Contextual Policy Transfer in Reinforcement Learning Domains via Deep Mixtures-of-Experts

Michael Gimelfarb* **Scott Sanner*** **Chi-Guhn Lee**

Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada

Abstract

In reinforcement learning, agents that consider the context or current state when transferring source policies have been shown to outperform context-free approaches. However, existing approaches suffer from limitations, including sensitivity to sparse or delayed rewards and estimation errors in values. One important insight is that explicit learned models of the source dynamics, when available, could benefit contextual transfer in such settings. In this paper, we assume a family of tasks with shared sub-goals but different dynamics, and availability of estimated dynamics and policies for source tasks. To deal with possible estimation errors in dynamics, we introduce a novel Bayesian mixture-of-experts for learning state-dependent beliefs over source task dynamics that match the target dynamics using state transitions collected from the target task. The mixture is easy to interpret, is robust to estimation errors in dynamics, and is compatible with most RL algorithms. We incorporate it into standard policy reuse frameworks and demonstrate its effectiveness on benchmarks from OpenAI gym.

1 INTRODUCTION

Reinforcement learning (RL) is a general framework for developing artificial agents that learn to make complex decisions by interacting with an environment. In recent years, RL algorithms have achieved state-of-the-art performance on simulated tasks such as Atari games [Mnih et al., 2015] and real-world applications [Gu et al., 2017]. However, model-free RL algorithms are sensitive to the choice of reward or hyper-parameters [Henderson et al., 2018, Seo et al., 2019], and are often not sample-efficient [Yarats et al., 2019].

To address this concern, *transfer learning* reduces the number of samples required to learn a new (target) task by reusing previously acquired knowledge from other similar (source) tasks [Lazaric, 2012, Taylor and Stone, 2009]. Many papers in this area focus on reusing policies, because it is intuitive and direct, and does not rely on value functions that can be difficult to estimate [Van Hasselt et al., 2016]. Furthermore, transferring policies from multiple sources can be more effective than a single policy [Fernández and Veloso, 2006, Rosman et al., 2016]. However, to make such transfer successful in practice, a learning agent should be able to identify which source policies are relevant in each state of the target environment, referred to as *contextual transfer* [Taylor and Stone, 2009]. In recent years, various frameworks have been proposed to tackle this problem. The *hierarchical RL* framework is naturally well-suited because it decomposes complex goals into simpler sub-goals – each requiring distinct yet complementary skills to be learned – which must then be combined to solve the original problem. Perhaps the best-known approach in this category is the *options* framework, which models the choice of source policy as a temporally-extended action, embedding it within MDPs [Sutton et al., 1999]. However, such transfer methods typically assign credit based on observed rewards, which could be sparse or significantly delayed, and can therefore be sensitive to estimation errors in value functions.

On the other hand, learned estimates of the source dynamics, when available, could offer a strong contextual indicator of which source policies to use, without suffering from the aforementioned problems. Such estimates have been used in many relevant areas, such as transferring from simulators to the real world [Tan et al., 2018] or fine-tuning policies obtained using model-based learners [Nagabandi et al., 2018]. They are also routinely available in many industrial and practical settings. For example, in asset maintenance, practitioners often rely on a digital reconstruction of a machine and its operating environment, called a *digital twin*, to learn a maintenance policy [Aivaliotis et al., 2019]. Here, source tasks could represent models of optimal control under a wide

*Affiliate to Vector Institute, Toronto, Canada.

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

range of conditions, such as different climate and weather forecasts, or hypothetical usage and degradation patterns associated with the asset. They are also prevalent in other fields, such as drug discovery [Durrant and McCammon, 2011], robotics [Christiano et al., 2016] or manufacturing [Shao and Helu, 2020]. Furthermore, in order to achieve robustness in these settings, one can simulate high-risk scenarios that do not occur often in practice and contextually transfer this information to the real-world target learner to enhance its safety during unexpected events.

