move this value away from its default value of 13, we can generate superficially-similar MDPs that to an agent, are easier or more difficult to solve.

To study the effects of curricula on optimization, we artificially generate two "buckets" of tasks: *UberLow*, where the MES is uniformly sampled at every episode from the values of $[8, 10]$; and *UberHigh*, where the MES is uniformly sampled at every episode from the values of $[18, 20]$. The buckets of tasks are boxed in red in Figure 1a.

We train two agents, both for one million time-steps. The only difference between the two agents are the ordering of task buckets they see: for the first 500,000 steps (cumulative through episodes), one agent sees *UberLow* MDPs, whereas the other sees *UberHigh* MDPs. At the 500,000 step mark, the tasks are switched.

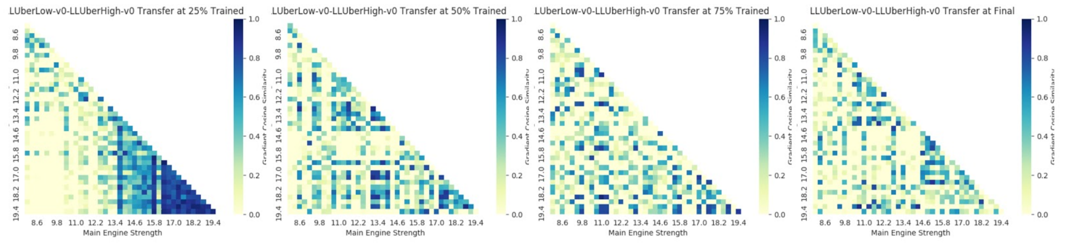
To simplify analysis, we use linear networks [Rajeswaran et al., 2017] trained with REINFORCE [Williams, 1992], which have been shown to perform well in simple continuous-control tasks. For every evaluation point, we train 10 seeds and generate 50 rollouts from each seed, totaling 500 independent evaluations for each point. We plot the mean average of all 500 runs, and shade in one standard deviation. To evaluate policy generalization and quality, we sweep across the entire MES randomization range of [8, 20] and roll out the final policy in the generated environments.

Figure 1a shows the results obtained from both experiments. The blue curve shows the results of generalization when the agent was shown *UberLow*, then *UberHigh*, where as the orange curve shows the switched curriculum. Curriculum seems to have a strong effect on stability of optimization, as well as average performance. These results hint at the notion of an underlying *optimal curriculum*, or an optimal sequence of tasks that an agent should train on. However, as this is a very strong assumption, in the following section, we explore the optimization of the two agents, and generate hypotheses to explain the vast difference in stability between the two.

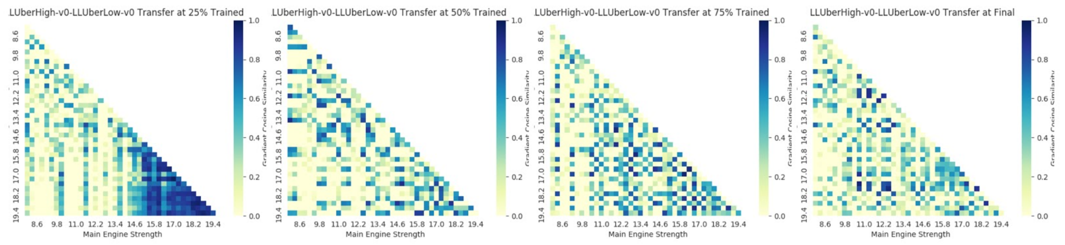
3.1 Gradient Interference in a Two-Task Setting

As noted in Section 2.2, continual learning researchers have long been aware of the issues of conflicting gradients across tasks and how they can prematurely halt learning, even when a network's capacity has not saturated [Kirkpatrick et al., 2017]. In this section, we take our toy example of "two" tasks (which are actually buckets of related tasks) and explore the compatibility of gradients through time.

To quantify *compatibility*, we use Equation 1 and take the two agents described above at various times through training, and generate heatmaps that show positive cosine similarity between gradients sampled from two tasks (shown on each axis of the heatmap). Briefly, we take an agent policy, sweep it across the full $[8, 20]$ range, and calculate the sampled gradients from that task (i.e a particular value for MES). We average these gradients over 500 independent runs, and use Equation 1 to generate a heatmap by performing an all-pairs comparison between gradients, shown in Figure 2.



(a) A bad curriculum, *UberLow* then *UberHigh*, as shown in the blue curve in Figure 1a, seems to generate patches of incompatible tasks, earlier in training, as shown by growing amounts of yellow.



(b) A good curriculum, *UberHigh* then *UberLow*, as shown in the orange curve in Figure 1a, seems to maintain better transfer through training, as shown by less patches of yellow in later stages of optimization (see Panel 3).

Figure 2: Gradient interference over time.

