Undo Maps: A Tool for Adapting Policies to Perceptual Distortions

Anonymous Authors[1]

Abstract

People adapt to changes in their visual field all the time, like when their vision is occluded while driving. Agents trained with RL struggle to do the same. Here, we address how to transfer knowledge acquired in one domain to another when the domains differ in their state representation. For example, a policy may have been trained in an environment where states were represented as colored images, but we would now like to deploy this agent in a domain where images appear black-and-white. We propose TAIL–task-agnostic imitation learning–a framework which learns to undo these kinds of changes between domains in order to achieve transfer. This enables an agent, regardless of the task it was trained for, to adapt to perceptual distortions by first mapping the states in the new domain, such as gray-scale images, back to the original domain where they appear in color, and then by acting with the same policy. Our procedure depends on an optimal transport formulation between trajectories in the two domains, shows promise in simple experimental settings, and resembles algorithms from imitation learning.

1. Introduction

We perceive the world as up-right even though the image that hits the retina of the human eye is inverted. What if everything was indeed flipped upside-down? The same question, first posed by psychologist George Stratton (Stratton, 1897), has become the focus of many experiments trying to understand visual perception today. Subjects are traditionally made to wear goggles equipped with prisms in order to distort their view. These optical illusions can be as subtle as shifting the visual field by a translation to as extreme as inverting the visual field altogether (Stratton, 1896; Linden et al., 1999). In both cases, individuals adapt to the transformation in remarkably less time than it would require for them to learn basic visuomotor commands from scratch.

[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.

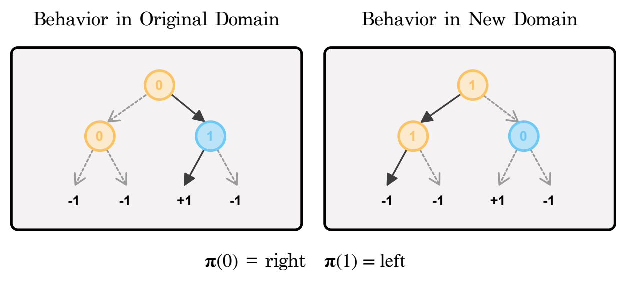

Figure 1. **Motivating example.** A high-reward seeking policy acts optimally in the domain where it was trained, but behaves suboptimally when the same states (i.e. yellow and blue nodes) are represented differently.

Although people adapt to their environment all the time, like when their vision is deliberately inverted, policies trained with RL struggle to do the same. Consider the problem of reaching a high-rewarding terminal state in a binary tree as shown in Figure 1. At every node, the state is represented by a 0 or a 1 as illustrated and the agent can either move to the left or to the right child. Starting at the root node, we train a policy with RL to reach the high rewarding leaf node in the tree. This policy is trained in a domain where the yellow nodes are represented as 0 and blue nodes are represented as 1, so the agent learns the rule $\pi(0) =$ right and $\pi(1) =$ left. We then purposely invert the state representation so yellow nodes now appear as 1 and blue nodes appear as 0. As expected, the same policy now behaves quite differently in the new domain.

In many scenarios, changing how the world is represented can debilitate a policy: an agent trained to solve a task from colored images will fail to perform in a setting where everything appears black and white, or an agent trained to solve a task from up-right observations will fail to perform in conditions where everything looks tilted. Just as how humans can adapt to such perceptual disturbances, a policy should be able to do the same without having to learn all over again. Transfer learning in RL is appealing for this reason, but has not received as much interest as supervised learning (Daumé, 2007; Ben-David et al., 2010; Weiss et al., 2016). Transfer in machine learning generally adapts a model to work well on a new dataset by finding correspondences between this dataset and the one used for training. This is different from RL where data typically contains variable-length sequences and a policy must be adapted to a new environment online rather than to a new dataset offline.

