Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers

Anonymous Authors[1]

Abstract

We propose a simple, practical, and intuitive approach for domain adaptation in reinforcement learning. Our approach stems from the idea that the agent's experience in the source domain should look similar to its experience in the target domain. Building off of a probabilistic view of RL, we formally show that we can achieve this goal by *compensating for the difference in dynamics by modifying the reward function*. This modified reward function is simple to estimate by learning auxiliary classifiers that distinguish source-domain transitions from target-domain transitions. Intuitively, the modified reward function penalizes the agent for visiting states and taking actions in the source domain which are not possible in the target domain. Our approach is applicable to domains with continuous states and actions and does not require learning an model of the dynamics.

1. Introduction

Reinforcement learning (RL) is often touted as a promising approach for costly and risk-sensitive applications, yet learning to act in those domains directly is costly and risky. How can an intelligent agent learn to solve tasks in environments in which it cannot practice? In this paper we study the problem of domain adaptation in reinforcement learning (RL). In the context of RL, domains refer to different environments (MDPs) that have different dynamics (transition functions). Our aim is to learn a policy in the source domain that will achieve high reward in a different target domain, using a limited amount of experience from the target domain.

RL algorithms today require a large amount of experience in the *target domain*. Experience in the target domain is expensive to collect: it costs time (e.g., when the target domain is the real world, we cannot progress faster than real-time); it costs money (e.g., a robot might break itself); it could even be dangerous to humans (Matsakis, 2018). For many tasks, such as assistive robotics and self-driving cars, we may have access to a different but structurally similar *source domain*. While the source domain has different dynamics than the target domain, experience in the source domain is much cheaper to collect. For example, a computer simulation of the real world can run much faster than real time, collecting (say) a year of experience in an hour; it is much cheaper to simulate 1000 robot manipulators in parallel than to maintain 1000 robot manipulators. The source domain need not be a simulator, but rather could be any "practice" facility, such as a "farm" of robot arms (Levine et al., 2018), a "playpen" for learning to walk (Raibert, 2019), or a testing facility for self-driving vehicles (Madrigal, 2018).

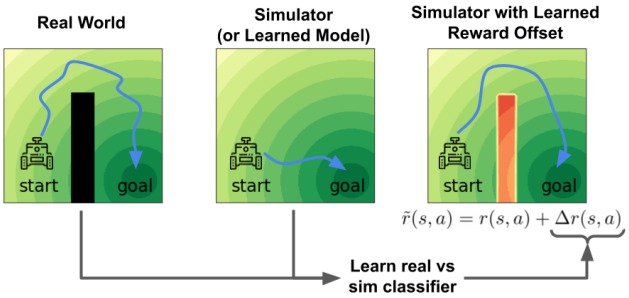

Figure 1. We will learn a policy for a target domain *(Left)* using experience from a source domain with different dynamics *(Center)*. *(Right)* Our method modifies the reward function to force the agent to learn behaviors that will be feasible in the target domain.

Domain adaptation in RL is challenging because strategies which are effective in the source domain may not be effective in the target domain. For example, a good approach to driving a car around a dry racetrack (the source domain) may entail aggressive acceleration and cutting corners. If the target domain is an icy, public road, this approach may cause the car to skid off the road or hit oncoming traffic. While prior work has thoroughly studied the domain adaptation of *observations* in RL (Bousmalis et al., 2018; Ganin et al., 2016; Higgins et al., 2017), it ignores the domain adaptation of the *dynamics*. For space constraints, we defer the discussion of related work to Appendix A.

This paper presents a simple and practical approach for domain adaptation in RL, illustrated in Fig. 1. Our ap-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

proach stems from the idea that the agent's experience in the source domain should look similar to its experience in the target domain. Building off of a probabilistic view of RL, we formally show that we can achieve this goal by *compensating for the difference in dynamics by modifying the reward function*. This modified reward function is simple to estimate by learning auxiliary classifiers that distinguish source-domain transitions from target-domain transitions. Because our method learns a classifier, rather than a dynamics model, we expect it to handle high-dimensional tasks better than model-based methods, a conjecture supported by experiments on the 111-dimensional Ant task. Intuitively, the modified reward function penalizes the agent for visiting states and taking actions where the source domain and target domain differ. The agent is penalized for taking transitions which would indicate whether the agent is interacting with the source or target domain. On a range of discrete and continuous control tasks, we both illustrate the mechanics of our approach and demonstrate its scalability to higher-dimensional tasks. broadly-applicable approach for learning from inaccurate models.

2. Preliminaries

Our problem setting will consider two MDPs: $\mathcal{M}_{\text{source}}$ represents the source domain (e.g., a practice facility, simulator, or learned approximate model of the target domain) while $\mathcal{M}_{\text{target}}$ represents a the target domain. We assume that the two domains have the same state space $\mathcal{S}$, action space $\mathcal{A}$, reward function r, and initially state distribution $p_1(s_1)$; the only difference between the domains is there dynamics, $p_{\text{source}}(s_{t+1} \mid s_t, a_t)$ and $p_{\text{target}}(s_{t+1} \mid s_t, a_t)$. We will learn a Markovian policy $\pi_\theta(a \mid s)$, parametrized by θ. Our objective is to learn a policy π that maximizes rewards on $\mathcal{M}_{\text{target}}$, $\mathbb{E}_{\pi,\mathcal{M}_{\text{target}}}[\sum_t \gamma^t r(s_t, a_t)]$. We now define our problem setting:

Definition 1. ***Domain Adaptation for RL*** *is the problem of using interactions in the source MDP $\mathcal{M}_{source}$ together with a small number of interactions in the target MDP $\mathcal{M}_{target}$ to acquire a policy that achieves high reward in the target MDP, $\mathcal{M}_{target}$.*

We will assume every transition with non-zero probability in the target domain will have non-zero probability in the source domain:

$$p_{\text{target}}(s_{t+1} \mid s_t, a_t) > 0 \implies p_{\text{source}}(s_{t+1} \mid s_t, a_t) > 0$$

for all states $s_t, s_{t+1} \in \mathcal{S}$ and actions $a_t \in \mathcal{A}$. This assumption is very weak, and common in work on importance sampling (Koller & Friedman, 2009, §12.2.2).

3. A Variational Perspective on Domain Adaptation in RL

The probabilistic inference interpretation of RL (Kappen, 2005; Todorov, 2007; Toussaint, 2009; Ziebart, 2010; Rawlik et al., 2013; Levine, 2018) treats the reward function as defining a desired distribution over trajectories. The agent's task is to sample from this distribution by picking trajectories with probability proportional to their exponentiated reward. This section will reinterpret this model in the context of domain transfer, showing that domain adaptation of *dynamics* can be done by modifying the *rewards*.