In the heatmaps shown in Figure 2, we plot the *positive* cosine similarity as dark blue ($\rho_{ab} = +1$). This scheme allows the more important metric, interference, to be grouped together into the yellow regions of the plots ($\rho_{ab} \leq 0$). When a point on the plot is colored yellow, we can denote the two tasks (on x and y axis) as *incompatible*, meaning that the two tasks' gradients interfere with one another. Keeping in mind that these are all high variance estimates, we focus on *patches* of incompatible tasks, which are more likely to actually be incompatible than randomly dispersed points of yellow.

In general, we find that the blue curve, the agent that sees *UberLow* then *UberHigh*, shows large, consistent patches of incompatible tasks much earlier in training than its counterpart. In addition, we find that these patches are consistent throughout training, which is not the case of the alternate-curriculum agent (compare both Panel 3s, Figure 2).

4 Increasing Complexity

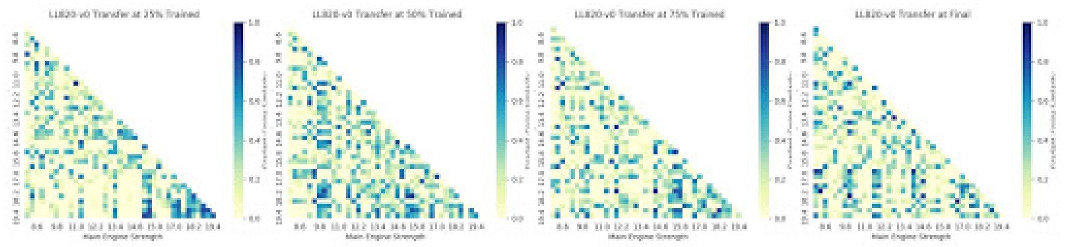
While interesting, the experiment detailed in the previous section is unrealistic. One advantage that domain randomization has over continual learning is that explicit tasks (or in our case, "buckets" of tasks) need not be defined; continual leaning research has often been criticized for engineering implicit task curricula based on difficulty (See OpenReview discussion of Riemer and Tesauro [2018]). On the contrary, domain randomization's random sampling breaks these types of sequences, treating each task on an equal level throughout training.

However, as we have seen above, even in domain randomization, the notion of a curriculum is helpful for optimization. In this section, we study a more realistic version of domain randomization: one where "buckets" of tasks are not designated. Rather, we specify only the general ranges from which to vary each environment parameter.

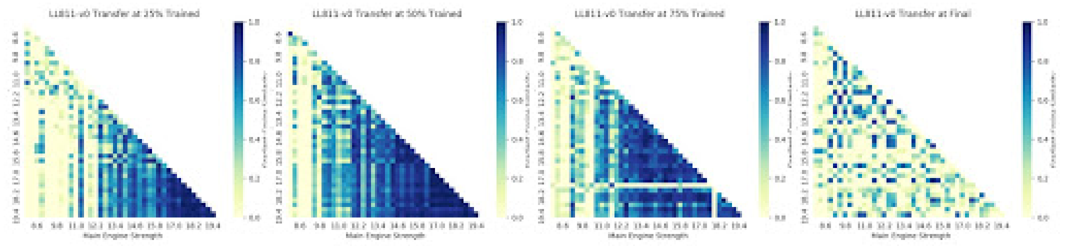
4.1 Effect of Curriculum with Domain Randomization

In `LunarLander-v2`, we continue to vary the main engine strength (MES). Again, we evaluate the generalization performance of each of our curriculum choices, and draw conclusions of our optimization analyses conditioned on what we know works empirically.

We train multiple agents, with the only difference being the randomization ranges for MES during training. Figure 1b shows the final generalization performance of each agent by sweeping across the entire randomization range of [8, 20] and rolling out the policy in the generated environments. We see that focusing on certain MDPs (labeled as *Oracle*, MES $\sim U(8, 11)$) improves generalization over traditional DR (labeled as *UDR*, MES $\sim U(8, 20)$), even when the evaluation environment is outside of the training distribution.



(a) Policies trained with traditional domain randomization show large patches of incompatible tasks that show up early in training, potentially leading to high variance in convergence, as noted by Mehta et al. [2019] and OpenAI et al. [2018].



(b) Policies trained with focused domain randomization seem to exhibit high transfer between tasks, which seems to enable better overall generalization by the end of training.

Figure 3: Gradient interference over time for agents who see different domain randomization ranges.

In Figure 3, we compare the gradient incompatibilities through training when an agent sees a random sample from the full range (MES $\sim [8, 20]$) or a sample from the subset of the range (MES $\sim [8, 11]$) (top and bottom Figure 3 respectively). Interestingly, we see the same types of patterns as the experiment described in Section 3.1. The bad curriculum (Figure 3a), which in this case is *traditional domain randomization* (MES $\sim [8, 20]$), shows large patches of incompatible tasks early in training. The agent exposed to *focused domain randomization* (MES $\sim [8, 11]$, Figure 3b) shows high task compatibility throughout training, until a final solution is found [1].