Contributions. We consider the setting where domains differ due to a drift in the representation of the state space, like the one outlined in Figure 1. Given a policy that has already been trained to solve a task, we adapt it to such perceptual distortions with TAIL. Our contributions are:

- **Optimal Transport Formulation.** We formulate the problem of transfer learning as matching the trajectory distributions of a policy acting in the original domain and in the new domain, respectively. We characterize the distributional distance with optimal transport (OT) and use the Dynamic Time-Warping distance as a ground metric in our OT formulation to handle trajectories with varying length and dynamics.
- **Undo Map as an Abstraction.** We explicitly learn the transformation between domains (i.e. the undo map), so we can adapt a policy that was unseen during training at no additional cost. We can also adapt multiple policies together, each living in the same domain, if they are all affected by the same perceptual distortion. The abstraction of the undo map provides greater capabilities for tackling transfer in RL compared to methods that directly adapt policies (Ho & Ermon, 2016; Xiao et al., 2019; Dadashi et al., 2020; Fickinger et al., 2021).
- **Empirical Evaluation.** We construct a grid world environment to evaluate the effectiveness of TAIL. We run our method on three different types of policies from the source domain to empirically understand when it is easier to transfer knowledge across domains.

2. Background

2.1. Imitation Learning

RL can solve a variety of control tasks provided that we can construct informative reward functions. There are many real-world cases where these reward functions are too difficult to specify, so imitation learning (IL) attempts to recover policies from data instead–specifically, demonstrations of expert behavior which are often available even when rewards are not. One approach includes Inverse-Reinforcement Learning (Abbeel & Ng, 2010) where first we estimate a reward function under which the observed demonstrations are optimal, and second we learn a policy with the estimated reward model.

2.2. Occupancy Measures

Consider an MDP $\mathcal{M} := (\mathcal{S}, \mathcal{A}, p, r, \gamma, d_0)$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $p(\cdot|s_t, a_t)$ is the transition distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and d_0 is the initial state distribution. A policy π induces a distribution ρ^π over trajectories $\tau := (s_0, a_0, s_1, a_1, \dots)$, where $\rho^\pi(\tau) = d_0(s_0) \prod_{t \geq 0} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$. The discounted state-action occupancy measure $\rho^\pi(s, a)$ is the state-action marginal of the trajectory distribution written as $\rho^\pi(s, a) := (1-\gamma)\pi(a|s)\sum_{t\geq 0} \mathbb{P}(S_t = s)$. Note that $\rho(s, a)$ uniquely defines a policy $\pi(a|s) := \frac{\rho(s,a)}{\sum_{a' \in \mathcal{A}} \rho(s,a')}$.

Generative Adversarial Imitation Learning (GAIL) (Ho & Ermon, 2016) bypasses the step of learning a reward function, as done in inverse-RL, by reformulating the problem as matching occupancy measures. From convex duality, IL can be described by the following optimization objective:

$$\min_{\pi \in \Pi} \max_{f \in \mathcal{D}} \mathbb{E}_{\rho^\pi}[\log f(s, a)] + \mathbb{E}_{\rho^\mu}[\log(1 - f(s, a))] \quad (1)$$

Here, μ is the expert that we intend to imitate with a policy π in the class Π and $\mathcal{D}$ is a set of classifiers in which $f : \mathcal{S} \times \mathcal{A} \to [0, 1]$ determines the likelihood of a state-action pair (s, a) being generated from the imitator as opposed to the expert. In fact, the inner maximization is exactly the Jensen-Shannon Divergence $D_{JS}(\rho^\mu, \rho^\pi)$ up to a constant.

2.3. Optimal Transport

The Wasserstein distance computes the distance between two distributions p_1 and p_2 with support in sets $\mathcal{X}_1$ and $\mathcal{X}_2$ when we have a cost function $c : \mathcal{X}_1 \times \mathcal{X}_2 \to \mathbb{R}$. For the purposes of this work, we focus on its dual formulation:

$$\begin{aligned} &\mathcal{W}_c(p_1, p_2) := \max_{h,g \in \mathcal{F}} \mathbb{E}_{p_1}[h(x)] + \mathbb{E}_{p_2}[g(x)] \\ &\text{s.t.} \quad h(x) + g(x') \leq c(x, x'),\ \forall (x, x') \in \mathcal{X}_1 \times \mathcal{X}_2, \end{aligned} \quad (2)$$

where $\mathcal{F}$ is a function class defined by the geometry that c induces on $\mathcal{X}_1 \times \mathcal{X}_2$. We will henceforth refer to functions in $\mathcal{F}$ as potentials. As an example, when $\mathcal{X}_1, \mathcal{X}_2 \subseteq \mathbb{R}^d$ and $c(x_1, x_2) = \|x_1 - x_2\|$, the set of functions $\mathcal{F}$ in the variational definition of the Wasserstein distance (2) corresponds to 1-Lipschitz functions. Solving this constrained optimization problem in practice can prove challenging, so we consider the regularized objective below,