To apply this model to domain adaptation, define $p(\tau)$ as the desired distribution over trajectories in the target domain,

$$p(\tau) \propto p_1(s_1)\left(\prod_t p_{\text{target}}(s_{t+1} \mid s_t, a_t)\right)\exp\left(\sum_t r(s_t, a_t)\right),$$

and $q(\tau)$ as our agent's distribution over trajectories in the source domain,

$$q(\tau) = p_1(s_1)\prod_t p_{\text{source}}(s_{t+1} \mid s_t, a_t)\pi_\theta(a_t \mid s_t).$$

As noted in Section 2, we assume both trajectory distributions have the same initial state distribution. Our aim is to learn a policy whose behavior in the source domain both receives high reward and has high likelihood under the target domain dynamics. We codify this objective by minimizing the reverse KL divergence between these two distributions:

$$\min_{\pi(a|s), q(s'|s,a)} D_{\text{KL}}(q \parallel p) =$$
$$-\mathbb{E}_q\left[\sum_t r(s_t, a_t) + \mathcal{H}_\pi[a_t \mid s_t] + \Delta r(s_{t+1}, s_t, a_t)\right] + c,$$

where

$$\Delta r(s_{t+1}, s_t, a_t) \triangleq \log p(s_{t+1} \mid s_t, a_t) - \log q(s_{t+1} \mid s_t, a_t).$$

The constant c is the partition function of $p(\tau)$, which is independent of the policy and source dynamics. While Δr is defined as the difference of transition probabilities, in Sec. 4.1 we show how to estimate Δr without learning transition probabilities directly. In the special case where the source and target dynamics are equal, the correction term Δr is zero and we recover maximum entropy RL (Ziebart, 2010; Todorov, 2007). We emphasize that our reward correction is different from prior work that adds $\log \beta(a \mid s)$ to the reward to regularize the policy to be close to the behavior policy β (Schulman et al., 2015; Jaques et al., 2017; Schroecker & Isbell, 2020; Abdolmaleki et al., 2018; Jaques et al., 2019; Thananjeyan et al., 2020).

In the case where the source dynamics are *not* equal to the true dynamics, this objective is not the same as maximum entropy RL on trajectories sampled from the source domain. Instead, this objective suggests a corrective term Δr that should be added to the reward function to account for the discrepancy between the source and target dynamics. The correction term, Δr, is quite intuitive. If a transition (s_t, a_t, s_{t+1}) has equal probability in the source and target domains, then $\Delta r(s_t, a_t) = 0$ so no correction is applied.

Algorithm 1 Domain Adaptation with Rewards from Classifiers [DARC]

for $t = 1, \cdots$, num iterations **do**
  $\mathcal{D}_{\text{source}} \leftarrow \mathcal{D}_{\text{source}} \cup \text{ROLLOUT}(\pi, \mathcal{M}_{\text{source}})$
  if $t \mod r = 0$ **then**
    $\mathcal{D}_{\text{target}} \leftarrow \mathcal{D}_{\text{target}} \cup \text{ROLLOUT}(\pi, \mathcal{M}_{\text{target}})$
  $\theta \leftarrow \theta - \eta \nabla_\theta \ell(\theta)$
  $\tilde{r}(s_t, a_t, s_{t+1}) \leftarrow r(s_t, a_t) + \Delta r(s_t, a_t, s_{t+1})$
  $\pi \leftarrow \text{MAXENT RL}(\pi, \mathcal{D}_{\text{source}}, \tilde{r})$
return π

For transitions that are likely in the source but are unlikely in the target domain, $\Delta r < 0$, so the agent is penalized for "exploiting" inaccuracies or discrepancies in the source domain by taking these transitions. For the example environment in Figure 1, transitions through the center of the environment are blocked in the target domain but not in the source domain. For these transitions, Δr would serve as a large penalty, discouraging the agent from taking these transitions and instead learning to navigate around the wall. Appendix C presents additional interpretations of Δr in terms of coding theory, mutual information, and a constraint on the discrepancy between the source and target dynamics. Appendix B shows how prior work that adapts observation is a special case of our approach.

4. Domain Adaptation with Learned Rewards

The variational perspective on model-based RL in the previous section suggests that we should modify the reward in the source domain by adding Δr. While Δr is defined above in terms of transition probabilities, we will show below how it can be estimated via binary classification, without learning an explicit dynamics model. We then use this observation to develop a practical algorithm for off-dynamics RL.

4.1. Estimating the Reward Correction with Classifiers

The transition probabilities in the modified reward function are rarely known and are hard to estimate. Instead, we show that we can estimate this log ratio using a pair of (learned) binary classifiers, which will infer whether transitions came from the source or target domain by rewriting $\Delta r(s_t, a_t, s_{t+1})$ as

$$\log p(\text{target} \mid s_t, a_t, s_{t+1}) - \log p(\text{target} \mid s_t, a_t)$$
$$- \log p(\text{source} \mid s_t, a_t, s_{t+1}) + \log p(\text{source} \mid s_t, a_t)$$

The orange terms are the difference in logits from the classifier conditioned on s_t, a_t, s_{t+1}, while the blue terms are the difference in logits from the classifier conditioned on just s_t, a_t. Intuitively, Δr answers the following question: for the task of predicting whether a transition came from the source or target domain, how much better can you perform after observing s_{t+1}?

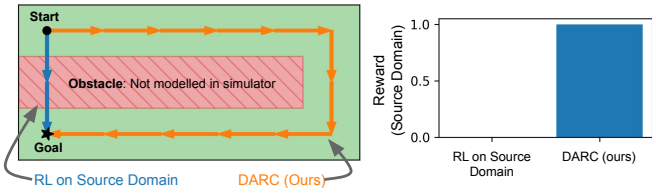

Figure 2. **Tabular example of off-dynamics RL**

4.2. Algorithm Summary

Our algorithm, Domain Adaptation with Rewards from Classifiers (DARC), is presented in Alg. 1. The algorithm modifies an existing MaxEnt RL algorithm to additionally learn two classifiers, $q_{\theta_{\text{SAS}}}(\text{target} \mid s_t, a_t, s_{t+1})$ and $q_{\theta_{\text{SAS}}}(\text{target} \mid s_t, a_t)$. We use the classifiers to modify the rewards from the *source* domain, and apply MaxEnt RL to this experience. We use SAC (Haarnoja et al., 2018) as our MaxEnt RL algorithm, but emphasize that DARC is applicable to any MaxEnt RL algorithm (e.g., on-policy, off-policy, and model-based).

5. Experiments

We start with a didactic experiment to build intuition for the mechanics of our method, and then evaluate on more complex tasks. Appendix D.3 includes additional experiments.