Starting with this insight, we consider a family of tasks with shared *sub-goals* [McGovern and Barto, 2001] but different dynamics [Eysenbach et al., 2021, Tirinzoni et al., 2019]. Furthermore, both the dynamics and control policies of the source tasks are estimated prior to transfer, although we do *not* require that the source dynamics are precisely learned, as demonstrated in our experiments. To enable contextual policy transfer, we introduce a novel Bayesian framework for autonomously identifying and combining promising subregions from multiple source tasks. This is done by placing state-dependent Dirichlet priors over source task dynamics, and learning the corresponding posterior distributions using state trajectories sampled from the target task. Furthermore, explicit knowledge of the target dynamics is not necessary, making our approach model-free with respect to the target task, which is a critical assumption when the target task has sparse data. Finally, naive tabulation of state-dependent priors is intractable in large or continuous state spaces, so we parameterize it as a deep neural network. Interpreted as a contextual *mixture-of-experts* (MoE) [Jacobs et al., 1991], it serves as a surrogate model for informing the state-dependent contextual selection of source policies for locally exploring promising actions in each state.

Our approach has several key advantages over other existing methods. First, Bayesian inference methods can effectively "smooth" the noise in the training data, allowing source tasks to be selected robustly even in the presence of noise in the estimates of the source dynamics [Gimelfarb et al., 2020]. Also, our method only requires the source dynamics to be used for measuring task similarity, and can thus be applied in typical model-free settings. Coupled with the Bayesian formulation, this makes our approach robust to estimation errors in source dynamics as we demonstrate empirically. Second, the mixture model is amenable to function approximation, and can therefore benefit from advances in deep function approximation [Krizhevsky et al., 2012], while still being Bayesian. Third, our approach separates reasoning about task similarity from policy learning, making it agnostic to the reinforcement learning algorithm and the type of knowledge transferred. This facilitates transfer of value functions, source dynamics, and other knowledge representations between tasks, though we only focus on policy transfer in this work. Finally, the learned posterior is easy to interpret, as we demonstrate empirically (Figures 4 and 5).

The main contributions of this paper are threefold:

1. We introduce a contextual mixture-of-experts model to efficiently learn state-dependent posterior distributions over source task models for discrete (Section 3.1) and continuous MDPs (Section 3.2);
2. We show how the trained mixture model can be incorporated into existing transfer learning frameworks, namely policy reuse for improving exploration (Section 3.3) and reward shaping for dealing with sparse rewards (Section 3.4);
3. We demonstrate the effectiveness and generality of our approach by testing it on problems with discrete and continuous spaces, including physics simulations (Section 4).

Overall, our paper is the first to leverage learned source dynamics to contextually transfer policies, and to demonstrate the ability for a target policy to adapt stably even in the presence of imperfect estimates of the dynamics.

2 PRELIMINARIES

Markov Decision Process We follow the framework of *Markov decision processes* (MDPs) [Puterman, 2014], defined as five-tuples $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma \rangle$: $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $P(s'|s,a)$ are the state dynamics, $R(s,a,s')$ is a bounded reward function, and $\gamma \in [0,1]$ is a discount factor. In deterministic problems, the state dynamics are typically represented as a deterministic map $s' = f(s,a)$. The objective of an agent is to find an optimal deterministic *policy* $\pi^* : \mathcal{S} \to \mathcal{A}$ that maximizes the discounted cumulative reward, defined as $Q^\pi(s,a) = \mathbb{E}_{s_t \sim P,\, a_t \sim \pi} \left[\sum_{t=0}^{T} \gamma^t r_t \,|\, s_0 = s, a_0 = a \right]$, starting from an initial state-action pair (s,a), where $r_t = R(s_t, a_t, s_{t+1})$.

Reinforcement Learning In the reinforcement learning setting, neither P nor R are assumed to be known by the agent. Instead, the agent collects data $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots s_T)$ by interacting with the environment through a randomized exploration policy. In *model-based RL* (MBRL), the agent uses this data to first estimate P and R, and then uses these estimates to learn the optimal policy π^*. In *model-free RL*, an agent learns the optimal policy π^* directly without estimating P nor R [Sutton and Barto, 2018].

