Towards Exploiting Geometry and Time for Fast Off-Distribution Adaptation in Multi-Task Robot Learning

K.R. Zentner* kzentner@usc.edu
Ryan Julian* rjulian@usc.edu
Ujjwal Puri ujjwalpu@usc.edu
Yulun Zhang yulunzha@usc.edu

Gaurav Sukhatme
gaurav@usc.edu
University of Southern California
Los Angeles, CA 90089

Abstract

We explore possible methods for multi-task transfer learning which seek to exploit the shared physical structure of robotics tasks. Specifically, we train policies for a base set of pre-training tasks, then experiment with adapting to new off-distribution tasks, using simple architectural approaches for re-using these policies as black-box priors. These approaches include learning an alignment of either the observation space or action space from a base to a target task to exploit rigid body structure, and methods for learning a time-domain switching policy across base tasks which solves the target task, to exploit temporal coherence. We find that combining low-complexity target policy classes, base policies as black-box priors, and simple optimization algorithms allows us to acquire new tasks outside the base task distribution, using small amounts of offline training data.

1 Introduction

Real world robotics tasks all share the rich and predictable structure imposed by the laws of physics and the nature of the physical world. The breakout success of deep learning approaches to domains such as computer vision, natural language processing, and recommender systems was precipitated by the design of model architectures which exploit the structure of the data in these domains, such as convolutional, auto-regressive, and graph neural networks respectively. Despite these strong precedents, research in deep reinforcement learning (RL) and imitation learning (IL) for robotics has seen comparatively few attempts to design robotics-specific architectures transfer learning methods which exploit the structure of robotics tasks. We believe that multi-task robot learning in particular is likely to benefit from transfer methods which exploit large amounts of shared physical and temporal structure between tasks, because this structure is very likely to exist between tasks performed by a single robot design which is asked to perform many tasks in just one or a few environments.

2 Problem Setting

We consider a multi-task reinforcement learning (RL) or imitation learning (IL) setting, defined by a possibly-unbounded discrete space of tasks $\mathcal{T}$. Each task $\tau \in \mathcal{T}$ is an infinite-horizon Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r_\tau)$. Motivated by our application to robotics,

*Equal Contribution

Challenges of Real World Reinforcement Learning, Virtual Workshop at NeurIPS 2020

we presume all tasks in $\mathcal{T}$ share a single continuous state space $\mathcal{S}$, continuous action space $\mathcal{A}$, and state transition dynamics $p(s'|s,a)$, and so tasks are differentiated only by their reward functions $r_\tau(s,a)$. Our goal is to eventually learn policies $\pi_\tau(a|s)$ for each of $\tau \in \mathcal{T}$ which maximizes the expected total discounted return across all tasks $\tau \in \mathcal{T}$.

Importantly, we do not presume that the learner ever has access to all tasks in $\mathcal{T}$ at once, or even a representative sample thereof. Instead, we divide the lifetime of the learner into two phases. First pre-training, in which the learner is given access to a biased subset of tasks $\mathcal{T}_{\text{base}}$ and allowed to learn a near-optimal policy $\pi_\tau(a|s)$ for each task in $\mathcal{T}_{\text{base}}$, for a total of $|\mathcal{T}_{\text{base}}| = N_{\text{base}}$ *base policies*. Second, adaptation, in which the learner is given access to data source $\mathcal{D}_{\text{target}}$ of trajectories for a target task $\tau_{\text{target}} \notin \mathcal{T}_{\text{base}}$ and base policies $\pi^{\tau \in \mathcal{T}_{\text{base}}}(a|s)$, and must quickly acquire a high-performance *target policy* π^{target} for the target task. As $\mathcal{T}_{\text{base}}$ is presumed to be a biased data-set with respect to $\mathcal{T}$, this is an off-distribution multi-task learning problem.

In this work, we presume $\mathcal{D}_{\text{target}}$ is static, small, and composed only of successful trajectories from an expert, so herein we discuss a few-shot imitation learning variant of this problem.