$$\widetilde{\mathcal{W}}_c(p_1, p_2) := \max_{h,g \in \mathcal{F}} \mathbb{E}_{p_1}[h(x)] + \mathbb{E}_{p_2}[g(x)] - \alpha \mathbb{E}_{p_1 \times p_2}[f(x, x')]$$

where $\alpha > 0$ is a regularization parameter, $p_1 \times p_2$ is the product distribution of p_1 and p_2 (i.e. i.i.d. samples from each), and $f(x, x') := \max\{0, h(x) + g(x') - c(x, x')\}$.

3. Methodology

3.1. Optimal Transport Formulation

We assume access to trajectories $\{\tau_i\}_{i=1}^N \sim \rho^\mu$ where μ is a *behavior* policy that we intend to imitate. We will later see that μ need not be an expert as often the case in IL. Our goal is to find a new policy π that minimizes the Wasserstein distance $\widetilde{\mathcal{W}}_c(\rho^\mu, \rho^\pi)$ between the two trajectory distributions. This time, however, the two sets of trajectories live in different domains. We model the source and target domains as the MDPs $\mathcal{M}_S = (\mathcal{S}^S, \mathcal{A}^S, p^S, r^S, \gamma, d_0^S)$ and $\mathcal{M}_T = (\mathcal{S}^T, \mathcal{A}^T, p^T, r^T, \gamma, d_0^T)$, respectively. Here, the agent would have first interacted with $\mathcal{M}_S$ in order to learn how to solve the task. Afterwards, the agent would interact with a target domain $\mathcal{M}_T$ where its learning objectives will hopefully have been made simpler due to knowledge of $\mathcal{M}_S$. Our underlying assumption remains that there is structure shared between $\mathcal{M}_S$ and $\mathcal{M}_T$.

We denote a policy in the source domain with $\mu : \mathcal{S}^S \to \mathcal{P}(\mathcal{A}^S)$ and its trajectory distribution in $\mathcal{M}_S$ with ρ^μ. Policies in the target domain are represented as $\pi : \mathcal{S}^T \to \mathcal{P}(\mathcal{A}^T)$ with their trajectory distribution in $\mathcal{M}_T$ as ρ^π. We consider source and target MDP pairs $(\mathcal{M}_S, \mathcal{M}_T)$ which have the same action space $\mathcal{A}$. Our assumption of a shared action space is less restrictive than many prior works that typically assume both shared state and action spaces (Teh et al., 2017; Moskovitz et al., 2022a), or even shared transition kernels (Barreto et al., 2020; Moskovitz et al., 2022b).

[**Distributional Equivalence**] We say two MDPs $\mathcal{M}_S$ and $\mathcal{M}_T$ are *distributionally equivalent* if there exists an undo map $u_* : \mathcal{S}^T \to \mathcal{S}^S$ and a target policy $\pi(\cdot|s) := \mu(\cdot|u_*(s))$ such that $\widetilde{\mathcal{W}}_c(\rho^\mu, u_*(\rho^\pi)) = 0$ for all source policies μ, where $u_*(\rho^\pi)$ is the distribution of $u_*(\tau)$ when $\tau \sim \rho^\pi$.

Since ρ^π is a distribution over trajectories as opposed to the state-action occupancy measure, we define $u_*(\tau) := (u_*(s_0), a_0, u_*(s_1), a_1, \cdots)$ as applying the undo map state-wise. The policy π in our definition can be viewed as the composition $\mu \circ u_*$. At any given time t in the target domain, the agent would act by first undoing the distortion between domains according to $s'_t = u_*(s_t)$ and then by reusing the controller from the source domain as $a_t \sim \mu(\cdot|s'_t)$.

3.2. Learning the Undo Map

When $(\mathcal{M}_S, \mathcal{M}_T)$ are distributionally equivalent and we have access to μ, we can transfer knowledge across domains by searching for an undo map where $\pi := \mu \circ u$:

$$\min_{u \in \mathcal{U}} \widetilde{\mathcal{W}}_c(\rho^\mu, u(\rho^\pi)) \qquad (3)$$

This insight is the basis of our approach. When μ is an expert with respect to the reward function r^S and the reward remains unchanged in the target domain, we may augment our distributional objective with rewards from $\mathcal{M}_T$. In all other cases where μ may be any behavior policy, the objective in 3 solves the problem of finding a policy in $\mathcal{M}_T$ that behaves similarly to the one in $\mathcal{M}_S$.