Learning Dynamics In model-based RL, a model of the state dynamics is defined and trained through repeated interactions with the environment. It returns an estimate of the next state directly $\hat{s}' = \hat{f}(s,a)$ in deterministic MDPs, or approximates its distribution in stochastic MDPs, for instance using a parameteric model such as a normal distribution $\hat{s}' \sim \mathcal{N}(\mu(s,a), \mathbf{\Sigma}(s,a))$ [Levine and Abbeel, 2014] or a

non-parameteric model such as a Gaussian process [Deisenroth and Rasmussen, 2011]. Subsequently, samples from the trained dynamics model can be used to augment the real experience when training the policy [Kaiser et al., 2020]. In this paper, we use dynamics to implement efficient and robust transfer of policies between tasks.

Transfer Learning We are interested in solving the following transfer learning problem. A library of $n \geq 1$ source tasks and a single target task are provided, with common $\mathcal{S}$ and $\mathcal{A}$, common sub-goals, but different dynamics. For each source task $i = 1, 2 \dots n$, the control policy π_i^* and the underlying dynamics $\hat{P}_i(s'|s,a)$ or $\hat{f}_i(s,a)$ are estimated from data. More generally, it is possible to transfer sample data, value functions, or other sources of domain knowledge in our framework, but we only study policy transfer in this paper. The main objective is to make use of this knowledge to solve the new target task in an efficient online manner.

3 CONTEXTUAL POLICY TRANSFER

In many domains, the state dynamics of a target task may be locally similar to one source task in one region of the state space, but a different source task in another region. By reasoning about task similarity *locally* in different regions of the state space, an RL agent can make more efficient use of source task knowledge. In this section, we proceed to model and learn state-dependent contextual similarity between source tasks and a target task. We also derive a theoretical result to justify our use of source dynamics.

An overview of our framework applied with DQN [Mnih et al., 2015] is illustrated in Figure 1, which will be described in detail in the following subsections. The main components of our framework include: the Q-value function, parameterized as ϑ with loss $\mathcal{L}_{DQN}$; the mixture network parameterized as θ with loss function $\mathcal{L}$, that learns the similarity $\mathbf{b}$ between source and target tasks in each state s; the environment Env. that is interacted with using a behavior policy π^b, derived from Q and reshaped by the source policies and the predictions of the mixture network; and a buffer of previous data $\mathcal{D}$ that trains both the DQN and the mixture network in an offline batched manner.

3.1 DEEP CONTEXTUAL MIXTURE-OF-EXPERTS

In order to develop a Bayesian framework for measuring task similarity, we first introduce a state-dependent prior $\mathbb{P}(\mathbf{w}|s)$ over combinations $\mathbf{w}$ of source task models. This prior will be updated to a posterior $\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)$ that tries to match the true (unknown) target dynamics, using transitions $\mathcal{D}_t = \{(s_\tau, a_\tau, s_{\tau+1}), \tau = 1, 2 \dots t\}$ collected from the target environment up to each time instant t. Here, $\mathbf{w} \in \mathbb{R}^n$ consists of non-negative elements such that $\sum_{i=1}^n w_i = 1$.

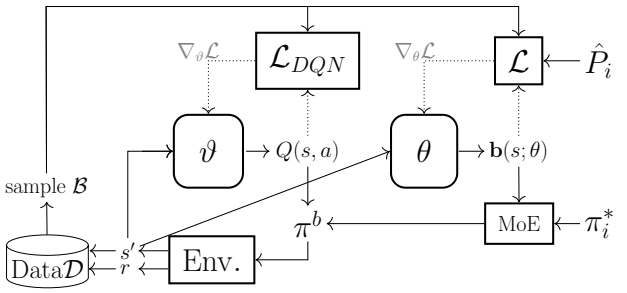


Figure 1: A conceptual illustration of the proposed framework with a source-policy guided behavior policy π^b for exploration (Algorithm 2). Here, the agent's Q-values are modeled using the DQN architecture [Mnih et al., 2015], with parameters ϑ and standard TD-loss $\mathcal{L}_{DQN}$ used to train Q-values offline on batches of samples from a replay buffer. Please note that the learning of the contextual mixture $\mathbf{b}(\cdot;\theta)$ is independent of the Q-values, so the DQN architecture could be replaced by other reinforcement learning architectures, including those that do not learn Q-values.