Illustrative example. We start with a simple gridworld example, shown on the right, where we can apply our method without function approximation. The goal is to navigate from the top left to the bottom left. The real environment contains an obstacle (shown in red), which is not present in the source domain. If we simply apply RL on the source domain, we obtain a policy that navigates directly to the goal (blue arrows), and will fail when used in the target domain. We then apply our method: we collect trajectories from the source domain and real world to fit the two tabular classifiers. These classifiers give us a modified reward, which we use to learn a policy in the source domain. The modified reward causes our learned policy to navigate around the obstacle.

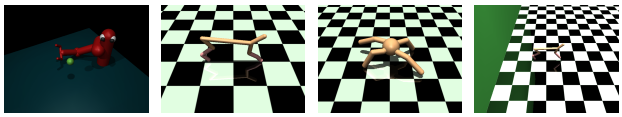

Figure 3. **Environments**: (L to R) broken reacher, broken half cheetah, broken ant, and half cheetah obstacle.

Scaling to more complex tasks. We now apply DARC to the more complex tasks shown in Fig. 3. We define three tasks by crippling one of the joints of each robot in the target domain, but using the fully-functional robot in the source domain. We use three simulated robots taken from OpenAI Gym (Brockman et al., 2016): 7 DOF reacher, half cheetah, and ant. The broken reacher is based on the task described by Vemula et al. (2020). We also include a task where the

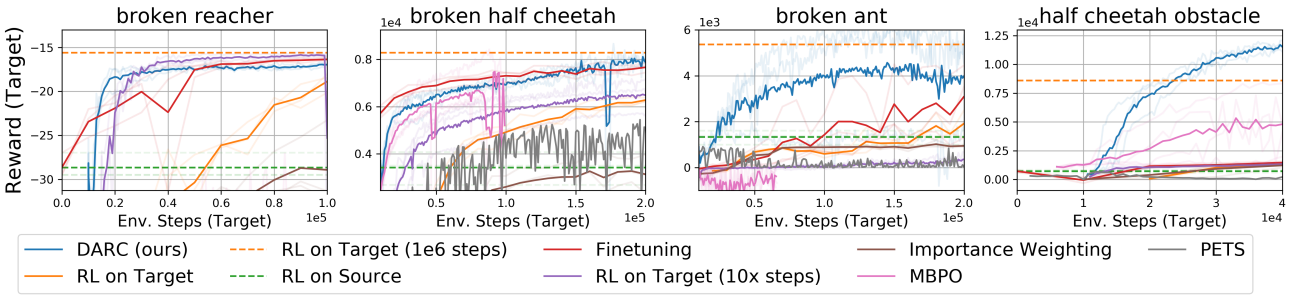

Figure 4. **DARC compensates for crippled robots and obstacles**: We apply DARC to four continuous control tasks: three tasks (broken reacher, half cheetah, and ant) which are crippled in the target domain but not the source domain, and one task (half cheetah obstacle) where the source domain omits the obstacle from the target domain. Note that naïvely ignoring the shift in dynamics (green dashed line) performs quite poorly, while directly learning on the crippled robot requires an order of magnitude more experience than our method.

shift in dynamics is external to the robot, by modifying the cheetah task to reward the agent for running both forward and backwards. It is easier to learn to run backwards, but the target domain contains an obstacle that prevents the agent from running backwards.

We compare our method to seven baselines. **RL on Source** and **RL on Target** directly perform RL on the source and target domains, respectively. The **Finetuning** baseline takes the result of running RL on the source domain, and further finetunes the agent on the target domain. The **Importance Weighting** baseline performs RL on importance-weighted samples from the source domain; the importance weights are $\exp(\Delta r)$. To account for the fact that our method performs more gradient updates per environment step in the source domain, we trained a version of the RL on source baseline likewise does 10 gradient updates per source domain step. Finally, we compared against two model-based RL methods: MBPO (Janner et al., 2019) and PETS (Chua et al., 2018).

We show the results of this experiment in Fig. 4, plotting the reward on the *target* domain as a function of the number of transitions in the *target* domain. On all tasks, the RL on source baseline (shown as a dashed line because it observes no target transitions) performs considerably worse than the optimal policy from RL on the target domain, suggesting that good policies for the source domain are suboptimal for the target domain. Nonetheless, on three of the four tasks our method matches (or even surpasses) the asymptotic performance of doing RL on the target domain, despite never doing RL on experience from the target domain, and despite observing 5 - 10x less experience from the target domain. On the broken reacher and broken half cheetah tasks, we observe that finetuning on the target domain performs on par with our method. On the simpler broken reacher task, just doing RL on the target domain with a large number of gradient steps works quite well (we did not tune this parameter for our method). However, as we scale to the more complex broken ant and half cheetah obstacle tasks, we observe that all baselines perform poorly.

To gain more intuition for our method, we recorded the

Figure 5. Without the reward correction, the agent takes transitions where the source domain and target domains are dissimilar; after adding the reward correction, the agent's transitions in the source domain are increasingly plausible under the target domain.

reward correction Δr throughout training on the broken reacher environment. In this experiment, we ran RL on the source domain for 100k steps before switching to our method. Said another way, we ignored Δr for the first 100k steps of training. As shown in Fig. 5, Δr steadily decreases during these first 100k steps, suggesting that the agent is learning a strategy that takes transitions where the source domain and target domain have different dynamics: the agent is making use of its broken joint. After 100k steps, when we maximize the combination of task reward and reward correction Δr, we observe that Δr increases, so the agent's transitions in the source domain are increasingly consistent with target domain dynamics. After around 1e6 training steps Δr is zero: the agent has learned a strategy that uses transitions that are indistinguishable between the source and target domains.

6. Discussion

In this paper, we proposed a simple, practical, and intuitive approach for domain adaptation to changing dynamics in RL. We formally motivate this method from a novel variational perspective on domain adaptation in RL, which suggests that we can compensate for differences in dynamics via the reward function. Experiments on a range of tasks show that our method can leverage the source domain to learn policies that work well in the target domain, despite observing only a handful of transitions from the target domain.

References

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., and Riedmiller, M. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Attias, H. Planning by probabilistic inference. In *AISTATS*. Citeseer, 2003.

Baktashmotlagh, M., Harandi, M. T., Lovell, B. C., and Salzmann, M. Domain adaptation on the statistical manifold. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2481–2488, 2014.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pp. 908–918, 2017.

Bickel, S., Brückner, M., and Scheffer, T. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pp. 81–88, 2007.

Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4243–4250. IEEE, 2018.

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

Chebotar, Y., Handa, A., Makoviychuk, V., Macklin, M., Issac, J., Ratliff, N., and Fox, D. Closing the sim-to-real loop: Adapting simulation randomization with real world experience. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8973–8979. IEEE, 2019.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4754–4765, 2018.

Clavera, I., Nagabandi, A., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 3, 2018.

Cortes, C. and Mohri, M. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014.

Cutler, M., Walsh, T. J., and How, J. P. Reinforcement learning with multi-fidelity simulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3888–3895. IEEE, 2014.