The last ingredient needed for a practical algorithm is a suitable cost function $c : \Gamma^S \times \Gamma^S \to \mathbb{R}$ where Γ is the space of trajectories in one of the domains. We choose the Dynamic Time Warping distance (DTW) as our cost in order to account for the variable length of these trajectories. This distance between two time-series of possibly different sizes is designed to measure their similarity even when their dynamics differ(200, 2007). We define the DTW distance between two trajectories with a unit cost equal to the Euclidean L2 distance between two states $\in \mathcal{S}$.

Algorithm 1 Task-Agnostic Imitation Learning (TAIL)

```
1: Input: source domain trajectories $\mathcal{D}_S = \{\tau_n\}_{i=1}^N \sim \rho^\mu$
   and behavior policy $\mu$ if available
2: Initialize: undo map $u_\omega$, potentials $h_{\xi_1}, g_{\xi_2}$ and policy
   $\mu_\theta$ if $\mu$ is not available
3: while not done do
4:     sample target domain trajectories $\mathcal{D}_T = \{\tau_i\}_{i=1}^N \sim$
       $\rho^\pi$ where $\pi := \mu_\theta \circ u_\omega$
5:     for $k = 1 \dots K$ do
6:         compute
           $$\widetilde{\mathcal{W}} = \frac{1}{N}\sum_{\tau \in D_S} h_{\xi_1}(\tau) + \frac{1}{N}\sum_{\tau \in D_T} g_{\xi_2}(\tau) - \alpha \frac{1}{N^2} \sum_{\tau \in D_S, \tau' \in D_T} [h(\tau) + g(\tau') - DTW(\tau, \tau')]_+$$
7:         update potentials with $-\nabla_{\xi_1,\xi_2}\widetilde{\mathcal{W}}$
8:     end for
9:     update undo map $u_\omega$ with $\nabla_\omega \widetilde{\mathcal{W}}$
10:    update policy $\mu_\theta$ with $\nabla_\theta \widetilde{\mathcal{W}}$
11: end while
```

3.3. GAIL Interpretation

TAIL is designed to transfer a policy in one domain to another via the undo map, yet it bears strong similarities with GAIL, an algorithm meant to imitate an expert policy in the same domain. This resemblance can be observed by considering an alternative objective to Equation 3: $\min_{u \in \mathcal{U}} D_{JS}(\rho^\mu, u(\rho^\pi))$, where ρ now refers to state-action occupancy measures. We can further re-write this objective as follows:

$$\min_{u \in \mathcal{U}} \max_{f \in \mathcal{D}} \mathbb{E}_{\rho^\pi}[\log f(u(s,a))] + \mathbb{E}_{\rho^\mu}[\log(1 - f(s,a))]$$

While the imitator π plays the role of the generator in GAIL, the undo-map u plays the same role in TAIL. The discriminator f now classifies whether a state-action pair belongs to the target domain or the source domain.

4. Experiments

4.1. Reasoning in Grid World

In order to understand the properties of Algorithm 1, we first consider the navigation task of reaching the bottom-right corner of an 8-by-8 grid. In this grid world, we represent the state as the (x, y) position of the agent where $(0, 0)$ denotes