Using combinations to model uncertainty in source task selection in this way can be viewed as *Bayesian model combination*. This is much more general than Bayesian model averaging techniques, since $\mathbf{w}$ can converge to a mixture over source dynamics, rather than collapse to a single source task. This has been shown to provide several advantages, including more stable convergence and robustness to model misspecification [Minka, 2000, Monteith et al., 2011].

In our setting, exact inference for $\mathbf{w}$ is intractable, so we model $\mathbf{w}$ using a surrogate probability distribution. Since each realization of $\mathbf{w}$ is a discrete probability distribution, a suitable conjugate prior for $\mathbf{w}$ in each state s is a *Dirichlet distribution* [Gimelfarb et al., 2018] with density

$$\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t) \propto \prod_{i=1}^n w_i^{\alpha_{t,i}(s)-1},$$

where $\alpha_{t,i} : \mathcal{S} \to \mathbb{R}$ for all $i = 1, 2 \dots n$ and $t = 0, 1, 2, \dots$ outputs a vector with strictly positive entries. By averaging out the uncertainty in $\mathbf{w}$, we can obtain a posterior estimator $\mathbb{P}(s'|s,a,\mathcal{D}_t)$ of target dynamics:

$$\begin{aligned}
\mathbb{P}(s'|s,a,\mathcal{D}_t) &= \int \mathbb{P}(s'|s,a,\mathbf{w})\,\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)\,\mathrm{d}\mathbf{w} \\
&= \int \sum_{i=1}^n \mathbb{P}(s'|s,a,\mathbf{w},i)\,\mathbb{P}(i|\mathbf{w},s)\,\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)\,\mathrm{d}\mathbf{w} \\
&= \int \sum_{i=1}^n \hat{P}_i(s'|s,a) w_i\,\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)\,\mathrm{d}\mathbf{w} \\
&= \sum_{i=1}^n \hat{P}_i(s'|s,a) \int w_i\,\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)\,\mathrm{d}\mathbf{w} \\
&= \sum_{i=1}^n \hat{P}_i(s'|s,a)\ \mathbb{E}_{\mathbf{w}\sim\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)}[w_i]. \qquad (1)
\end{aligned}$$

In the following sections, we will instead refer to the following normalized form of (1)

$$\mathbb{P}\left(s'|s,a,\mathcal{D}_t\right) = \sum_{i=1}^{n} \hat{P}_i(s'|s,a)\, b_{t,i}(s), \tag{2}$$

where $b_{t,i}(s) = \mathbb{E}_{\mathbf{w}\sim\mathbb{P}(\mathbf{w}|s,\mathcal{D}_t)}\left[w_i\right] = \frac{\alpha_{t,i}(s)}{\sum_{j=1}^{n}\alpha_{t,j}(s)}$ is seen as a contextual mixture or *belief* over source task models.

