A Learnable Similarity Metric for Transfer Learning with Dynamics Mismatch

Ram Ananth Sreenivasan [1] *, Hyun-Rok Lee [2], Yeonjeong Jeong [3], Jongseong Jang [4], Dongsub Shim [4], Chi-Guhn Lee [5]

[1] ArenaX Labs
[2] Inha University
[3] York University
[4] LG AI Research
[5] University of Toronto

ram.sreenivasan@mail.utoronto.ca , hyunrok.lee@inha.ac.kr , yjjeong@yorku.ca , {j.jang, dongsub.shim}@lgresearch.ai , cglee@mie.utoronto.ca

Abstract

When transferring knowledge from previously mastered source tasks to a new target task, the similarity between the source and target tasks can play a key role in whether such transfer is beneficial or harmful. In this paper, we develop an upper-bound of difference in action value function of source and target tasks with dynamics mismatch, and use the bound as a metric for dissimilarity between two tasks. The proposed metric does not require additional samples and adds little extra computation to the reinforcement learning algorithm for the target task. Also, the metric is highly portable so that it can be integrated into a wide range of algorithms. We showcase the effectiveness of the metric by incorporating it as a gatekeeper in the knowledge transfer step of transfer reinforcement learning algorithms. Numerical results on a suite of transfer learning scenarios demonstrate the benefits of preventing negative transfer in case of severe mismatch while accelerating learning otherwise.

Introduction

Reinforcement Learning (RL) has proven to be effective in a variety of tasks such as video games (Mnih et al. 2015) and robotic control (Gu et al. 2017). However, RL algorithms require a large number of samples, which is costly in case of real-world applications. Imagine that we want to train an expensive robot in a production line where the robot undergoes wear and tear over hundreds of thousands of rather random trials. Transfer learning seeks to utilize prior knowledge from tasks already tackled so as to reduce the additional training and samples in a new task (target task) (Taylor and Stone 2009). In this work, we focus on the transfer learning setting where the transition dynamics varies across source and target tasks. We further assume that the source task has already been solved. The chosen problem setting has several possible applications in practice. For example, knowledge about repair and replacement policies for a machine under different degradation conditions could be transferred to a scenario when only the degradation pattern is different from those observed before. Autonomous vehicles could reuse knowledge gained from other weather conditions when driving in unfamiliar weather conditions. RL agents could be trained for modeling pandemic spread with dynamics based on compartmental models in epidemiology for well-studied diseases (Kompella et al. 2020). This knowledge could be transferred to model pandemic spread of different diseases or when dynamics of pandemic spread change due to new variants of concern.

Knowledge transferred can be detrimental to learning of a target task if the source task is significantly different (Taylor and Stone 2009). Thus, it is essential to check task similarity before transferring knowledge across tasks. Ideally, knowledge transfer should be selective so that the learning of the target task should be accelerated whenever possible while avoiding negative transfer. The selective transfer requires a metric to measure similarity between tasks, and similarity metrics have been studied in two categories: model-based metrics and performance-based metrics (Visús, García, and Fernández 2021). Model-based metrics, most notably the bisimulation distance, have been shown to be effective for transfer RL (Castro and Precup 2010), but they are computationally expensive and require explicit modeling of the environment. Performance-based Similarity metrics, such as reuse gain (Fernández and Veloso 2006), can be learned online in the target task and are easy to compute. However, they do not consider the difference in components of the tasks such as the transition dynamics or reward function which could be utilized to more accurately measure similarity and instead measure similarity based on target task performance. These metrics are also more suited for multi-source task setting as a means of selecting the most appropriate task given multiple source tasks.

In this work, we focus on the development of a measure of (dis)similarity that combines the analytical benefits of model-based metric and lightweight nature of performance-based metrics by finding a relatively lightweight measure that analytically considers the difference of the components of the MDP across the tasks (more specifically the difference in dynamics). We define a quantity, termed as δ, to measure the impact of difference in dynamics between a source and a target task under a fixed policy. The defined quantity can be learned during target task learning, adding no extra burden with respect to target task samples and only minor additional computation. We demonstrate the utility of the proposed metric δ by adopting it as a gatekeeper in knowledge

*Work done while at University of Toronto

transfer pipeline from a source to a target task using two different transfer RL algorithms. The results show that the δ-based gatekeeper successfully filters out irrelevant knowledge to prevent negative transfer. The specific contribution of this work are as follows:

- We show analytically that the quantity δ is a upper bound of difference in action value functions of tasks with dynamics mismatch.
- We demonstrate the utility of δ when it is used to control knowledge transfer by augmenting existing algorithms with similarity information provided by δ.
- We experimentally validate that the algorithms augmented with δ can avoid negative transfer while maintaining the benefits of transfer.
- The proposed metric δ is highly portable and can be integrated into a wide range of existing reinforcement learning algorithms such as dynamic potential based advice (DPBA) method (Behboudian et al. 2020).

Related Work

Our work is closely related to literature regarding similarity metrics between MDPs. Similarity metrics across different MDPs could be categorized into two subsets; model-based metrics and performance-based metrics (Visús, García, and Fernández 2021). Model-based metrics require modelling of the environment to identify structural similarities across MDPs. Bisimulation metric has been used to compute a measure of task similarity for transfer learning (Castro and Precup 2010). Other model-based similarity metrics include constructing bipartite graphs of finite MDPs and using graph similarity measures (Wang, Dong, and Shao 2019) and aggregating compliance measure that calculates the probability that observed samples from the target task are generated by the source task (Lazaric, Restelli, and Bonarini 2008). However, the model-based similarity metric requires explicitly modeling the environment and hence computationally expensive. Our proposed metric for task similarity in the dynamics mismatch setting does not require explicit modelling and hence overcomes the limitation of the above-mentioned methods.

Performance-based metrics are viable alternatives to model-based metrics. These metrics include the mean squared error of the optimal action values after completing target task learning (Carroll and Seppi 2005) or matching latent embeddings of the action values of different tasks (Zhou and Yang 2020), which is useful when comparing among multiple source tasks but incapable of assessing the task similarity in any absolute sense. In this work, we learn a proxy of task similarity in an absolute sense so that the metric can be computed between a single source and a target task as well as used to assess relative similarity among multiple source tasks to a target task. The closest to our work is the Target Transfer Q learning (Wang et al. 2020), in which maximum norm of difference of optimal action value functions across tasks is computed. However, the task similarity in (Wang et al. 2020) depends on fluctuating action values. The proposed metric δ in this study is computed using samples to estimate expectation, and hence more stable.