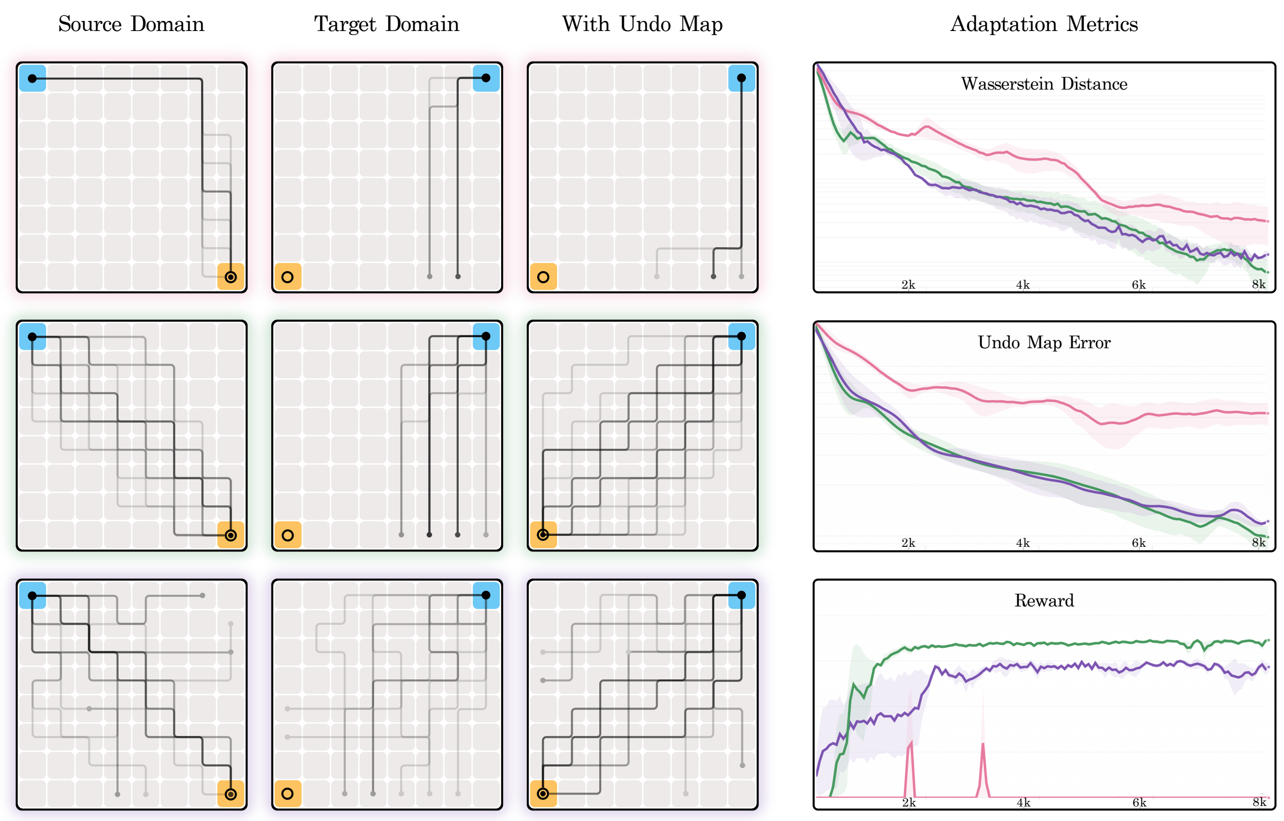

Figure 2. **Experiments.** We visualize the behavior of three different policies all trained in the source domain. For each policy, we also visualize its behavior in the target domain when deployed naively versus when deployed with the learned undo map.

its initial location. At any position in the grid, there are four available actions: move left, move right, move up, and move down. The agent always moves to the immediate cell in the direction of the chosen action unless it runs into a wall, in which case the agent remains at the same place. An episode terminates either when the agent reaches the goal $(7, 7)$ or $H = 50$ timesteps have passed. Finally, we provide a per-step reward of -1 in order to encourage the agent to reach the destination as quickly as possible.

From this source MDP $\mathcal{M}_S$, we define a related but different target MDP $\mathcal{M}_T$ by considering transformations to the state space of $\mathcal{M}_S$. In particular, we rotate the grid world in the original domain so in $\mathcal{M}_T$ every position $\binom{x}{y}$ appears to be $T_\theta\binom{x}{y}$ instead, where $T_\theta \in \mathbb{R}^{2\times 2}$ is a rotation matrix parameterized by a single angle θ. Although we consider a variety of angles, we only visualize experiments for $\theta = \frac{\pi}{2}$ because they are much easier to interpret. In this setting, the optimal undo map u_* rotates the grid world defined in $\mathcal{M}_T$ by $-\theta$ in order to undo the transformation between the two domains, therefore $u_\star\binom{x}{y} = T_\theta^{-1}\binom{x}{y}$. Since T_θ is an orthonormal matrix, the inverse T_θ^{-1} is equivalent to $T_\theta^\top$. We can easily observe that for any policy μ acting in the source domain, rolling out the the policy $\mu \circ T_\theta^\top$ in the target domain produces a trajectory distribution which when transformed with $T_\theta^\top$ would match the trajectory distribution of μ in the source.