In a tabular setting, it is feasible to maintain separate estimates of $b_{t,i}(s)$ per state using approximate Bayes' inference [Andrieu et al., 2003, Gimelfarb et al., 2018]. However, maintaining such estimates for large or continuous state spaces presents inherent computational challenges. Fortunately, as (2) showed, the posterior mean $\mathbf{b}_t(s)$ is a sufficient estimator of $\mathbb{P}\left(s'|s,a,\mathcal{D}_t\right)$. Therefore, we can approximate $\mathbf{b}_t(s)$ directly using a feed-forward neural network $\mathbf{b}(s;\theta)$ with parameters θ. The input of $\mathbf{b}(s;\theta)$ is a vectorized representation of s, and the outputs $z_i^b(s;\theta)$ are fed through the $\mathrm{softmax}$ function $b_i(s;\theta) \propto \exp\left(z_i^b(s;\theta)\right)$ to guarantee that $\mathbf{b}_t(s) \geq \mathbf{0}$ and $\sum_{i=1}^{n} b_{t,i}(s) = 1$. Now it is no longer necessary to store all $\mathcal{D}_t$, since each sample can be processed online or in batches. Furthermore, since θ is a neural network approximation of (2), we can write $\mathbb{P}\left(s'|s,a,\mathcal{D}_t\right) \simeq \mathbb{P}\left(s'|s,a,\theta\right)$.

In order to learn θ, we minimize the empirical *negative log-likelihood function*[1], given by (2) as:

$$\begin{aligned}
\mathcal{L}(\theta) &= -\log \mathbb{P}_{s'\sim P_{target}(s'|s,a)}\left(\mathcal{D}_t|\theta\right) \\
&= -\log\left(\prod_{(s,a,s')\in\mathcal{D}_t}\sum_{i=1}^{n}\mathbb{P}\left(s'|s,a,i\right)\mathbb{P}\left(i|\theta\right)\right) \\
&= -\sum_{(s,a,s')\in\mathcal{D}_t}\log\left(\sum_{i=1}^{n}\hat{P}_i(s'|s,a)\, b_i(s;\theta)\right).
\end{aligned} \tag{3}$$

The gradient of $\mathcal{L}(\theta)$ has a Bayesian interpretation, as the following result shows.

Proposition 1. *For a single sample (s,a,s'):*

$$\begin{aligned}
\nabla_\theta\mathcal{L}(\theta) &= \sum_{j=1}^{n}\frac{\partial z_j^b(s;\theta)}{\partial\theta}\left(b_j(s;\theta) - p_j(s;\theta)\right) \\
p_j(s;\theta) &= \frac{\hat{P}_j(s'|s,a)\, b_j(s;\theta)}{\sum_{i=1}^{n}\hat{P}_i(s'|s,a)\, b_i(s;\theta)}.
\end{aligned} \tag{4}$$

Proof. Let $Z = \sum_{i=1}^{n}\hat{P}_i(s'|s,a)\, b_i(s;\theta)$ and apply the chain rule with $b_i(s;\theta) \propto \exp\left(z_i^b(s;\theta)\right)$:

$$\begin{aligned}
\nabla_\theta\mathcal{L}(\theta) &= -\nabla_\theta\log\left(\sum_{i=1}^{n}\hat{P}_i(s'|s,a)\, b_i(s;\theta)\right) \\
&= -\frac{1}{Z}\sum_{i=1}^{n}\hat{P}_i(s'|s,a)\,\frac{\partial b_i(s;\theta)}{\partial\theta} \\
&= -\frac{1}{Z}\sum_{i=1}^{n}\hat{P}_i(s'|s,a)\sum_{j=1}^{n}\frac{\partial b_i(s;\theta)}{\partial z_j^b}\frac{\partial z_j^b(s;\theta)}{\partial\theta} \\
&= -\frac{1}{Z}\sum_{j=1}^{n}\frac{\partial z_j^b(s;\theta)}{\partial\theta}\sum_{i=1}^{n}\hat{P}_i(s'|s,a)\frac{\partial b_i(s;\theta)}{\partial z_j^b} \\
&= \frac{-1}{Z}\sum_{j=1}^{n}\frac{\partial z_j^b(s;\theta)}{\partial\theta}\sum_{i=1}^{n}\hat{P}_i(s'|s,a)b_i(s;\theta)\left(\delta_{ij} - b_j(s;\theta)\right) \\
&= -\sum_{j=1}^{n}\frac{\partial z_j^b(s;\theta)}{\partial\theta}\sum_{i=1}^{n}p_i(s;\theta)\left(\delta_{ij} - b_j(s;\theta)\right) \\
&= \sum_{j=1}^{n}\frac{\partial z_j^b(s;\theta)}{\partial\theta}\left(b_j(s;\theta) - p_j(s;\theta)\right).
\end{aligned}$$