Dann, C., Neumann, G., Peters, J., et al. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15:809–883, 2014.

Dayan, P. and Hinton, G. E. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

Deisenroth, M. and Rasmussen, C. E. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Dudík, M., Langford, J., and Li, L. Doubly robust policy evaluation and learning. *arXiv preprint arXiv:1103.4601*, 2011.

Eysenbach, B., Gu, S., Ibarz, J., and Levine, S. Leave no trace: Learning to reset for safe and autonomous reinforcement learning. *arXiv preprint arXiv:1711.06782*, 2017.

Farchy, A., Barrett, S., MacAlpine, P., and Stone, P. Humanoid robots learning to walk faster: From the real world to simulation and back. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 39–46, 2013.

Feldbaum, A. Dual control theory. i. *Avtomatika i Telemekhanika*, 21(9):1240–1249, 1960.

Fernando, B., Habrard, A., Sebban, M., and Tuytelaars, T. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pp. 2960–2967, 2013.

Finn, C. and Levine, S. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

Gamrian, S. and Goldberg, Y. Transfer learning for related reinforcement learning tasks via image-to-image translation. *arXiv preprint arXiv:1806.07377*, 2018.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Harris, C., Vanhoucke, V., et al. Tf-agents: A library for reinforcement learning in tensorflow, 2018.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1480–1490. JMLR. org, 2017.

Hoffman, J., Wang, D., Yu, F., and Darrell, T. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016.

Huszár, F. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pp. 12498–12509, 2019.

Jaques, N., Gu, S., Bahdanau, D., Hernández-Lobato, J. M., Turner, R. E., and Eck, D. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1645–1654. JMLR. org, 2017.

Jaques, N., Ghandeharioun, A., Shen, J. H., Ferguson, C., Lapedriza, A., Jones, N., Gu, S., and Picard, R. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Kanamori, T., Hido, S., and Sugiyama, M. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009.

Kappen, H. J. Path integrals and symmetry breaking for optimal control theory. *Journal of statistical mechanics: theory and experiment*, 2005(11):P11011, 2005.

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

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Kouw, W. M. and Loog, M. A review of domain adaptation without target labels. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

Levine, S. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Levine, S. and Koltun, V. Variational policy search via trajectory optimization. In *Advances in neural information processing systems*, pp. 207–215, 2013.

Levine, S., Pastor, P., Krizhevsky, A., Ibarz, J., and Quillen, D. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37(4-5):421–436, 2018.

Lipton, Z. C., Wang, Y.-X., and Smola, A. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018.

Ljung, L. System identification. *Wiley encyclopedia of electrical and electronics engineering*, pp. 1–19, 1999.

Madrigal, A. C. Waymo built a secret world for self-driving cars, Dec 2018. URL https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facility/537648/.

Matsakis, L. Amazon has a history of bear repellent accidents, Dec 2018. URL https://www.wired.com/story/amazon-first-bear-repellent-accident/.

Mihatsch, O. and Neuneier, R. Risk-sensitive reinforcement learning. *Machine learning*, 49(2-3):267–290, 2002.

Mohamed, S. and Lakshminarayanan, B. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Munos, R., Stepleton, T., Harutyunyan, A., and Bellemare, M. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1054–1062, 2016.

Peng, X. B., Andrychowicz, M., Zaremba, W., and Abbeel, P. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 1–8. IEEE, 2018.

Polydoros, A. S. and Nalpantidis, L. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

Raibert, M. The best robots on four legs with marc raibert (boston dynamics), Apr 2019. URL https://youtu.be/tAhxi8WldCU?t=485.

Rajeswaran, A., Ghotra, S., Ravindran, B., and Levine, S. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

Rawlik, K., Toussaint, M., and Vijayakumar, S. On stochastic optimal control and reinforcement learning by approximate inference. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

Ross, S. and Bagnell, J. A. Agnostic system identification for model-based reinforcement learning. *arXiv preprint arXiv:1203.1007*, 2012.

Sadeghi, F. and Levine, S. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

Sastry, S. S. and Isidori, A. Adaptive control of linearizable systems. *IEEE Transactions on Automatic Control*, 34(11):1123–1131, 1989.

Schroecker, Y. and Isbell, C. Universal value density estimation for imitation learning and goal-conditioned reinforcement learning. *arXiv preprint arXiv:2002.06473*, 2020.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sønderby, C. K., Caballero, J., Theis, L., Shi, W., and Huszár, F. Amortised map inference for image super-resolution. *arXiv preprint arXiv:1610.04490*, 2016.

Song, H. F., Abdolmaleki, A., Springenberg, J. T., Clark, A., Soyer, H., Rae, J. W., Noury, S., Ahuja, A., Liu, S., Tirumala, D., et al. V-mpo: On-policy maximum a posteriori policy optimization for discrete and continuous control. *arXiv preprint arXiv:1909.12238*, 2019.

Sugiyama, M. and Müller, K.-R. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23(4/2005):249–279, 2005a.

Sugiyama, M. and Müller, K.-R. Model selection under covariate shift. In *International Conference on Artificial Neural Networks*, pp. 235–240. Springer, 2005b.

Sugiyama, M., Krauledat, M., and MÃžller, K.-R. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P. V., and Kawanabe, M. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pp. 1433–1440, 2008.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., and Okanohara, D. Conditional density estimation via least-squares density ratio estimation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 781–788, 2010.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Tamar, A., Xu, H., and Mannor, S. Scaling up robust mdps by reinforcement learning. *arXiv preprint arXiv:1306.6189*, 2013.

Tan, J., Xie, Z., Boots, B., and Liu, C. K. Simulation-based design of dynamic controllers for humanoid balancing. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2729–2736. IEEE, 2016.

Tanaskovic, M., Fagiano, L., Smith, R., Goulart, P., and Morari, M. Adaptive model predictive control for constrained linear systems. In *2013 European Control Conference (ECC)*, pp. 382–387. IEEE, 2013.

Thananjeyan, B., Balakrishna, A., Rosolia, U., Li, F., McAllister, R., Gonzalez, J. E., Levine, S., Borrelli, F., and Goldberg, K. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *IEEE Robotics and Automation Letters*, 5(2):3612–3619, 2020.

Theodorou, E., Buchli, J., and Schaal, S. A generalized path integral control approach to reinforcement learning. *journal of machine learning research*, 11(Nov):3137–3181, 2010.

Tiao, L. C., Bonilla, E. V., and Ramos, F. Cycle-consistent adversarial learning as approximate bayesian inference. *arXiv preprint arXiv:1806.01771*, 2018.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Todorov, E. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pp. 1369–1376, 2007.

Toussaint, M. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056, 2009.

Uehara, M., Sato, I., Suzuki, M., Nakayama, K., and Matsuo, Y. Generative adversarial nets from a density ratio estimation perspective. *arXiv preprint arXiv:1610.02920*, 2016.