4.2. Interacting with the Source and Target Domains

We assume knowledge of solving the task in $\mathcal{M}_S$ in the form of a parametric policy μ_θ. Although there are many policies that solve the task of reaching the destination in the grid, each one may take a different path. We specifically consider three kinds of interaction with the source domain.

Low-Entropy, Optimal Behavior The agent tends to take the same route to the destination, reaching there as quickly as possible but only ever visiting a few of the cells in the grid. We train a low-entropy, optimal policy in the source domain with PPO (Schulman et al., 2017).

High-Entropy, Optimal Behavior The agent takes various paths to reach the destination, visiting a majority of cells in the grid. Although each route is different, the agent reaches the target with minimal steps. We train a high-entropy, optimal policy in the source domain with a discrete action-space formulation of SAC (Haarnoja et al., 2018).

High-Entropy, Sub-optimal Behavior The agent again visits many cells in the grid, though this time it does not necessarily reach the destination as quickly as possible. At times, the agent even fails to reach the target and instead runs into a wall. In order to recover a high-entropy, sub-optimal policy, we train a policy in the source domain with SAC for only a few epochs.

We visualize each of the behaviors described above in our grid world. Figure 2 depicts three columns and three rows, where the first column shows the trajectory distribution of a policy μ_θ trained in the source domain, the second column shows the trajectory distribution of the same policy now acting in the target domain, and the last column shows the trajectory distribution of the policy $\mu_\theta \circ u_\omega$. Each row corresponds to a different policy μ_θ in the source domain trained to demonstrate either low-entropy and optimal behavior, high-entropy and optimal behavior, or high-entropy and sub-optimal behavior. A path or trajectory is represented as a line starting at the initial state, highlighted in blue, and ending possibly at the destination, highlighted in yellow, or another cell. Frequently occurring trajectories appear in a darker shade than others.

In the source domain, trajectories from a low-entropy, optimal policy tend to be the same, always moving along the edges of the grid, while trajectories from a high-entropy, optimal policy all end at the bottom-right corner of the grid in many different ways. In contrast, trajectories from a high-entropy, sub-optimal policy look chaotic, often times ending at a cell other than the destination. In the target domain, we observe how a policy trained originally in the source domain behaves when placed in the target domain. Note that the grid in the target domain is rotated by 90^o from the grid in the source domain. Regardless of the behavior in the source domain, we see that the same policy never reaches the destination. Every policy in the source domain fails to solve the task even once in the target domain.

4.3. Adapting with the Undo Map

We evaluate how well the policies μ_θ trained in the source domain $\mathcal{M}_S$ can be adapted to the target domain $\mathcal{M}_T$ with the undo map. Following the procedure in Algorithm 1, our agent acts in the rotated grid world according to the policy $a \sim \pi(\cdot|s) = \mu_\theta(\cdot|u_\omega(s))$. We visualize the trajectory distribution of $\mu_\theta \circ u_\omega$ in the last column of Figure 2. We also track three important metrics over the course of adaptation:

1. **Wasserstein Distance** We estimate $\widetilde{\mathcal{W}}_c(\rho^\mu, u(\rho^\pi))$ from sample trajectories in the source and target domains.
2. **Target Domain Return** We compute $\mathbb{E}_{\tau\sim\rho^\pi}[R(\tau)]$, where $R(\tau) := \sum_{t=0}^{H} \gamma^t r_t^T$ in order to measure the performance of $\pi := \mu_\theta \circ u_\omega$. We use rewards in the target domain only to compute this metric.
3. **Undo Map Error** In order to evaluate if we are learning the correct undo map, we calculate $\mathbb{E}_{(x,y)\sim\rho^\mu}\left[\left|\left|\binom{x}{y} - u_\omega \circ T_\theta \binom{x}{y}\right|\right|^2\right]$. Here, the distribution ρ^μ refers to the state visitation frequency or occupancy measure induced in the source domain when acting with μ_θ. Intuitively, this measures how close the composition $u_\omega \circ T_\theta$ is to the identity function provided that we have knowledge of T_θ, the transformation applied to the state space of the source domain in order to construct the target domain. We use T_θ only as a performance metric.