This completes the proof. □

Here, $b_i(s;\theta)$ can be interpreted as a prior. Once a new sample (s,a,s') is observed, the posterior distribution $p_i(s;\theta)$ is computed using Bayes' rule, and θ is updated according to the difference between prior and posterior, scaled by state features z_i^b. It is thus expected that by training θ using gradient descent on (4), e.g. by updating $\theta' = \theta - \lambda\nabla_\theta\mathcal{L}(\theta)$, the prior $b_i(s;\theta)$ will eventually be driven to its correct posterior as the number of updates is increased. Regularization of $b_i(s;\theta)$ can also be easily incorporated by using informative priors $\mathbb{P}\left(\theta\right)$ (e.g. isotropic Gaussian, Laplace) in (3), and can lead to smoother posteriors.

Unfortunately, proving the convergence of gradient descent is outside the scope of this work. However, the framework proposed above could potentially leverage recent theoretical developments concerning the convergence of deep learning models in more general settings [He and Tao, 2020]. Furthermore, we can also show that incorporating dynamics as part of the transfer learning process in a contextual way can lead to better transfer than using a single source task.

Proposition 2. *Consider an MDP $\langle\mathcal{S},\mathcal{A},P,R,\gamma\rangle$ with finite $\mathcal{S}$ and $\mathcal{A}$ and bounded reward $R:\mathcal{S}\to\mathbb{R}$. Let $\mathbf{R}$ be the reward function in vector form, $\hat{\mathbf{P}}^\pi$ be an estimate of the transition probabilities induced by a policy $\pi:\mathcal{S}\to\mathcal{A}$ in matrix form, and $\hat{\mathbf{V}}^\pi$ be the corresponding value function in vector form. Also, let $\mathbf{P}^\pi$ and $\mathbf{V}^\pi$ be the corresponding values under the true dynamics. Then for any policy π,*

$$\|\hat{\mathbf{V}}^\pi - \mathbf{V}^\pi\|_\infty \leq \frac{\gamma}{(1-\gamma)^2}\|\mathbf{R}\|_\infty\|\hat{\mathbf{P}}^\pi - \mathbf{P}^\pi\|_\infty.$$

Proof. First, observe that for any stochastic matrix $\mathbf{P}$, $\|\mathbf{P}\| = 1$, where $\|\cdot\|$ is the infinity norm, and $\mathbf{I} - \gamma\mathbf{P}$ is always invertible. Therefore:

$$\|(\mathbf{I}-\gamma\mathbf{P})^{-1}\| = \left\|\sum_{t=0}^{\infty}(\gamma\mathbf{P})^t\right\| \leq \sum_{t=0}^{\infty}\gamma^t\|\mathbf{P}\|^t = \frac{1}{1-\gamma}.$$

To simplify notation, we write $\mathbf{V}_1 = \hat{\mathbf{V}}^\pi$, $\mathbf{V}_2 = \mathbf{V}^\pi$, $\mathbf{P}_1 = \hat{\mathbf{P}}^\pi$ and $\mathbf{P}_2 = \mathbf{P}^\pi$. Then $\mathbf{V}_1 = (\mathbf{I}-\gamma\mathbf{P}_1)^{-1}\mathbf{R}$ and $\mathbf{V}_2 =$

[1] This corresponds to a uniform (improper) prior $\mathbb{P}\left(\theta\right)$.