3 Target Policy Classes for Structural Adaptation

In this section, we enumerate a set of low-dimensional target policy classes for fast adaptation between base and target tasks which share similar dynamical structure. In the next section, we measure the performance of these model classes under few-shot adaptation using a simulated 2D car-driving environment with non-trivial dynamics.

3.1 Observation Alignment

Observation alignment uses a target policy class that contains a single source policy. This target policy class applies a transformation $T_\theta(s)$ to the observation before passing it to the base policy to generate actions. In our experiments we use an affine (linear transformation plus bias) transformation of the observation to simplify optimization, and to exploit simple geometric priors such as rigid transformation. As the optimization process is efficient, we choose a base policy by training a $T_\theta^\tau(a)$ for all $\tau \in \mathcal{T}_{\text{base}}$ and using the lowest loss member of the ensemble for the final target policy.

$$\pi_\theta^{\text{target}}(a|s) = \pi^{\text{base}}(a|T_\theta(s)) \tag{1}$$

3.2 Action (Re-)Alignment

Like observation alignment, action alignment learns a low-dimensional affine transformation function, but instead transforming the state input of a base policy, this function $T_\theta(a)$ transforms the action output. Like observation alignment, we train an ensemble of these target policies for a target task, and choose the one with the lowest loss.

$$\pi_\theta^{\text{target}}(a|s) = \pi^{\text{base}}(T_\theta(a)|s) \tag{2}$$

We find that naive action alignment performs poorly, because the final output layer of π^{base} often destroys necessary information. Action re-alignment instead uses all but the last layer of the base policy (hereafter referred to as π_{-1}^{base}) to encoder the current state into a latent encoding h. Action re-alignment then uses a learned transformation function $T_\theta(h)$ to transform that encoding into an action on the target task.

$$\pi_\theta^{\text{target}}(a|s) = \pi_{-1}^{\text{base}}(T_\theta(h)|s) \tag{3}$$

3.3 Time-Domain Switching and Mixing

3.3.1 Mixing (Soft Switching)

In order to select actions in the target task, soft switching policy computes a state-conditioned weighting function over the base policies, $W_\theta(s,\tau)$. It then computes an action distribution as a

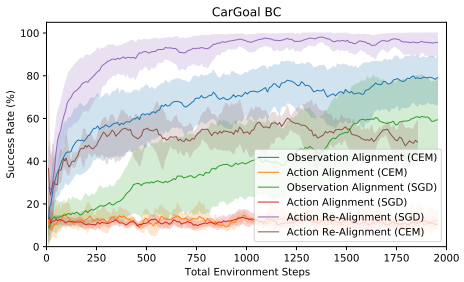

Figure 1: This figure shows the performance of different target policy classes and optimization methods using a behavioral cloning loss. The shaded regions represent a 95% confidence interval. The highest performing methods are Action Re-Alignment with $T_\theta(h)$ trained using SGD, and Observation Alignment with $T_\theta(s)$ trained using CEM. This environment takes roughly 200,000 time steps to solve using PPO.

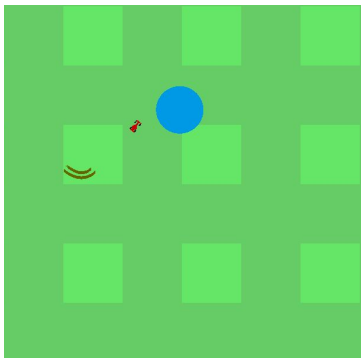

Figure 2: A screenshot of the CarGoal environment. The goal region shown in the image is not observable by the policy.

mixture of the base policies action distributions using that weighting.

$$\pi_\theta^{\text{target}}(a|s) = \sum_{\tau \in \mathcal{T}_{\text{base}}} W_\theta(s,\tau)\pi^{\text{base},\tau}(a|s) \tag{4}$$