As shown in Figure 1b, we find that high task compatibility throughout training correlates with higher empirical generalization performance, although this statement needs to be rigorously tested before further claims are made.

4.2 Analyzing Active Domain Randomization

Active Domain Randomization (ADR) was motivated by the fact that the ranges shown in *focused domain randomization* will not generally be known, so an adaptive algorithm was proposed that focused on finding environment instances (a curriculum) that highlighted the agent policy's current weaknesses. Posed as a higher level reinforcement learning problem, ADR showed empirical

[1] An investigation on why even the good agent shows large patches of incompatible tasks at the end of training is needed. A simple explanation might be that the final solution has nestled into a local optima.

improvements over standard domain randomization in zero-shot learning settings, including sim2real robotic transfer. In Figure 1b, we see that ADR approaches generalization levels of the Oracle.

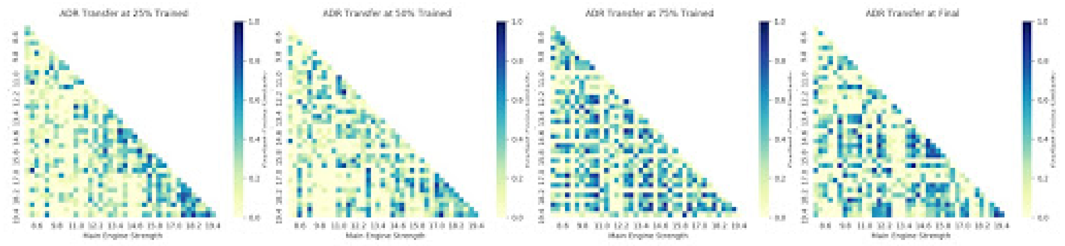


Figure 4: Active Domain Randomization shows its ability to adapt and bring the policy back to areas with less inter-task interference, as shown between Panel 2 and Panel 3.

We can see a different visualization of ADR's adaptiveness in Figure 4; by learning a curriculum, we can recover and move out from regions of policy space that show large amounts of interference, as seen by the gradual decrease in task incompatibility over time (specifically between Panel 2 and Panel 3, or at 50% and 75% of training completion respectively).

5 Anti-Patterns in Optimization

However, this initial investigation of learning dynamics has not resolved all problems in continual and multi-task learning. In fact, in this section, we show that these learning schemes produce counter-intuitive *final* solutions, despite the fact that their learning dynamics point to gradient interference being the main culprit.

Using techniques from Ahmed et al. [2018], we use perturbative methods to characterize the final solutions found by each agent. While subject to many conditions and constraints, in general, we expect (a) better performing policies to be at a *local maxima* (as discounted return maximization is a maximization problem) and (b) better performing policies to have "smoother" optimization paths.

However, when we analyze our solutions, shown in Figure 5 and 6 in Appendix A, we get counter-intuitive results. Worse performing policies seem to be at local maximas or saddle points, but more importantly, our better performers seem to be at local minimas. In addition, when we plot the curvature of the direction of highest improvement (whose volatility seems to be a good metric for optimization smoothness), we find that the better performing policies seem to be more volatile. While only a single, representative seed is shown here, the results across many seeds seem to be consistent.

6 Conclusion and Future Work

Studying learning dynamics in multi-task learning is an exciting area of research. Gradient interference is known to show up even when explicit multi-task dynamics are not present [Schaul et al. [2019]], but in a multi-task setting, we can take advantage of these dynamics for better optimization. We have provided an empirical look at the importance of curricula in policy optimization, and have reinforced the need of research regarding (the learning of) *optimal curricula*. Excitingly, our experiments here have shown that the premise of Active Domain Randomization (searching for informative environments can be modelled as a higher level MDP) is an approximation at best. Section 3.1 shows us that there seems to be a *time dependence* on the curriculum of environments, which does not maintain the Markov property (else, the two curricula would produce the same results, despite what order they were shown to the agent). As we continue this work, we are excited to view this problem from a dynamical systems lens, which we are actively pursuing with collaborators. We hope that this report encourages research on the dynamics of policy optimization under these conditions.

References

Z. Ahmed, N. L. Roux, M. Norouzi, and D. Schuurmans. Understanding the impact of entropy on policy optimization. *CoRR*, abs/1811.11214, 2018. URL `http://arxiv.org/abs/1811.11214`.

A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. S. Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *The European Conference on Computer Vision (ECCV)*, September 2018.

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. ISSN 0027-8424. doi: 10.1073/pnas.1611835114. URL `https://www.pnas.org/content/114/13/3521`.