Preliminaries

Markov Decision Processes Each task in the transfer learning scenario can be formulated as a Markov Decision Process (MDP). A MDP can be defined in terms of the tuple $M = \langle S, A, R, P, \gamma \rangle$, where S denotes the space containing all possible states, A denotes the space of actions, R represents the bounded scalar reward function $S \times A \mapsto R$; $P(s'|s,a)$ represents the transition probability corresponding to taking action a in state s resulting in a next state s' and $\gamma \in [0,1)$ represents the discount factor. A policy π is a mapping from S to a distribution over A that specifies the behavior of the RL agent.

The objective of the agent is to find an optimal policy that maximizes the expected return $J(\pi) = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k r(s_k, a_k)\Big]$. Rather than directly comparing $J(\pi)$, most RL algorithms estimate state value function $V^\pi(s)$ or action value function $Q^\pi(s,a)$ that can be defined by recursive structure based on bellman expectation equation.

By the principle of optimality, value functions of an optimal policy π^* satisfy the following relation.

$$V^*(s) = \max_a Q^*(s,a) = \max_a E_\pi\Big[r(s,a) + \gamma V^*(s'); s, a\Big]$$

Q-Learning and Deep Q-Networks Q-learning is an off-policy control algorithm that learns action values that converge to optimal action values (Watkins and Dayan 1992). It updates $Q(s,a)$ using reward r and next state s' information obtained from the environment by taking action a at state s.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \max_{a'} Q(s',a') - Q(s,a)) \quad (1)$$

Under the proper assumptions on learning rate α, $Q(s,a)$ converges to optimal action value $Q^*(s,a)$. For better estimation of $Q(s,a)$, ϵ-greedy policy is usually used as behavior policy in practice. It chooses a random action with probability ϵ and otherwise chooses $\max_a Q(s,a)$.

Deep-Q Network (DQN) is the extension of Q-learning using function approximation through a deep-neural network for high-dimensional state spaces (Mnih et al. 2015). While approximating action values as $Q_\theta(s,a)$, it exploits experience replay and fixed q-target to stabilize neural networks in learning action values.The useful output from DQN is the learned Q-network that estimates $Q^*(s,a)$.

Delta Function

Delta Function Definition

The target task and the source task are denoted as MDPs $M_{trg} = \langle S, A, R, P_{trg}, \gamma \rangle$ and $M_{src} = \langle S, A, R, P_{src}, \gamma \rangle$, respectively. We define a quantity $\delta^\pi(s,a)$ as

$$\delta^\pi(s,a) = \gamma \sum_{s' \in S} \left(p_{trg}(s'|s,a) - p_{src}(s'|s,a)\right) V^\pi_{src}(s')$$

where $V^\pi_{src}(s)$ is the state value function of policy π evaluated at task M_{src}. Intuitively, larger δ value implies larger dissimilarity of dynamics between the two tasks at (s,a).

The defined term $\delta^\pi(s,a)$ allows us to infer the influence of dynamics mismatch at (s,a) on the change in action value

functions when a policy π is deployed at target task. By definitions of action values and $\delta^\pi(s,a)$, differences of action values of π in different tasks satisfy the following relation;

$$Q^\pi_{trg}(s,a) - Q^\pi_{src}(s,a) = \delta^\pi(s,a) + \gamma \sum_{s' \in S} p_{trg}(s'|s,a)\{V^\pi_{trg}(s') - V^\pi_{src}(s')\}$$

These relations can lead to bounds on the difference in action values and a detailed discussion is provided in the Appendix.

The optimal Q values can also be bounded in terms of $\delta^\pi(s,a)$ as shown in Theorem 1. Because one measure of similarity between two MDPs M_{src} and M_{trg} can be captured using the maximum difference in their optimal action values (Wang et al. 2020), Theorem 1 implies that $\delta^{\pi^*_{src}}$ provides an upper bound on the task (dis-)similarity $\Delta(M_{src}, M_{trg}) = \max_{s,a} \left| Q^*_{src}(s,a) - Q^*_{trg}(s,a) \right|$.

Theorem 1. *Define π^*_{trg} as the target optimal policy and π^*_{src} as the source optimal policy. $Q^\pi(s,a)$ denotes the action value of following policy π for state s and action a. Then,*

$$Q^{\pi^*_{trg}}_{trg}(s,a) - Q^{\pi^*_{src}}_{src}(s,a) \leq \Delta(M_{src}, M_{trg}) \leq \frac{1}{1-\gamma} \max_{(s,a)} \left| \delta^{\pi^*_{src}}(s,a) \right|$$

Note that our proposed (dis-)similarity metric according to Theorem 1 is basically an upper bound on the difference of action value function between the source and the target tasks. Moreover, the computation of the bound requires an maximization of $\delta^{\pi^*_{src}}(s,a)$ over all combinations of s and a.

Computing Delta Function

The most naive approach to computing δ would be to learn the dynamics model of the source and target tasks. By using such a method, it is possible to calculate approximations $\hat{p}_{trg}(s'|s,a)$ and $\hat{p}_{src}(s'|s,a)$ corresponding to target task dynamics and source task dynamics, respectively. We seek to avoid the overhead of explicitly modeling the environment at both the source and target task while learning δ, thereby attempting to make the computation of δ more scalable. We show an alternate method to compute δ by utilizing the definition of δ and the fact that the reward function is the same across tasks.

By the definition of δ, for the transfer learning scenario with dynamics mismatch we have

$$\begin{aligned}
\delta^\pi(s,a) &= \gamma \sum_{s' \in S} (p_{trg}(s'|s,a) - p_{src}(s'|s,a)) V^\pi_{src}(s') \\
&= r(s,a) + \gamma \sum_{s' \in S} p_{trg}(s'|s,a) V^\pi_{src}(s') - r(s,a) \\
&\quad - \gamma \sum_{s' \in S} p_{src}(s'|s,a) V^\pi_{src}(s') \\
&= E_{s' \sim P_{trg}}[r(s,a) + \gamma V^\pi_{src}(s'); s, a] - Q^\pi_{src}(s,a)
\end{aligned}$$

Since δ can be written as an expectation in terms of target task transition dynamics, it can be estimated via stochastic approximation in an off-policy manner with target task samples used for training the baseline RL algorithm. The following update rule can be used to estimate δ

$$\delta^\pi(s,a) \leftarrow \delta^\pi(s,a) + \alpha_d \beta_\delta \tag{2}$$

where

$$\beta_\delta = \{r(s,a) + \gamma V^\pi_{src}(s') - Q^\pi_{src}(s,a) - \delta^\pi(s,a)\}$$

and α_d is learning rate for estimating $\delta^\pi(s,a)$.