3.3.2 Hard Switching

As in soft switching, this target policy class computes a state-conditioned weighting function $W_\theta(s,\tau)$ over the base policies. Unlike soft switching, the hard switching policy uses actions from only a single base policy at any given time step. To discourage switching too often, the hard switching policy enforces hysteresis during sampling: it maintains a state, h of the the most recently selected base policy, and continues to use that base policy until another base policy has a weight which is ϵ larger than the current base policy.

$$\pi_\theta^{\text{target}}(a|s,h) = \pi^{\text{base},h'}(a|s) \tag{5}$$

$$\text{where} \quad h' = \begin{cases} h & \text{if } W_\theta(s,h) + \epsilon > W_\theta(s,\tau)\ \forall \tau \in \tau_{\text{base}} \\ \text{argmax}_\tau\ W_\theta(s,\tau) & \text{otherwise} \end{cases}$$

4 Experiments

4.1 Training $T_\theta(s)$, $T_\theta(a)$, and $T_\theta(h)$

For this work, we have limited ourselves to affine T transformations (i.e. $T(x) = Ax + b$ and $\theta = (A, b)$), which allows us to exploit a geometric prior common in robotics: that states and actions typically represent the position or velocity of rigid bodies in SE(3), or other grounded physical quantities which can be aligned between tasks using an affine transformation.

We find that gradient-based RL and IL methods (BC, AWR, and PPO) can encounter local optima and have difficulty training such low-dimensional function approximators. However, we find that the Cross-Entropy Method (CEM) can be used to reliably train the parameters of T. Furthermore, by using CEM with a behavioral cloning loss, instead of Monte-Carlo estimates of expected returns, we are able to re-use a small number of demonstration timesteps to train as success target policy using roughly 2000 timesteps of expert demonstrations in the CarGoal environment. More details of these results can be seen in Figure 1.

4.2 Training $W_\theta(s,\tau)$

In our experiments, we use a fully connected neural network followed by softmax to approximate $W_\theta(s,\tau)$. We find that it's possible to infer θ by minimizing a standard behavioral cloning loss (BC) across the target dataset $\mathcal{D}_{\text{target}}$. However, we would prefer target policies which switch between base policies less frequently. To discourage switching, we regularize the standard cross-entropy loss between $W_\theta(s,\tau)$ and the posterior action probabilities $\pi^{\text{base},\tau}(s,a)$, using the cross-entropy loss between $W_\theta(s,\tau)$ and $W_\theta(s',\tau)$, the weights on adjacent states under the target dataset, and combine

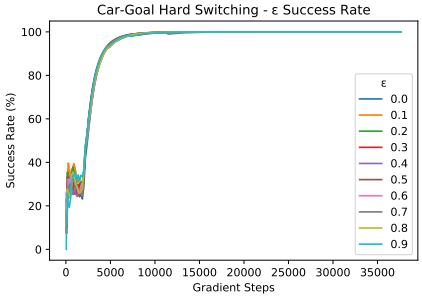

Figure 3: This figure shows the shows the performance of hard switching during the training process for a variety of ϵ values. Note that $\epsilon = 0$ performs equivalently to soft-switching. The base policies are three policies trained to reach different goal regions in CarGoal. The target task goal region is outside of the convex hull of those goal regions.

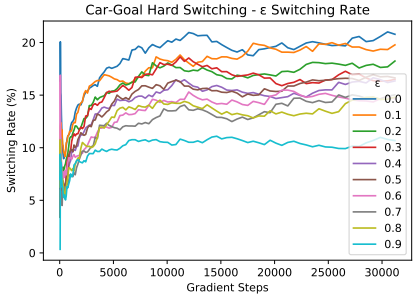

Figure 4: This figure shows the average switching rate decreases as epsilon is increased. In conjunction with Figure 3, it shows that a policy trained using our loss function can switch policies relatively slowly without any visible decrease in the success rate. In this experiment, $\alpha = 0.9$.