B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. *CoRR*, abs/1904.04762, 2019. URL `http://arxiv.org/abs/1904.04762`.

OpenAI, :, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation, 2018.

E. Park and J. B. Oliva. Meta-curvature. *CoRR*, abs/1902.03356, 2019. URL `http://arxiv.org/abs/1902.03356`.

A. Rajeswaran, K. Lowrey, E. Todorov, and S. Kakade. Towards generalization and simplicity in continuous control. *CoRR*, abs/1703.02660, 2017. URL `http://arxiv.org/abs/1703.02660`.

A. L. R. T. Riemer, Cases and Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*, 2018.

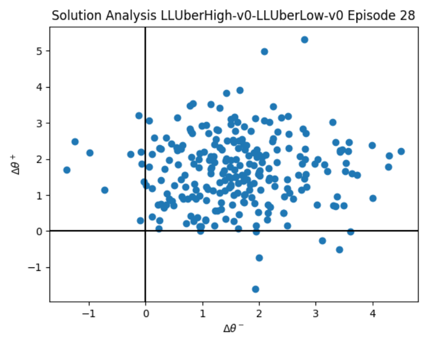
T. Schaul, D. Borsa, J. Modayil, and R. Pascanu. Ray interference: a source of plateaus in deep reinforcement learning, 2019.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

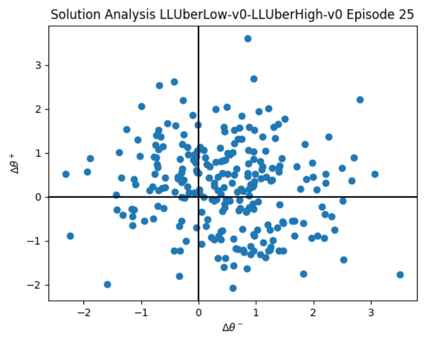
J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 23–30. IEEE, 2017.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

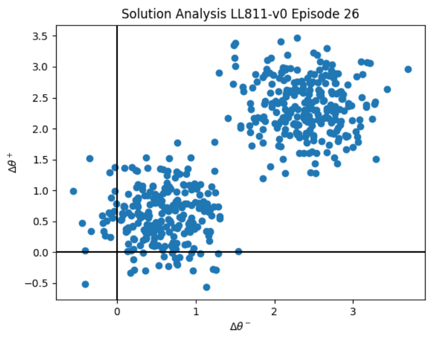
A Figures for Section 5



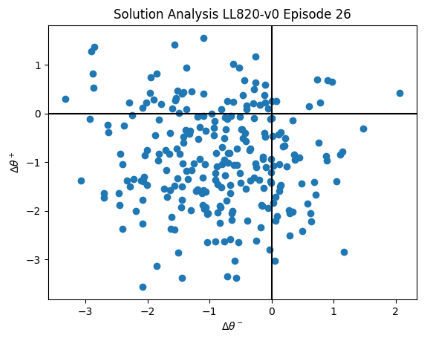
(a) Final policies trained with *UberHigh, UberLow* (the orange curve in Figure 1a), seem to be local minima.



(b) Final policies trained with *UberLow, UberHigh* (the blue curve in Figure 1a), seem to be saddles.

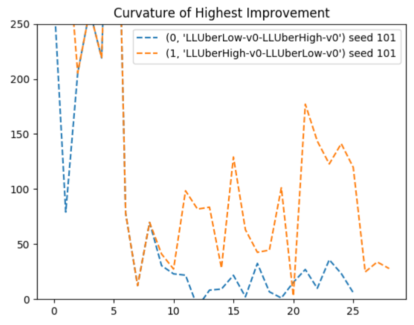


(c) Final Policies trained with MES $\sim U(8, 11)$ (Oracle in Figure 1b) seem to be local minima.

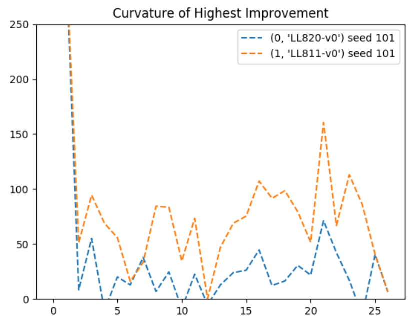


(d) Final Policies trained with MES $\sim U(8, 20)$ (UDR in Figure 1b) seem to be saddles.

Figure 5: Optimization analysis of final solutions for various curricula.



(a) The worse performing agent (blue curve, *UberLowUberHigh*), seems to have a smoother optimization path than its more better-performing, flipped-curriculum counterpart.



(b) Policies trained with full domain randomization (blue curve), also empirically shown to be less stable in practice, show smoother optimization paths.

Figure 6: Curvature analysis through training for various curricula

.