We utilize δ as it provides several advantages for transfer RL algorithms in dynamics mismatch setting. First, task similarity $\Delta(M_{src}, M_{trg})$ defined with the differences of optimal action values in each task cannot be measured before knowing an optimal policy at the target environment. Using estimates of $\Delta(M_{src}, M_{trg})$ as such could lead to instability due to fluctuating Q values during learning which is mitigated when using δ, since δ simply uses target samples to compute the expectation. $\delta^{\pi^*_{src}}$ defined with source optimal policy gives an upper bound on task similarity that does not require knowledge of target optimal policy (Theorem 1). Hence, we use $\delta^{\pi^*_{src}}$ as an indicator of task similarity while learning policy at target environment. Second, to measure task similarity between source and target tasks, we can not avoid collecting transition samples at target task but there is an inherent requirement to minimize the need for additional samples required for task similarity estimation. Because δ can be computed with transition samples collected for learning target policy, we do not need extra interactions with target environment to identify task similarity. Estimation of δ can be added to any transfer RL algorithm without additional sampling costs to determine task similarity. Third, computation of δ doesn't require any explicit modelling of the environment unlike model-based similarity metrics.

Delta Function in Transfer RL with Dynamics Mismatch

We provide two simple yet effective use cases of δ in the single source task transfer setting. It is more critical in single source task setting than multi source transfer to judge whether source knowledge is beneficial. In multi-source transfer, we can selectively choose the best source with the expectation of having beneficial knowledge among multiple sources. Although δ can be used in multi source transfer setting, we restrict to a single source transfer where source knowledge is not guaranteed to be beneficial and avoiding negative transfer is more crucial. We use $\delta^{\pi^*_{src}}$ as an indicator of task similarity to identify whether to transfer source knowledge. The first method uses $\delta^{\pi^*_{src}}$ to decide whether to reuse source policy directly in target task. Starting from the initial prior that source and target task are similar, we estimate $\delta^{\pi^*_{src}}$ value to monitor the efficiency of policy reuse. The second method is controlling the effects of action advice based on $\delta^{\pi^*_{src}}$ while source knowledge is used in action selection. These methods aim to prevent negative transfer while keeping positive bias from source knowledge. We slightly abuse notation and use δ and $\delta^{\pi^*_{src}}$ interchangeably.

Delta as Criteria for Policy Reuse

The simplest way of transferring source knowledge is directly deploying a source policy at target environment. Suppose we have set up a target task and decide to transfer source policy for this task. This setup usually begins with the prior belief that the source policy will perform well at target because those two tasks are similar. Direct transfer could be a sufficiently effective transfer method unless source and target tasks are quite dissimilar. Hence, a more important question here is how to prevent negative transfer if the source policy unexpectedly performs poorly.

Algorithm 1: Q-learning with $\delta^{\pi^*_{src}}$ as a measure of task similarity

Input γ, T_{max}, Q^*_{src}, ϵ, Δ, learning rates α, α_d
Initialize $Q_{trg}(s,a) = 0, \delta^{\pi^*_{src}}(s,a) = 0 \; \forall s \in S, a \in A$, and set initial state s_0
for $t \leq T_{max}$ **do**
  if $\max_{(s,a)} \left|\delta^{\pi^*_{src}}(s,a)\right| \leq \Delta$ **then**
    $a_t = \arg\max_a Q^*_{src}(s,a)$
  else
    a_t = ϵ-greedy policy corresponding to Q_{trg}
  end if
  Execute a_t and observe reward r and next state s_{t+1}
  Update $\delta^{\pi^*_{src}}(s_t, a_t)$ according to equation 2
  Update $Q_{trg}(s_t, a_t)$ according to equation 1
  $t \leftarrow t+1$
end for

Algorithm 1 shows how we can add a mechanism to prevent negative transfer using $\delta^{\pi^*_{src}}$ while directly using source policy at target environment. By simply adding a criterion of comparing a threshold value Δ to $\max_{(s,a)} \left|\delta^{\pi^*_{src}}\right|$, we choose whether to follow source policy or current target policy. Note that this criterion more conservatively follows source policy than using the same threshold value on the task similarity at the convergence according to Theorem 1. Starting with the prior belief that source and target tasks are equal ($\delta^{\pi^*_{src}}(s,a) = 0$), we update this belief with newly collected target transition samples. Then, task similarity is more clearly identified as the $\delta^{\pi^*_{src}}$ converges. Compared to directly reusing source optimal policy in the target task, Algorithm 1 prevents negative effects of following source policy obtained from a dissimilar task while also taking advantage of source optimal policy if tasks are found to be similar.

Delta as Advice Control Factor

Policy invariant explicit shaping (PIES) (Behboudian et al. 2020) has been proposed to utilize arbitrary advice while keeping policy invariant property by explicitly shaping the action selection instead of modifying the reward function. The action selection is modified to include a bias based on the potential function $\phi^\dagger(s,a)$ such that the selected action is given by

$$a = \arg\max_a Q(s,a) - \kappa_t \Phi^\dagger(s,a) \tag{3}$$

where κ_t denotes the relative importance given to the advice.Because PIES preserves policy optimality and could be extended to function approximation setting, this method provides a good baseline framework to incorporate $\delta^{\pi^*_{src}}$ for continuous state space environments. For using δ in continuous state space tasks, we parameterize δ as neural network and estimate it by minimizing the loss function:

$$L_\delta = E_{\tau\sim D}\left(r + \gamma \max_b Q^*_{src}(s',b) - Q^*_{src}(s,a) - \delta(s,a)\right)^2 \tag{4}$$

Although the PIES framework provides a mechanism on how to reuse arbitrary information, there is also the necessity of a mechanism to decide when to reuse such information without detrimental effects for the transfer learning setting. We utilize $\delta^{\pi^*_{src}}$ to control the effects of action advice $\Phi^\dagger(s,a)$ based on a proxy for task similarity, by adding an additional factor to control the relative importance of the bias term, as follows:

$$a = \arg\max_a Q(s,a) - \kappa_t \exp\left(-\max_a |\delta(s,a)|\right)\Phi^\dagger(s,a) \tag{5}$$

The term κ_t, where t denotes the number of training steps completed, is decayed to zero using a constant C, starting with $\kappa_0 = 1$ as follows (Behboudian et al. 2020)

$$\kappa_t = max(0, \kappa_{t-1} - C) \tag{6}$$

The additional factor $\exp\left(-\max_a \left|\delta^{\pi^*_{src}}(s,a)\right|\right)$ is used since a smaller value of $\max_a \left|\delta^{\pi^*_{src}}(s,a)\right|$ corresponds to tasks being more similar. This particular action selection mechanism reduces the effects of bias term $\Phi^\dagger(s,a)$ based on source knowledge on action selection substantially if $\max_a \left|\delta^{\pi^*_{src}}(s,a)\right|$ is large. This particular usage of $\delta^{\pi^*_{src}}$, where the maximum value is taken over the actions only for the particular state, is a local and contextual approximation used to ensure scalability. This factor is empirically observed to work well, providing useful action advice in case of using similar source task and preventing negative transfer in case of large dynamics discrepancy setting. The entire description of the algorithm is provided in Algorithm 2.