In our experiments, we find that it is difficult to recover the optimal undo map from a low-entropy, optimal policy in the source domain (plots shown in pink). Although the Wasserstein distance decreases during adaptation, the undo map error plateaus and the agent never learns to reach the destination. This is shown in the last grid of the first row of Figure 2 where trajectories in the target domain after using the undo map always run into a wall. Surprisingly, we find that training the undo map proves challenging only when the source policy has narrow state coverage. For instance, we are able to learn the undo map very well in the case where the source policy has high entropy, whether the behavior is optimal (plots shown in green) or sub-optimal (plots shown in purple). This reiterates the idea that the undo map depends on the source and target domains rather than an already trained, optimal policy in the original environment. A policy with enough state coverage in the source domain can learn the undo map even if the policy is not optimal. After learning the undo map u_ω in this way, we can later compose it with an optimal policy μ_θ in the source domain to construct an optimal policy $\mu_\theta \circ u_\omega$ in the target domain.

5. Conclusion

We have so far considered transfer settings where knowledge about the source domain is available as a parametric policy μ_θ. When only expert demonstrations $\{\tau_i\}_{i=0}^N$ from the source domain are available, we can again follow the procedure in Algorithm 1 with the exception that we must now learn the source policy μ_θ in addition to the undo map u_ω. This works well even when the demonstrations only cover a small portion of the state space. With these grid world experiments, we come to the conclusion that our algorithm has two principle use cases: solving the same task in a new domain from only demonstrations in the original domain or, more importantly, learning a task-agnostic undo map which allows for reuse but requires high state coverage demonstrators.

References

Dynamic time warping. In *Information Retrieval for Music and Motion*, pp. 69–84. Springer Berlin Heidelberg, 2007. doi: 10.1007/978-3-540-74048-3_4. URL https://

doi.org/10.1007/978-3-540-74048-3_4.

Abbeel, P. and Ng, A. Y. Inverse reinforcement learning., 2010.

Barreto, A., Hou, S., Borsa, D., Silver, D., and Precup, D. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1907370117. URL https://www.pnas.org/content/117/48/30079.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Mach. Learn.*, 79(1):151–175, May 2010. ISSN 0885-6125, 1573-0565. doi: 10.1007/s10994-009-5152-4.

Dadashi, R., Hussenot, L., Geist, M., and Pietquin, O. Primal wasserstein imitation learning. *arXiv preprint arXiv:2006.04678*, 2020.

Daumé, III, H. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pp. 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Fickinger, A., Cohen, S., Russell, S., and Amos, B. Cross-domain imitation learning via optimal transport. *arXiv preprint arXiv:2110.03684*, 2021.

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

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Linden, D. E. J., Kallenbach, U., Heinecke, A., Singer, W., and Goebel, R. The myth of upright vision. a psychophysical and functional imaging study of adaptation to inverting spectacles. *Perception*, 28:469 – 481, 1999.

Moskovitz, T., Arbel, M., Parker-Holder, J., and Pacchiano, A. Towards an understanding of default policies in multitask policy optimization. In Camps-Valls, G., Ruiz, F. J. R., and Valera, I. (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 10661–10686. PMLR, 28–30 Mar 2022a. URL https://proceedings.mlr.press/v151/moskovitz22a.html.

Moskovitz, T., Wilson, S. R., and Sahani, M. A first-occupancy representation for reinforcement learning. In *International Conference on Learning Representations*, 2022b. URL https://openreview.net/forum?id=JBAZe2yN6Ub.

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

Stratton, G. M. Some preliminary experiments on vision without inversion of the retinal image. *Psychological Review*, 3(6):611–617, 1896. doi: 10.1037/h0072918.

Stratton, G. M. Vision without inversion of the retinal image. *Psychological review*, 4(4):341, 1897.

Teh, Y., Bapst, V., Czarnecki, W. M., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N., and Pascanu, R. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, May 2016. ISSN 2196-1115, 2196-1115. doi: 10.1186/s40537-016-0043-6.

Xiao, H., Herman, M., Wagner, J., Ziesche, S., Etesami, J., and Linh, T. H. Wasserstein adversarial imitation learning. *arXiv preprint arXiv:1906.08113*, 2019.