Vemula, A., Oza, Y., Bagnell, J. A., and Likhachev, M. Planning and execution using inaccurate models with provable guarantees. *arXiv preprint arXiv:2003.04394*, 2020.

Wang, T., Bao, X., Clavera, I., Hoang, J., Wen, Y., Langlois, E., Zhang, S., Zhang, G., Abbeel, P., and Ba, J. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

Werbos, P. J. Neural networks for control and system identification. In *Proceedings of the 28th IEEE Conference on Decision and Control,*, pp. 260–265. IEEE, 1989.

White, M. Unifying task specification in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3742–3750. JMLR. org, 2017.

Williams, G., Aldrich, A., and Theodorou, E. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.

Wittenmark, B. Adaptive dual control methods: An overview. In *Adaptive Systems in Control and Signal Processing 1995*, pp. 67–72. Elsevier, 1995.

Wulfmeier, M., Bewley, A., and Posner, I. Addressing appearance change in outdoor robotics with adversarial domain adaptation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1551–1558. IEEE, 2017.

Yu, W., Tan, J., Liu, C. K., and Turk, G. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

Yu, Y. and Szepesvári, C. Analysis of kernel mean matching under covariate shift. *arXiv preprint arXiv:1206.4650*, 2012.

Zadrozny, B. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 114, 2004.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017a.

Zhu, S., Kimmel, A., Bekris, K. E., and Boularias, A. Fast model identification via physics engines for data-efficient policy search. *arXiv preprint arXiv:1710.08893*, 2017b.

Ziebart, B. D. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.

A. Related Work

While our work will focus on domain adaptation applied to RL, we start by reviewing more general ideas in domain adaptation, and defer to Kouw & Loog (2019) for a recent review of the field. Two common approaches to domain adaptation are importance weighting and domain-agnostic features. *Importance-weighting methods* (e.g., (Zadrozny, 2004; Cortes & Mohri, 2014; Lipton et al., 2018)) estimate the likelihood ratio of examples under the target domain versus the source domain, and use this ratio to re-weight examples sampled from the source domain. To estimate the likelihood ratio, some methods directly estimate two density models and then take the difference (e.g., (Sugiyama & Müller, 2005a; Yu & Szepesvári, 2012; Baktashmotlagh et al., 2014)), other methods directly estimate the ratio (Sugiyama & Müller, 2005b; Sugiyama et al., 2007; 2008; Mohamed & Lakshminarayanan, 2016; Kanamori et al., 2009; Uehara et al., 2016). A number of these direct estimation methods operate by learning a classifier to distinguish examples from the source domain versus examples from the target domain (Bickel et al., 2007; Sønderby et al., 2016; Mohamed & Lakshminarayanan, 2016; Uehara et al., 2016). We refer to Huszár (2017) for a recent review of direct estimation methods. Methods based on *domain-agnostic features* aim to map examples from the source and target domain into a common feature space (Fernando et al., 2013; Hoffman et al., 2016; Wulfmeier et al., 2017). Our method is similar to classifier-based density-ratio estimation, with two important distinctions. First, we will need to estimate the density ratio of conditional distributions (transition probabilities), which is different from modeling conditional distributions as a density ratio (Sugiyama et al., 2010). To do this, we will learn not one but two classifiers. Second, we will use the logarithm of the density ratio to modify the reward function instead of weighting samples by the density ratio, which is often numerically unstable (see, e.g., Schulman et al. (2017, §3)).

Prior methods for applying domain adaptation to RL include approaches based on system identification, domain randomization, and observation adaptation. Perhaps the most established approach, system identification (Ljung, 1999), uses observed data to tune the parameters of a simulator (Feldbaum, 1960; Werbos, 1989; Wittenmark, 1995; Ross & Bagnell, 2012; Tan et al., 2016; Zhu et al., 2017b; Farchy et al., 2013) More recent work has successfully used this strategy to bridge the sim2real gap (Chebotar et al., 2019; Rajeswaran et al., 2016). Closely related is work on online system identification and meta-learning, which directly uses the inferred system parameters to update the policy (Yu et al., 2017; Clavera et al., 2018; Tanaskovic et al., 2013; Sastry & Isidori, 1989). However, these approaches typically require either a model of the environment or a manually-specified distribution over potential test-time dynamics, requirements that our method will lift. Another approach, *domain randomization*, randomly samples the parameters of the source domain and then finds the best policy for this randomized environment (Sadeghi & Levine, 2016; Tobin et al., 2017; Peng et al., 2018; Cutler et al., 2014). While often effective, this method is sensitive to the choice of which parameters are randomized, and the distributions from which these simulator parameters are sampled. A third approach, *observation adaptation*, modifies the observations of the source domain to appear similar to those in the target domain. While this approach has been successfully applied to video games (Gamrian & Goldberg, 2018) and robot manipulation (Bousmalis et al., 2018), it ignores the fact that the source and target domains may have differing dynamics.

The theoretical derivation of our method is heavily inspired by prior work which formulates control as a problem of probabilistic inference (Dayan & Hinton, 1997; Attias, 2003; Kappen, 2005; Todorov, 2007; Toussaint, 2009; Ziebart, 2010; Theodorou et al., 2010; Rawlik et al., 2013; Levine & Koltun, 2013; Levine, 2018; Abdolmaleki et al., 2018). These methods aim to make an agent's experience in the target domain look like the expert's experience in the target domain, whereas our method aims to make an agent's experience in the source domain look like the expert's experience in the target domain. We emphasize that the domain shift we consider is caused by domains having different dynamics, not by actions being sampled from different policies. Algorithms for model-based RL (e.g., (Polydoros & Nalpantidis, 2017; Sutton, 1991; Janner et al., 2019; Deisenroth & Rasmussen, 2011; Wang et al., 2019; Williams et al., 2015; Hafner et al., 2018; Chua et al., 2018; Finn & Levine, 2017) and off-policy RL (e.g., (Munos et al., 2016; Fujimoto et al., 2018; Dann et al., 2014; Dudík et al., 2011) similarly aim to improve the sample efficiency of RL, but do use the source domain to accelerate learning. Our method is applicable to any maximum entropy RL algorithm, including on-policy (Song et al., 2019), off-policy (Abdolmaleki et al., 2018; Haarnoja et al., 2018), and model-based (Janner et al., 2019; Williams et al., 2015) algorithms. We will use the soft actor critic algorithm (Haarnoja et al., 2018) in our experiments. Recently, Vemula et al. (2020) proposed a method for planning with an inaccurate model by assigning a high, fixed, cost to transitions where the model was inaccurate. Our method similarly accounts for discrepancies in dynamics via rewards, but does so with a learned classifier, allowing our method to be applied in stochastic environments with continuous states and actions.