Unlike the base algorithm of PIES, we use static potential functions as advice rather than learning a separate secondary action value as it requires on-policy samples. The type of information reused from the source task as advice can directly affect the performance of the target task agent. For the scenario of transfer learning, it can be beneficial to bias the different actions differently. The source optimal advantage function is chosen such that $\phi^\dagger(s,a) = -A^*_{src}(s,a)$ because it expects to provide a bias of scale similar to the action value function that distinguishes the actions suboptimal for the source task when compared to using the source policy.

Experiments

In this section, we provide experiments that test our hypothesis that δ can be used as a proxy for task similarity. We mainly validate the utility of δ as a similarity metric by

Algorithm 2: DQN with Similarity based Explicit Shaping

Input γ, T_{max}, Q^*_{src}, ϵ, learning rate α_d
Initialize Neural network parameters θ_δ, θ_Q, and replay buffer D
for $t \leq T_{max}$ **do**
  a_t = ϵ-greedy with respect to action selection mechanism in equation 5
  Decay κ_t based on decay rule in equation 6
  Execute a_t and observe reward r and next state s_{t+1}
  Store (s_t, a_t, r_t, s_{t+1}) in D
  Update θ_δ to minimize loss in equation 4 using gradient descent: $\theta_\delta \leftarrow \theta_\delta - \alpha_d \nabla_{\theta_\delta} L_\delta$
  Sample minibatch B of (s, a, r, s') from D
  Update θ_Q based on DQN algorithm with random minibatch B
  $t \leftarrow t + 1$
end for

showing that algorithms augmented with such a task similarity can selectively avoid negative transfer while maintaining the positive biases otherwise. We validate the hypothesis using the two different algorithms that showcase the utility of δ using tasks with discrete (2 different tasks with 2 transfer learning scenarios each) as well as continuous state spaces (4 different tasks with 2 transfer learning scenarios each) where dynamics varies across source and target task. In particular, we focus on answering the following questions through our experiments:

1. Does using δ help capture the similarity or differences across the source and target task?
2. Can augmenting the two different algorithms with δ selectively incorporate source knowledge by considering when to transfer?

The source task is solved till convergence using baseline RL algorithms of Q-Learning and DQN for discrete and continuous state spaces respectively and the resultant action values are utilized to compute necessary source task knowledge.

In order to measure the relative improvement in performance of the transfer learning algorithms, we utilize the transfer ratio (Taylor and Stone 2009). Given the area under the learning curve (AUC) with transfer, A_t and AUC without transfer, A_w, transfer ratio tr can be calculated as [1]

$$tr = \frac{A_t - A_w}{|A_w|}$$

A positive value of tr implies that the transfer learning improved learning compared to not using transfer whereas a negative value of tr implies that transfer learning harms target task learning and indicates negative transfer.

Delta as Criteria for Policy Reuse

For the first use case of δ in Algorithm 1, we focus on two discrete state-action space benchmark tasks, one being a modified version of Windy Gridworld (Sutton and Barto 2018) and the other being a supply chain management (SCM) task (Kemmer et al. 2018). A full description of the tasks and transfer learning scenarios is mentioned in the Appendix.

[1]Deviating from the original definition, we used absolute value in the denominator to preserve interpretation of relative improvement when A_w is negative.

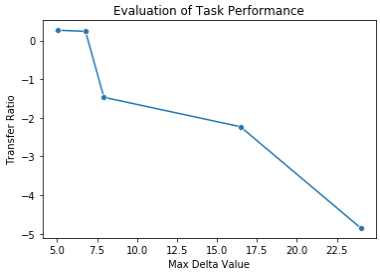


Figure 1: Evaluation of delta as a measure of similarity

We first empirically evaluate how $\max_{(s,a)} \left|\delta^{\pi^*_{src}}\right|$ indicates if direct transfer is useful or not thus acting as a proxy of how (dis)similar the tasks are. This evaluation is carried out in the Windy Gridworld environment by choosing various pairs of (M_{src}, M_{trg}) and directly reusing the source optimal policy π^*_{src} in the target task. In Figure 1, we plot the performance of reusing π^*_{src} as measured by the transfer ratio tr. As $\max_{(s,a)} \left|\delta^{\pi^*_{src}}\right|$ increases, it can be seen that reusing π^*_{src} performs poorer indicating that the tasks are more different. It can be inferred from this performance that using a proxy based on δ captures the (dis)similarities across source and target task based on the intuition that if target task and source task are more similar directly reusing π^*_{src} results in better target task performance.

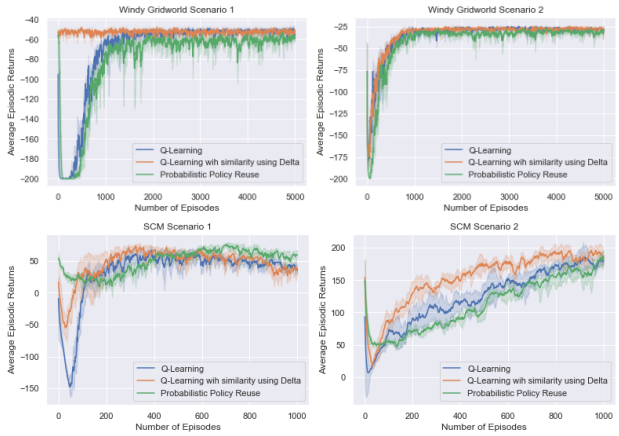


Figure 2: Average evaluation performance computed using greedy policy. The shaded area represents 95% Confidence Interval of the results for 5 different trials

Based on the validation that $\max_{(s,a)} \left|\delta^{\pi^*_{src}}\right|$ can determine if π^*_{src} is useful or not, we evaluate the performance of Algorithm 1. The results for the two different transfer scenarios are shown in Figure 2. It can be observed that for cases where positive gains from transfer are possible, like Windy Gridworld scenario 1, the algorithm learns to

Table 1: Average AUC score over five different random seeds and transfer ratio tr calculated based on average in percentage. Parentheses indicate standard deviation.