these terms using a coefficient α, which we typically set slightly below 1. The effectiveness of this loss function can be seen in Figure 3 and Figure 4.

$$L(\theta, s, a, s') = \alpha \mathcal{CE}[W_\theta(s, \tau), \pi^{\text{base},\tau}(a|s))] + (1 - \alpha)\mathcal{CE}[W_\theta(s, \tau), W_\theta(s', \tau)]$$

4.3 Environment

We use a simple goal-conditioned toy environment we call CarGoal, which is based on the CarRacing environment from OpenAI Gym [1]. The objective of CarGoal is to drive a car to a goal point in the environment, which is hidden from the policy. The policy is rewarded for pointing the car towards the goal, getting the car near the goal, and reaching the goal region. Reaching the goal region within 1000 time steps is considered "success," and terminates the episode. Different tasks in this environment correspond to different goal points. A screenshot of CarGoal is shown in Figure 2.

5 Conclusion

Our results show that the combination of simple architectural approaches, a base set of black-box policies, and simple optimization algorithms like CEM and SGD can produce strong off-distribution transfer results in an environment with non-trivial dynamics. In the future, we look forward to applying these ideas to much more complex problems using real and simulated robotics tasks, and using them to design algorithms for continual robot learning.

6 Related Work

Both recent [2][3] and less-recent [4],[5] work have explored automatically identifying and exploiting structure, especially symmetries, in general MDPs to speed learning, outside of the transfer learning setting. Structural priors, such as symmetry, temporal coherence, and rigid body transformations, have been previously used successfully to adapt deep learning methods to the robotics domain [6][7]. Prior works in this area have mostly focused on learning sparse and informative state representations, rather than adapting policies to new tasks using these priors.

Time-domain composition of sub-policies has been studied extensively in hierarchical reinforcement learning, most notably by works using the options framework [8] in which RL sub-policies (options) choose their own termination conditions. Other works have studied a problem setting that is more comparable to our own, in which the subpolicies are chosen by a higher-level control process [9][5]. Most work in this area has focused on fast transfer by conditioning learned policies on pre-defined goal spaces [10] or grounded representations such as language [11] or known object identities [12].

The cross-entropy method [13] has been used both for direct policy search [14], and more recently as an inner search component of value-based RL algorithms, such as to choose actions given a continuous Q-value approximation [15], and to regularize a policy gradient-based action selection algorithm [16].

Our work is most similar in spirit to Residual Policy Learning [17], which also uses a pre-trained policy in one task for structured exploration in another task, and uses a deterministic policy target model class of the form $\pi_{\text{target}}(s) = \pi_{\text{base}}(s) + f_\delta(s)$ to facilitate fast adaptation.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[2] Elise van der Pol, Daniel E Worrall, Herke van Hoof, Frans A Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *arXiv preprint arXiv:2006.16908*, 2020.

[3] Elise van der Pol, Thomas Kipf, Frans A Oliehoek, and Max Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. *arXiv preprint arXiv:2002.11963*, 2020.

[4] B. Ravindran and A. G. Barto. Symmetries and model minimization in markov decision processes. Technical report, USA, 2001.

[5] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, page 720–727, New York, NY, USA, 2006. Association for Computing Machinery.

[6] Rico Jonschkowski and Oliver Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[7] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017.

[8] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[9] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[10] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 3303–3313, 2018.

[11] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 9419–9431, 2019.

[12] Amy Zhang, Sainbayar Sukhbaatar, Adam Lerer, Arthur Szlam, and Rob Fergus. Composable planning with attributes. In *International Conference on Machine Learning*, pages 5842–5851, 2018.

[13] Reuven Y Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

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

[15] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

[16] Lin Shao, Yifan You, Mengyuan Yan, Qingyun Sun, and Jeannette Bohg. Grac: Self-guided and self-regularized actor-critic. *arXiv preprint arXiv:2009.08973*, 2020.

[17] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.