B. The Special Case of an Observation Model

To highlight the relationship between domain adaptation of dynamics versus observations, we now consider a special case. In this subsection, we will assume that the state $s_t \triangleq (z_t, o_t)$ is a combination of the system latent state z_t (e.g., the poses of all objects in a scene) and an observation o_t (e.g., a camera observation). We will define $q(o_t \mid z_t)$ and $p(o_t \mid z_t)$ as the *observation models* for the source and target domains. In this special case, we can decompose the KL objective into three terms:

$$D_{\mathrm{KL}}(q \parallel p) = -\mathbb{E}_q \Bigg[\sum_t \underbrace{r(s_t, a_t) + \mathcal{H}_\pi[a_t \mid s_t]}_{\text{MaxEnt RL objective}} + \underbrace{\log p_{\text{target}}(o_t \mid z_t) - \log p_{\text{source}}(o_t \mid z_t)}_{\text{Observation Adaptation}} \underbrace{\log p_{\text{target}}(z_{t+1} \mid z_t, a_t) - \log p_{\text{source}}(z_{t+1} \mid z_t, a_t)}_{\text{Dynamics Adaptation}} \Bigg].$$

Prior methods that perform observation adaptation (Bousmalis et al., 2018; Gamrian & Goldberg, 2018) effectively minimize the observation adaptation term,[1] but ignore the effect of dynamics. In contrast, the Δr reward correction in our method provides one method to address both dynamics and observations. These approaches could be combined; we leave this as future work.

C. Additional Interpretations of the Reward Correction

This section presents four additional interpretations of the reward correction, Δr.

C.1. Coding Theory

The reward correction Δr can also be understood from the perspective of coding theory. Suppose that we use a data-efficient replay buffer that exploits that fact that the next state s_{t+1} is highly redundant with the current state and action, s_t, a_t. If we assume that the replay buffer compression has been optimized to store transitions from the target environment, (negative) Δr is the number of additional bits (per transition) needed for our source replay buffer, as compared with our target replay buffer. Thus, an agent which maximizes Δr will seek those transitions that can be encoded most efficiently, minimizing the size of the source replay buffer.

C.2. Mutual Information

We can gain more intuition in the modified reward by writing the expected value of Δr in terms of mutual information:

$$\mathbb{E}[\Delta r(s_t, a_t, s_{t+1})] = I(s_{t+1}; \text{target} \mid s_t, a_t) - I(s_{t+1}; \text{source} \mid s_t, a_t).$$

The mutual information $I(s_{t+1}; \text{target} \mid s_t, a_t)$ reflects how much better you can predict the next state if you know that you are interacting with the target domain, instead of the source domain. Our approach does exactly this, rewarding the agent for taking transitions that provide information about the target domain while penalizing transitions that hint to the agent that it is interacting with a source domain rather than the target domain.

[1] Tiao et al. (2018) show that observation adaptation using CycleGan (Zhu et al., 2017a) minimizes a Jensen-Shannon divergence. Assuming sufficiently expressive models, the Jensen-Shannon divergence and the reverse KL divergence above have the same optimum.

C.3. Lower bound on the risk-sensitive reward objective.

While we derived DARC by minimizing a reverse KL divergence, we can also show that DARC maximizes a lower bound on a risk-sensitive reward objective (Mihatsch & Neuneier, 2002):

$$\begin{aligned}
&\log \mathbb{E}_{\substack{s' \sim p_{\text{target}}(s'|s,a), \\ a \sim \pi(a|s)}} \left[\exp\left(\sum_t r(s_t, a_t) \right) \right] \\
&= \log \mathbb{E}_{\substack{s' \sim p_{\text{source}}(s'|s,a), \\ a \sim \pi(a|s)}} \left[\left(\prod_t \frac{p_{\text{target}}(s_{t+1} \mid s_t, a_t)}{p_{\text{source}}(s_{t+1} \mid s_t, a_t)} \right) \exp\left(\sum_t r(s_t, a_t) \right) \right] \\
&= \log \mathbb{E}_{\substack{s' \sim p_{\text{source}}(s'|s,a), \\ a \sim \pi(a|s)}} \left[\exp\left(\sum_t r(s_t, a_t) + \underbrace{\log p_{\text{target}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{source}}(s_{t+1} \mid s_t, a_t)}_{\Delta r(s_t, a_t, s_{t+1})} \right) \right] && (1) \\
&\geq \mathbb{E}_{\substack{s' \sim p_{\text{source}}(s'|s,a), \\ a \sim \pi(a|s)}} \left[\sum_t r(s_t, a_t) + \Delta r(s_t, a_t, s_{t+1}) \right]. && (2)
\end{aligned}$$

The inequality on the last line is an application of Jensen's inequality. One interesting question is when it would be preferable to maximize Eq. 1 rather than Eq. 2. While Eq. 2 provides a loser bound on the risk sensitive objective, empirically it may avoid the risk-seeking behavior that can be induced by risk-sensitive objectives. We leave the investigation of this trade-off as future work.

C.4. A Constraint on Dynamics Discrepancy

Our method regularizes the policy to visit states where the transition dynamics are similar between the source domain and target domain:

$$\max_{\pi} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t r(s_t, a_t) + \underbrace{\log p_{\text{target}}(s_{t+1} \mid s_t, a_t) - \log p_{\text{source}}(s_{t+1} \mid s_t, a_t)}_{-D_{\text{KL}}(p_{\text{source}} \,\|\, p_{\text{target}})} + \mathcal{H}_\pi[a_t \mid s_t] \right].$$

This objective can equivalently be expressed as applying MaxEnt RL to only those policies which avoid exploiting the dynamics discrepancy. More precisely, the KKT conditions guarantee that there exists a positive constant $\epsilon > 0$ such that our objective is equivalent to the following constrained objective:

$$\max_{\pi \in \Pi_{\text{DARC}}} \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t r(s_t, a_t) + \mathcal{H}_\pi[a_t \mid s_t] \right],$$

where Π_{DARC} denotes the set of policies that do not exploit the dynamics discrepancy:

$$\Pi_{\text{DARC}} \triangleq \left\{ \pi \middle| \mathbb{E}_{\substack{a \sim \pi(a|s) \\ s' \sim p(s'|s,a)}} \left[\sum_t D_{\text{KL}}(p_{\text{source}}(s_{t+1} \mid s_t, a_t) \,\|\, p_{\text{target}}(s_{t+1} \mid s_t, a_t)) \right] \leq \epsilon \right\}.$$

One potential benefit of considering our method as the unconstrained objective is that it provides a principled method for increasing or decreasing the weight on the Δr term, depending on how much the policy is currently exploiting the dynamics discrepancy. We leave this investigation as future work.

D. Additional Experiments

D.1. Visualizing the reward modification in stochastic domains.