	DQN		ES		δ based ES	
Environment (Factor)	AUC	$tr(\%)$	AUC	$tr(\%)$	AUC	$tr(\%)$
CartPole (5)	139.4 (11.7)	0	157.0 (11.1)	12.9	163.8 (14.1)	**17.8**
CartPole (7)	131.8 (10.9)	0	127.6 (14.2)	-2.9	158.6 (2.4)	**21.0**
Acrobot (1.5)	-172.7 (26.9)	0	-161.3 (14.4)	6	-150.1 (5.0)	**13.0**
Acrobot (2)	-228.6 (12.0)	0	-233.0 (20.4)	-1.9	-228.4 (15.6)	**0.1**
LunarLander (1.1)	178.3 (13.8)	0	171.1 (7.8)	-4	181.5 (7.4)	**1.7**
LunarLander (0.75)	223.5 (6.8)	0	219.8 (11.0)	-1.63	228.0 (5.0)	**3.70**
Reacher (0.75)	-5.96 (0.1)	0	-5.71 (0.1)	3.9	-5.66 (0.08)	**4.8**
Reacher (1)	-6.38 (0.28)	0	-6.17 (0.09)	3.3	-6.09 (0.22)	**4.4**

directly reuse π^*_{src} very quickly. On the other hand, for cases like Windy Gridworld scenario 2, where π^*_{src} would perform poorly, the algorithm learn to conservatively select the current target policy and avoid the reuse of π^*_{src}. For both scenarios for SCM environment, the algorithm learns to achieve an initial jumpstart by learning about task similarity thus showcasing the benefits of transfer. The algorithm, in general, improves over both the"no-transfer" condition (Q-learning) and probabilistic policy reuse (Fernández and Veloso 2006) which probabilistically ("random" source selection) decides whether to use the source optimal policy, compared to the similarity-aware Algorithm 1

Delta as Advice Control Factor

The proposed method of using δ as advice control factor is tested using different classic control benchmarks provided by OpenAI gym (Brockman et al. 2016) namely **CartPole, Acrobot and LunarLander**. Additionally, a continuous control robotics system, **Reacher** is used with a discretized action space (Barreto et al. 2017). The default OpenAI gym environments act as source tasks and target tasks are defined by changing the transition dynamics through environment specific parameters, the details of which are explained in the appendix. We study two transfer learning scenarios for each environment and compare **DQN**, DQN with transfer learning based PIES, referred henceforth as **ES**, and also with the proposed modification of using δ to decide how to shape action selection based on task similarity, referred to as δ**-based ES**. The hyperparameters used for the different experiments are provided in the Appendix. The algorithms are compared based on the transfer ratio for the AUC score obtained by evaluating the greedy policy at fixed intervals during the learning process and the result is tabulated in Table 1. where Factor indicates by how much the environment specific parameters have been changed to form the target task

It can be observed that adding the δ-based term maintains or improves performance of explicit shaping with respect to tr on average. In cases where negative transfer is possible like Acrobot with link mass twice as the source task, the addition of δ prevents the negative transfer and performs similarly to the DQN baseline whereas the explicit shaping algorithm results in negative transfer. For Acrobot with link mass 1.5 times that of the source task, the δ based algorithm improves slightly on the benfits that transfer provides. Similarly, for CartPole, the proposed modifications maintain the benefits when it is useful to transfer, namely the transfer scenario where target task gravity is 5 times that of the source task. A positive transfer is maintained by the algorithm augmented with δ when the source information is slightly detrimental too, namely the transfer scenario where target task gravity is 7 times that of the source task, whereas the explicit shaping algorithm results in negative transfer. For LunarLander, the explicit shaping mechanism fails partially in both the transfer learning scenarios, whereas there is still some positive gains due to using the additional factor based on δ. For the discretized version of Reacher, the algorithm augmented with δ maintains the positive bias from transfer for both transfer learning scenarios.

Conclusions

In this work, we utilized an easy-to-compute quantity, δ, to capture the difference in dynamics across tasks and showed it to be a proxy for task similarity between source and target tasks for transfer learning with dynamics mismatch. We augment existing algorithms - Algorithm 1 using the policy re-use and Algorithm 2 using the explicit reward shaping - with task similarity based on the quantity δ to better understand when the relevant source task information can be incorporated into the learning process and to address the question of "when to transfer". We would highlight that the proposed metric can be adopted by a wide range of existing algorithms.

Experiments on a range of tasks and transfer learning scenarios show that utilizing δ helps maintain positive bias from source information in cases where it is beneficial to transfer and selectively avoids source information when it can be detrimental. Even though δ can be used as a proxy for task similarity, it only takes into consideration the difference in one-step transition dynamics. Also, overcoming the computational intractability of taking a maximum over δ would give a better opportunity for a role of δ in other transfer RL

algorithms beyond its natural application in discrete action-space domains. In future work, we aim to utilize δ to decide the most appropriate source task given a library of multiple available source tasks. The task differences based on δ could also be used to ground the source task to more closely match the target task by minimizing task differences (Desai et al. 2020) and avoiding the explicit computation of the transition models for source or target task.

Limitations and Future Work Transfer learning for RL, especially transfer learning with dynamics mismatch, must be used with utmost caution as it can be used to transfer knowledge from simulators to real-world systems. When doing so, exploration can be risky in terms of damage to the real-world system or damage to its surroundings which could include humans. Hence, proper precautions must be taken to ensure safe deployment, especially in safety-critical settings like autonomous vehicles or healthcare.

Acknowledgments

This research was supported by LG AI Research.

References

Barreto, A.; Dabney, W.; Munos, R.; Hunt, J. J.; Schaul, T.; van Hasselt, H. P.; and Silver, D. 2017. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, 4055–4065.

Behboudian, P.; Satsangi, Y.; Taylor, M. E.; Harutyunyan, A.; and Bowling, M. 2020. Useful Policy Invariant Shaping from Arbitrary Advice. *arXiv preprint arXiv:2011.01297*.

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

Carroll, J. L.; and Seppi, K. 2005. Task similarity measures for transfer in reinforcement learning task libraries. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, 803–808. IEEE.

Castro, P.; and Precup, D. 2010. Using bisimulation for policy transfer in MDPs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24.

Desai, S.; Durugkar, I.; Karnan, H.; Warnell, G.; Hanna, J.; Stone, P.; and Sony, A. 2020. An Imitation from Observation Approach to Transfer Learning with Dynamics Mismatch. *Advances in Neural Information Processing Systems*, 33.

Fernández, F.; and Veloso, M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 720–727.

Gimelfarb, M.; Sanner, S.; and Lee, C.-G. 2020. Epsilon-BMC: A Bayesian Ensemble Approach to Epsilon-Greedy Exploration in Model-Free Reinforcement Learning. In *Uncertainty in Artificial Intelligence*, 476–485. PMLR.

Gu, S.; Holly, E.; Lillicrap, T.; and Levine, S. 2017. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, 3389–3396. IEEE.

Kemmer, L.; von Kleist, H.; de Rochebouët, D.; Tziortziotis, N.; and Read, J. 2018. Reinforcement learning for supply chain optimization. In *European Workshop on Reinforcement Learning*, volume 14.

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

Kompella, V.; Capobianco, R.; Jong, S.; Browne, J.; Fox, S.; Meyers, L.; Wurman, P.; and Stone, P. 2020. Reinforcement Learning for Optimization of COVID-19 Mitigation policies. *arXiv preprint arXiv:2010.10560*.

Lazaric, A.; Restelli, M.; and Bonarini, A. 2008. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, 544–551.

Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J. E.; and Stoica, I. 2018. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv preprint arXiv:1807.05118*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533.

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

Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul): 1633–1685.

Visús, Á.; García, J.; and Fernández, F. 2021. A Taxonomy of Similarity Metrics for Markov Decision Processes. *arXiv preprint arXiv:2103.04706*.

Wang, H.; Dong, S.; and Shao, L. 2019. Measuring Structural Similarities in Finite MDPs. In *IJCAI*, 3684–3690.

Wang, Y.; Liu, Y.; Chen, W.; Ma, Z.-M.; and Liu, T.-Y. 2020. Target transfer q-learning and its convergence analysis. *Neurocomputing*.

Watkins, C. J.; and Dayan, P. 1992. Q-learning. *Machine learning*, 8(3-4): 279–292.

Zhou, Y.; and Yang, F. 2020. Latent Structure Matching for Knowledge Transfer in Reinforcement Learning. *Future Internet*, 12(2): 36.

Appendix

A Proofs

In order to prove the theorems, we assume that the source task and target task differ only with respect to the transition dynamics. The source task can be denoted as the MDP defined by the tuple $M_{src} = \langle S, A, R, P_{src}, \gamma \rangle$ and the target task can be denoted as $M_{trg} = \langle S, A, R, P_{trg}, \gamma \rangle$. Let $\Delta(M_{src}, M_{trg}) = \max_{s,a} \left| Q^*_{src}(s,a) - Q^*_{trg}(s,a) \right|$ where $Q^*(s,a)$ represents the optimal action value function for state-action pair (s,a) and following optimal policy π^* thereafter.

Theorem 2. *Define π^*_{trg} as optimal policy for target task and π^*_{src} as optimal policy in source task. $Q^\pi(s,a)$ denotes the action value of following policy π for state s and action a. Then*

$$Q^{\pi^*_{trg}}_{trg}(s,a) - Q^{\pi^*_{src}}_{src}(s,a) \leq \Delta(M_{src}, M_{trg}) \leq \frac{1}{1-\gamma}\max_{(s,a)}\left|\delta^{\pi^*_{src}}(s,a)\right|$$

Proof. The proof is analogous to Lemma 1 in (Barreto et al. 2017)

Let $\max_{(s,a)}\left|Q^{\pi^*_{trg}}_{trg}(s,a) - Q^{\pi^*_{src}}_{src}(s,a)\right| = \Delta$. $\left|Q^{\pi^*_{trg}}_{trg}(s,a) - Q^{\pi^*_{src}}_{src}(s,a)\right|$ is nothing but the difference between action values of the two MDPs.

$$\begin{aligned}
&\left|Q^{\pi^*_{trg}}_{trg}(s,a) - Q^{\pi^*_{src}}_{src}(s,a)\right| \\
&= \Bigg|r(s,a) + \gamma\sum_{s'\in S} p_{trg}(s'|s,a)\max_b Q^{\pi^*_{trg}}_{trg}(s',b) \\
&- r(s,a) - \gamma\sum_{s'\in S} p_{src}(s'|s,a)\max_b Q^{\pi^*_{src}}_{src}(s',b)\Bigg| \\
&= \Bigg|\gamma\sum_{s'\in S}[\{p_{trg}(s'|s,a) - p_{src}(s'|s,a)\}\max_b Q^{\pi^*_{src}}_{src}(s',b) \\
&+ p_{trg}(s'|s,a)\{\max_b Q^{\pi^*_{trg}}_{trg}(s',b) - \max_b Q^{\pi^*_{src}}_{src}(s',b)\}]\Bigg| \\
&= \Bigg|\gamma\sum_{s'\in S} p_{trg}(s'|s,a)\{\max_b Q^{\pi^*_{trg}}_{trg}(s',b) - \max_b Q^{\pi^*_{src}}_{src}(s',b)\} \\
&+ \delta^{\pi^*_{src}}(s,a)\Bigg| \\
&\leq \gamma\sum_{s'\in S} p_{trg}(s'|s,a)\left|\max_b Q^{\pi^*_{trg}}_{trg}(s',b) - \max_b Q^{\pi^*_{src}}_{src}(s',b)\right| \\
&+ \left|\delta^{\pi^*_{src}}(s,a)\right| \\
&\leq \gamma\sum_{s'\in S} p_{trg}(s'|s,a)\max_b\left|Q^{\pi^*_{trg}}_{trg}(s',b) - Q^{\pi^*_{src}}_{src}(s',b)\right| \\
&+ \left|\delta^{\pi^*_{src}}(s,a)\right| \\
&\leq \left|\delta^{\pi^*_{src}}(s,a)\right| + \gamma\Delta \qquad (7)
\end{aligned}$$

Inequality 7 holds even when we take max operator over (s,a) in both sides.

Therefore replacing LHS of inequality 7 with Δ we get,

$$\Delta \leq \max_{(s,a)}\left|\delta^{\pi^*_{src}}(s,a)\right| + \gamma\Delta$$

$$\Delta \leq \frac{1}{1-\gamma}\max_{(s,a)}\left|\delta^{\pi^*_{src}}(s,a)\right| \qquad (8)$$

□

The following relation leads to bounds on action value differences which could be computed without deploying π at target task. This can be a different use case of δ where policies can be evaluated at target task without deploying them. This could be useful in settings such as safety-critical systems where rolling out the policy may be catastrophic.

Theorem 3. *Bounds on differences of action values for the same π evaluated in different tasks can be represented with δ for the current time step and maximum (minimum) of δ.*

$$\delta^\pi(s,a) + \frac{\gamma}{1-\gamma}\min_{(s,a)}\delta^\pi(s,a) \leq Q^\pi_{trg}(s,a) - Q^\pi_{src}(s,a) \leq \delta^\pi(s,a) + \frac{\gamma}{1-\gamma}\max_{(s,a)}\delta^\pi(s,a)$$

Proof. By the definition of δ, the state-action values and δ satisfy the following relation,

$$\begin{aligned}
&Q^\pi_{trg}(s,a) - Q^\pi_{src}(s,a) = \\
&\delta^\pi(s,a) + \gamma E_{\pi,P_{trg}}\Big[\sum_{l=1}^{\infty}\gamma^{l-1}\delta^\pi(s_{k+l},a_{k+l}); s_k = s, a_k = a\Big] \\
&\leq \delta^\pi(s,a) + \gamma E_{\pi,P_{trg}}\Big[\sum_{l=1}^{\infty}\gamma^{l-1}\max_{(s,a)}\delta^\pi(s,a)\Big] \\
&= \delta^\pi(s,a) + \frac{\gamma}{1-\gamma}\max_{(s,a)}\delta^\pi(s,a)
\end{aligned}$$