In this experiment, we use an "archery" task to visualize how the modified reward accounts for differences in dynamics. The task, shown in Fig. 6, requires choosing an angle at which to shoot an arrow. The practice range (i.e., the source domain) is outdoors, with wind that usually blows from left to right. The competition range (i.e., the target domain) is indoors with no wind. The reward is the negative distance to the target. We plot the reward as a function of the angle in both domains in Fig. 6. The optimal strategy for the outdoor range is to compensate for the wind by shooting slightly to the left ($\theta = -0.8$),

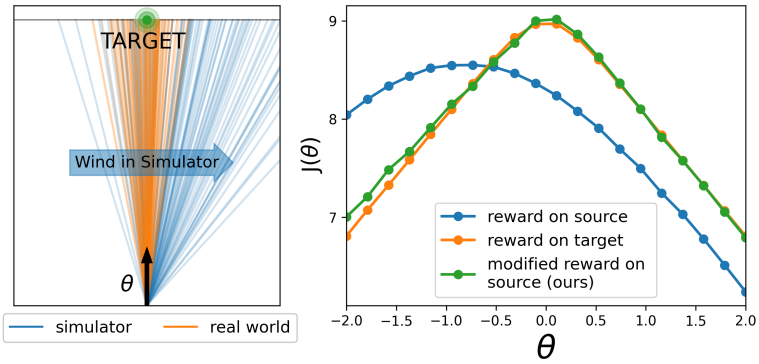

Figure 6. **Visualizing the modified reward**

while the optimal strategy for the indoor range is to shoot straight ahead ($\theta = 0$). We estimate the modified reward function with DARC, and plot the modified reward in the windy outdoor range and indoor range. We aggregate across episodes using $J(\theta) = \log \mathbb{E}_{p(s'|\theta)}[\exp(r(s'))]$; see Appendix E.4 for details. We observe that maximizing the modified reward in the windy range does not yield high reward in the windy range, but does yield a policy that performs well in the indoor range.

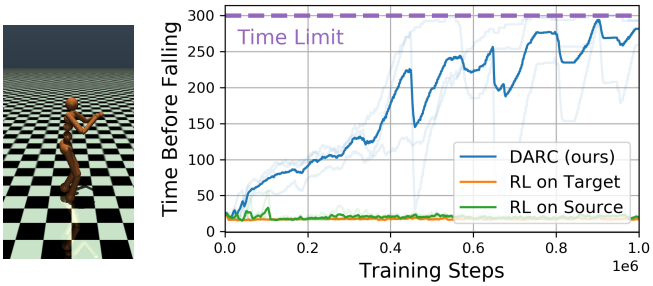

Figure 7. Our method accounts for domain shift in the termination condition, causing the agent to avoid transitions that cause termination in the target domain.

D.2. Safety emerges from domain adaptation to the termination condition.

The termination condition is part of the dynamics (White, 2017), and our next experiment studies how our method copes with domain shift in the termination condition. We use the humanoid shown in Fig. 7 for this experiment and set the task reward to 0. In the source domain episodes have a fixed length of 300 steps; in the target domain the episode terminates when the robot falls. The scenario mimics the real-world setting where robots have freedom to practice in a safe, cushioned, practice facility, but are preemptively stopped when they try to take unsafe actions in the real world. Our aim is for the agent to learn to avoid unsafe transitions in the source domain that would result in episode termination in the target domain (see Broader Impacts for more discussion). As shown in Fig. 7, our method learns to remain standing for nearly the entire episode. As expected, baselines that maximize the zero reward on the source and target domains fall immediately. While DARC was not designed as a method for safe RL (Tamar et al., 2013; Achiam et al., 2017; Eysenbach et al., 2017; Berkenkamp et al., 2017), this experiment suggests that safety may emerge automatically from DARC, without any manual reward function design.

D.3. Ablation Experiments

Our final experiment examines the importance of using two classifiers to estimate Δr. We compared our method to an ablation that does not learn the SA classifier, effectively ignoring the blue terms in Δr. As shown in Fig. 8, this ablation performs considerably worse than our method. Intuitively, this makes sense: we might predict that a transition came from the source domain not because the next state had higher likelihood under the source dynamics, but rather because the state or action was visited more frequently in the source domain. The second classifier used in our method corrects for this distribution shift.

Finally, we examine the importance of input noise regularization in classifiers. As we observe only a handful of transitions from the target domain, we hypothesized that regularization would be important to prevent overfitting. We test this hypothesis in Fig. 9 by training our method on the broken reacher environment with varying amounts of input noise. With no noise

or little noise our method performs poorly (likely due to overfitting); too much noise also performs poorly (likely due to underfitting). We used a value of 1 in all our experiments, and did not tune this value. See Appendix D.3 for more plots of both ablation experiments.

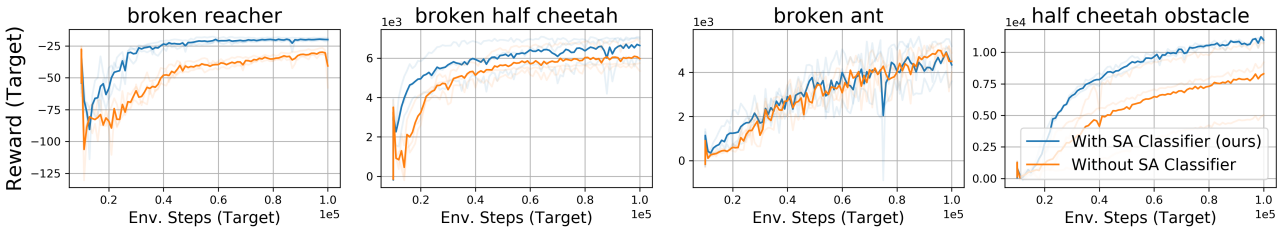

Figure 8. **Importance of using two classifiers**: DARC performs worse when only one classifier is used.

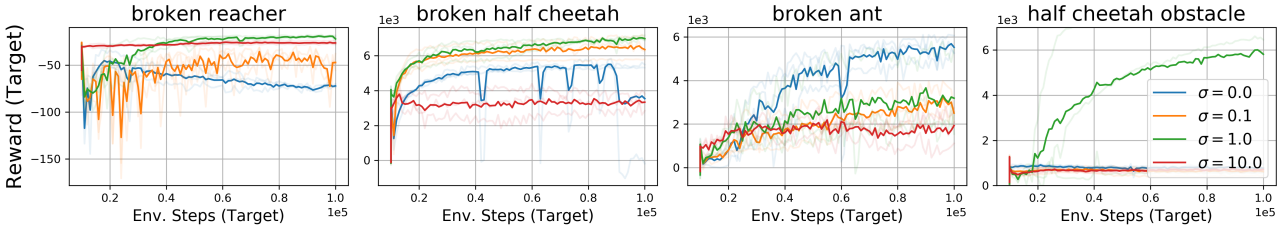

Figure 9. **Importance of regularizing the classifiers**: Using input noise to regularize the classifiers boosts performance.