Similarly,

$$\begin{aligned}
&Q^\pi_{trg}(s,a) - Q^\pi_{src}(s,a) \\
&\geq \delta^\pi(s,a) + \gamma E_{\pi,P_{trg}}\Big[\sum_{l=1}^{\infty}\gamma^{l-1}\min_{(s,a)}\delta^\pi(s,a)\Big] \\
&= \delta^\pi(s,a) + \frac{\gamma}{1-\gamma}\min_{(s,a)}\delta^\pi(s,a)
\end{aligned}$$

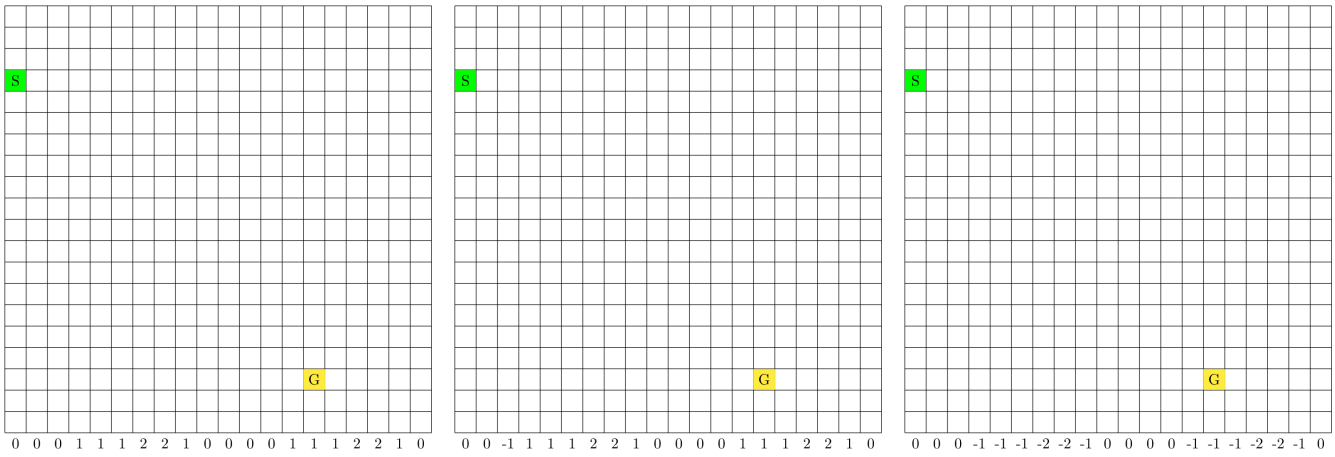
□

B Environment Details and Design of Transfer Learning Scenarios

Delta as Criteria for Policy Reuse

Windy GridWorld The Windy GridWorld environment is a 20 by 20 gridworld with prespecified start and goal states. The four possible actions are $\{up, down, left, right\}$. The actions move the agent by one cell in the corresponding direction with a probability of 0.75, otherwise the agent moves in a random direction. Additionally the agent movements are modified due to the presence of "wind" causing the movements to be shifted upwards based on the strength of the "wind" mentioned under each column in Figure 3a. If a wall is encountered based on the chosen action, the agent stays in the cell next to the wall. The agent incurs a penalty of -1 for each step till it reaches the goal state thereby enforcing the goal of reaching the goal state in least number of steps. The target task for different transfer learning scenarios corresponding to Windy Gridworld scenarios 1 and 2 are depicted in Figure 3b and 3c respectively

In order to empirically evaluate if $\max_{(s,a)}\left|\delta^{\pi^*_{src}}\right|$ can determine if directly reusing source policy works well,based



(a) (b) (c)

Figure 3: Depiction of the Windy Gridworld environment used in the experiments with numbers under each column denoting the strength of the wind in the upward direction for each column (a) Source Task (b) Small Changes in Wind leading to target task for Scenario 1 (c) Target task corresponding to the wind direction being reversed compared to the source task which is used in Scenario 2

on Figure 1 in the manuscript , two additional target tasks are utilized and in total 5 different target tasks are used including the ones in Figures 3a,3b,3c. The additional target tasks used are tabulated in Table 2. The source task used for all cases is the one shown in Figure 3a. For each source task, target task pair, the source optimal policy is directly deployed and the performance is compared with Q Learning, trained for 5000 episodes with hyperparameters as given in Table 4, based on the transfer ratio tr

Supply Chain Management The Supply Chain Management (SCM) environment consists of a factory and a warehouse (Kemmer et al. 2018; Gimelfarb, Sanner, and Lee 2020). The agent's goal is to meet the demand for a particular good by deciding how much to produce at the factory and how much should be shipped to the warehouse. The agent is rewarded based on revenue generated, using a price per unit of 0.5 and penalized based on per unit production cost of 0.1, per unit storage cost of 0.02, per unit penalty cost of 0.1 when the stock level in warehouse is less than 0, and transportation cost based on number of trucks with each truck having a capacity of 5 and costing 0.1. Initially, a quantity of 10 goods are assumed to be available in the factory and the warehouse is empty. The factory and warehouse have a storage limit of 50. The transportation limits are set to 10 for each period and unsatisfied demand is not logged for later but assumed to be lost. The demand is assumed to follow a Poisson distribution with $\lambda = 2.5$ for the source task and $\lambda = 2$, $\lambda = 5$ for transfer learning scenarios 1 and 2 respectively.

Delta as Advice Control Factor

The proposed method of using δ as advice control factor is tested using different classic control benchmarks provided by OpenAI gym (Brockman et al. 2016) namely **CartPole, Acrobot and LunarLander**.Additionally, a continuous control robotics system, **Reacher** is used with a discretized action space (Barreto et al. 2017) where there are 3 possible actions for each joint corresponding to (maximum positive torque, zero torque and maximum torque in the negative direction). The default OpenAI gym environments act as source tasks and target tasks are defined by changing the transition dynamics through environment specific parameters mentioned in Table 3. Each parameter is multiplied by dynamics factors in Table 3 for defining target tasks except for Reacher where the values indicate the absolute magnitude of the force along the y-axis which is absent in the source task (y-axis force in source task is 0). We study two transfer learning scenarios for each environment and the parameter used for each of the scenarios is provided in a comma separated manner in Table 3

C Algorithm Details

Delta as Criteria for Policy Reuse

For the discrete state space environments of Windy Gridworld and Supply Chain Management, Q-Learning was used as the base RL algorithm. The hyperparameters for Q Learning are shown in Table 4 where t is a counter for episodes. For probabilistic policy reuse (Fernández and Veloso 2006), the probability of selecting the source optimal policy, ψ was decayed every episode till it reached 0. This was achieved based on the update rule $\psi \leftarrow \psi\nu$ where ν was set to be the same value as the discount factor γ after sweeping over values of $\nu \in [0.99, 0.9, 0.8, 0.7]$.The additional hyperparameters that are required to use δ as a criteria for direct transfer are mentioned in Table 5. The source task is solved till convergence based on same values of hyperparameters

Table 2: Wind Magnitudes in the upward direction for each column for the different target tasks

Transfer Scenario	Wind Vector
Small changes in wind	[0, 0, -1, 1, 1, 1, 2, 2, 1, 0, 0, 0, 0, 1, 1, 1, 2, 2, 1, 0]
Wind Direction Reversed	[0, 0, 0, -1, -1, -1, -2, -2, -1, 0, 0, 0, 0, -1, -1, -1, -2, -2, -1, 0]
Additional Transfer Scenario 1	[0, 1, 1, 1, 2, 1, -1, 1, -1, 0, 0, 1, 1, 1, 2, 1, -1, 1, -1, 0]
Additional Transfer Scenario 2	[0, 1, -1, 1, 1, 1, -2, 2, 1, 0, 0, 1, -1, 1, 1, 1, -2, 2, 1, 0]

Table 3: Environment Modifications to Design Transfer Learning Scenarios

Environment	Parameter	Target Dynamics factor
CartPole	Gravity	5.0, 7.0
Acrobot	Link Mass	1.5, 2.0
LunarLander	Lander Mass	1.1, 1.5
Reacher	Y-Axis Force	0.75, 1

mentioned in Table 4. The learning rates was swept over $\alpha \in [0.5, 0.1, 0.05, 0.01]$ and the results are reported for the best hyperparameter from source task, averaged over 5 different random seeds, based on area under the learning curve. The tuned values of learning rate work well across different transfer learning scenarios.

Delta as Advice Control Factor

For tasks like CartPole, Acrobot and LunarLander, Deep Q Networks (DQN) was used as the base Reinforcement Learning algorithm to allow explicit shaping to directly affect the action selection mechanism. The DQN algorithm was implemented using PyTorch and the weights for the neural networks were initialized using Xavier initialization provided by PyTorch. The Adam optimizer (Kingma and Ba 2014) was used to train all neural networks. Exploration is performed using ϵ-greedy mechanism where ϵ is decay according to an exponential schedule. The hyperparameters used for DQN are shown in Table 6. The additional hyperparameters that are required to use δ as advice control factor are mentioned in Table 7. The source task is assumed to be solved till convergence based on same values of hyperparameters mentioned in Table 6. The learning rates for state-action values were swept over $\alpha \in [0.0001, 0.0005, 0.005, 0.001]$ and the results are reported for the best hyperparameter found in the source task, averaged over 5 different random seeds, based on area under the evaluation curve found using Tune package from Ray (Liaw et al. 2018). The algorithm is evaluated using the greedy policy after every 100 episodes of learning by averaging over performance from 50 episodes. The tuned values of learning rate work well across different transfer learning scenarios. The decay factor κ_t is updated for every training step based on the decay constant C. The decay constant value C is set to be such that the advice is utilized for 25 percent of the total timesteps and it was the best from among the choices of 25 percent, 50 percent and 100 percent based on area under the evaluation curve. An example of the variation of performance as C varies is provided in Figure 4 based on results from Cartpole where the gravity in the target task is 5 times that of the source task. Here, the different bars correspond to decay constants C such that the bias from advice is used for 25%, 50% and 100% of total timesteps during training. It can be seen that average performance of delta-based explicit shaping is similar across different values of C while the explicit shaping algorithm is relatively more sensitive to C and has higher variances across different runs. It must be noted that setting $C = 0$ such that $\kappa_t = 1$ throughout learning will affect the policy-invariance property.

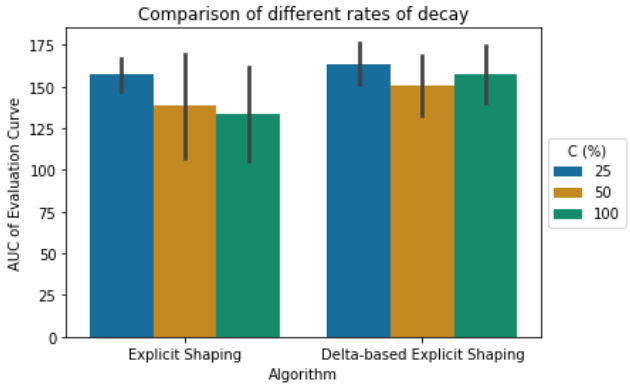


Figure 4: Comparison of different rates of advice decay constant C with error bars representing standard deviation of the results for 5 different trials.

Table 4: Hyperparameters for Q Learning

Parameters	**Windy Gridworld**	**Supply Chain Management**
Number of Episodes	5000	1000
Discount Factor	0.99	0.99
Learning Rate α	0.5	0.5
Initial Exploration Probability ϵ_0	0.1	0.1
Exploration Schedule ϵ_t	$\max(\epsilon_{t-1} - 5 \times 10^{-4}, 0)$	$\max(\epsilon_{t-1} - 5 \times 10^{-4}, 0)$

Table 5: Additional Hyperparameters for using Delta as Criteria for Policy Reuse

Parameters	**Windy Gridworld**	**Supply Chain Management**
Learning Rate α	0.5	0.5
Threshold value Δ	10	5
Delta Learning Rate α_d	0.05	0.1

Table 6: Hyperparameters for DQN

Parameters	CartPole	Acrobot	LunarLander	Reacher
Number of Episodes	4000	2000	3000	3000
Discount Factor	0.99	0.99	0.99	0.99
Number of Hidden Layers	2	2	2	2
Layer Size	64	256	256	256
Hidden Layer Activation	ReLU	ReLU	ReLU	ReLU
Maximum Value for Gradient Clipping	10	10	10	10
Target Update Frequency (in terms of steps)	2000	1000	1000	1000
Replay Buffer Capacity	10000	10000	10000	10000
Batch Size	64	128	128	128
Learning Rate	0.0001	0.0001	0.0001	0.0001
Initial Exploration Probability ϵ_0	1	1	1	1
Final Exploration Probability ϵ_T	0.01	0.01	0.01	0.01
Exploration Probability Decay ϵ_d	500	500	500	500

Table 7: Additional Hyperparameters when using DQN with Explicit Shaping

Parameters	CartPole	Acrobot	LunarLander	Reacher
Number of Hidden Layers for δ	2	2	2	2
Layer Size for δ	64	256	256	256
Hidden Layer Activation for δ	ReLU	ReLU	ReLU	ReLU
Maximum Value for Gradient Clipping for δ	10	10	10	10
Learning Rate for δ	0.0005	0.0005	0.0005	0.0005