E. Experiment Details and Hyperparameters

Our implementation of DARC is built on top of the implementation of SAC from Guadarrama et al. (2018). Unless otherwise specified, all hyperparameters are taken from Guadarrama et al. (2018). All neural networks (actor, critics, and classifiers) have two hidden layers with 256-units each and ReLU activations. Since we ultimately will use the *difference* in the predictions of the two classifiers, we use a residual parametrization for the SAS classifier $q(\text{target} \mid s_t, a_t, s_{t+1})$. Using $f_{\text{SAS}}(s_t, a_t, s_{t+1}), f_{\text{SA}}(s_t, a_t) \in \mathbb{R}^2$ to denote the outputs of the two classifier networks, we compute the classifier predictions as follows:

$$q_{\theta_{\text{SA}}}(\cdot \mid s_t, a_t) = \textsc{SoftMax}(f_{\text{SA}}(s_t, a_t))$$
$$q_{\theta_{\text{SAS}}}(\cdot \mid s_t, a_t, s_{t+1}) = \textsc{SoftMax}(f_{\text{SAS}}(s_t, a_t, s_{t+1}) + f_{\text{SA}}(s_t, a_t))$$

For the SAS classifier we propagate gradients back through both networks parameters, θ_{SAS} and θ_{SA}. Both classifiers use Gaussian input noise with $\sigma = 1$. Optimization of all networks is done with Adam (Kingma & Ba, 2014) with a learning rate of 3e-4 and batch size of 128. Most experiments with DARC collected 1 step in the target domain every 10 steps in the source domain (i.e., $r = 10$). The one exception is the half cheetah obstacle domain, where we tried increasing r beyond 10 to 30, 100, 300, and 1000. We found a large benefit from increasing r to 30 and 100, but did not run the other experiments long enough to draw any conclusions. Fig. 4 uses $r = 30$ for half cheetah obstacle. We did not tune this parameter, and expect that tuning it would result in significant improvements in sample efficiency.

We found that DARC was slightly more stable if we warm-started the method by applying RL on the source task *without* Δr for the first t_{warmup} iterations. We used $t_{\text{warmup}} = 1e5$ for all tasks except the broken reacher, where we used $t_{\text{warmup}} = 2e5$. This discrepancy was caused by a typo in an experiment, and subsequent experiments found that DARC is relatively robust to different values of t_{warmup}; we did not tune this parameter.

E.1. Baselines

The **RL on Source** and **RL on Target** baselines are implemented identically to our method, with the exception that Δr is not added to the reward function. The **RL on Target (10x)** is identical to RL on Target, with the exception that we take 10 gradient steps per environment interaction (instead of 1). The **Importance Weighting** baseline estimates the importance weights as $p_{\text{target}}(s_{t+1} \mid s_t, a_t)/p_{\text{source}}(s_{t+1} \mid s_t, a_t) \approx \exp(\Delta r)$. The importance weight is used to weight transitions in the SAC actor and critic losses.

PETS (Chua et al., 2018) : The **PETS** baseline is implemented using the default configurations used by (Chua et al., 2018) for the environments evaluated. That is, the `broken-half-cheetah` environment uses the hyperparameters as used by the `half-cheetah` environment in (Chua et al., 2018).

MBPO (Janner et al., 2019)

E.2. Environments

Broken Reacher This environment uses the 7DOF robot arm from the Pusher environment in OpenAI Gym. The observation space is the position and velocities of all joints and the goal. The reward function is

$$r(s, a) = -\frac{1}{2}\|s_{\text{end effector}} - s_{\text{goal}}\|_2 - \frac{1}{10}\|a\|_2^2,$$

and episodes are 100 steps long. In the target domain the 2nd joint (0-indexed) is broken: zero torque is applied to this joint, regardless of the commanded torque.

Broken Half Cheetah This environment is based on the HalfCheetah environment in OpenAI Gym. Episodes are 1000 steps long. In the target domain the 0th joint (0-indexed) is broken: zero torque is applied to this joint, regardless of the commanded torque.

Broken Ant This environment is based on the Ant environment in OpenAI Gym. We use the standard termination condition and cap the maximum episode length at 1000 steps. In the target domain the 3rd joint (0-indexed) is broken: zero torque is applied to this joint, regardless of the commanded torque.

In all the broken joint environments, we choose which joint to break to computing which joint caused the "RL on Source" baseline to perform worst on the target domain, as compared with the "RL on Target" baseline.

Half Cheetah Obstacle This environment is based on the HalfCheetah environment in OpenAI Gym. Episodes are 1000 steps long. We modified the standard reward function to use the absolute value in place of the velocity, resulting the following reward function:

$$r(s, a) = s_{\text{x vel}} \cdot \Delta t - \|a\|_2^2,$$

where $s_{\text{x vel}}$ is the velocity of the agent along the forward-aft axis and $\Delta t = 0.01$ is the time step of the simulator. In the target domain, we added a wall at $x = -3m$, roughly 3 meters behind the agent.

Humanoid Used for the experiment in Fig. 7, we used a modified version of Humanoid from OpenAI Gym. The source domain modified this environment to ignore the default termination condition and instead terminate after exactly 300 time steps. The target domain uses the unmodified environment, which terminates when the agent falls.

E.3. Figures

Unless otherwise noted, all experiments were run with three random seeds. Figures showing learning curves (Figures 4, 5, 8, and 9) plot the *mean* over the three random seeds, and also plot the results for each individual random seed with semi-transparent lines.

E.4. Archery Experiment

We used a simple physics model for the archery experiment. The target was located 70m North of the agent, and wind was applied along the East-West axis. The system dynamics:

$$s_{t+1} = 70\sin(\theta) + f/\cos(\theta)^2 \qquad \begin{cases} f \sim \mathcal{N}(\mu = 1, \sigma = 1) & \text{in the target domain} \\ f \sim \mathcal{N}(\mu = 0, \sigma = 0.3) & \text{in the source domain} \end{cases}$$

We trained the classifier by sampling $\theta \sim \mathcal{U}[-2, 2]$ (measured in degrees) for 10k episodes in the source domain and 10k episodes in the target domain. The classifier was a neural network with 1 hidden layer with 32 hidden units and ReLU activation. We optimized the classifier using the Adam optimizer with a learning rate of 3e-3 and a batch size of 1024. We trained until the validation loss increased for 3 consecutive epochs, which took 16 epochs in our experiment. We generated Fig. 6 by sampling 10k episodes for each value of θ and aggregating the rewards using $J(\theta) = \log \mathbb{E}_{p(s'|\theta)}[\exp(r(s'))]$. We found that aggregating rewards by taking the mean did not yield meaningful results, perhaps because the mean corresponds to a (possibly loose) lower bound on J (see Appendix C.3).