$(\mathbf{I} - \gamma\mathbf{P}_2)^{-1}\mathbf{R}$ [Ng and Russell, 2000]. Now, making use of the identity $\mathbf{X}^{-1} - \mathbf{Y}^{-1} = \mathbf{X}^{-1}(\mathbf{Y} - \mathbf{X})\mathbf{Y}^{-1}$:

$$\begin{aligned}
\|\hat{\mathbf{V}}^\pi - \mathbf{V}^\pi\| &= \|(\mathbf{I} - \gamma\mathbf{P}_1)^{-1}\mathbf{R} - (\mathbf{I} - \gamma\mathbf{P}_2)^{-1}\mathbf{R}\| \\
&\leq \|(\mathbf{I} - \gamma\mathbf{P}_1)^{-1} - (\mathbf{I} - \gamma\mathbf{P}_2)^{-1}\|\|\mathbf{R}\| \\
&= \|(\mathbf{I} - \gamma\mathbf{P}_1)^{-1}\gamma(\mathbf{P}_2 - \mathbf{P}_1)(\mathbf{I} - \gamma\mathbf{P}_2)^{-1}\|\|\mathbf{R}\| \\
&\leq \gamma\|(\mathbf{I} - \gamma\mathbf{P}_1)^{-1}\|\|(\mathbf{I} - \gamma\mathbf{P}_2)^{-1}\|\|\mathbf{P}_2 - \mathbf{P}_1\|\|\mathbf{R}\| \\
&\leq \gamma\left(\frac{1}{1-\gamma}\right)^2\|\mathbf{P}_2 - \mathbf{P}_1\|\|\mathbf{R}\|,
\end{aligned}$$

and so the proof is complete. □

This result justifies our methodology of using source task dynamics similarity to guide state-dependent policy reuse, since our choice of contextual $\hat{\mathbf{P}}$ (equation (2)) for sampling policies would generally result in lower error $\|\hat{\mathbf{P}}^\pi - \mathbf{P}^\pi\|_\infty$, and thus better values, than that obtained by committing to a single source task, e.g. $\mathbf{P}_i^\pi$. This claim will be validated empirically in Section 4.

3.2 CONDITIONAL RBF NETWORK

In continuous-state tasks with deterministic transitions, $P_i(s'|s,a)$ correspond to Dirac measures, in which case it is more suitable to learn a model that can predict the next state $\hat{f}_i(s,a)$ directly. In order to tractably update the mixture model in this setting, we assume that, given source task i is the correct model of target dynamics, the probability of observing a transition from state s to state s' is a decreasing function of the prediction error $\|s' - \hat{s}'\| = \|s' - \hat{f}_i(s,a)\|$. More formally, given an arbitrarily small region $S = [s', s' + \mathrm{d}s']$,

$$\mathbb{P}\left(s' \in S \mid s, a, \mathbf{w}, i\right) = \rho_i\left(\|s' - \hat{f}_i(s,a)\|\right)\,\mathrm{d}s', \quad (5)$$

where $\rho_i : \mathbb{R} \to \mathbb{R}$ can be interpreted as a normalized[2] *radial basis function* (RBF). A popular choice of ρ_i, and implemented in this paper, is the *Gaussian kernel*, which for $\nu_i > 0$ is given as $\rho_i(r) \propto \exp\left(-\nu_i r^2\right)$. In principle, ν_i could be modeled as an additional output of the mixture model, $\nu_i(\theta)$, and learned from data [Bishop, 1994], although we treat it as a constant in this paper.

By using (5) and following the derivations leading to (1), we obtain the following result in direct analogy to (2)

$$\mathbb{P}\left(s' \in S \mid s, a, \mathcal{D}_t\right) = \sum_{i=1}^{n} \rho_i\left(\|s' - \hat{f}_i(s,a)\|\right) b_{t,i}(s)\,\mathrm{d}s'. \quad (6)$$

Consequently, the results derived in the previous sections, including the mixture model and loss function for $\mathbf{b}(s;\theta)$, hold by replacing $\hat{P}_i(s'|s,a)$ with $\rho_i\left(\|s' - \hat{f}_i(s,a)\|\right)$. Furthermore, since (6) approximates the target dynamics as a mixture of kernel functions, it can be viewed as a conditional analogue of the *radial basis function network* [Broomhead and Lowe, 1988]. It remains to show how to make use of this model and the source policies in new target tasks.

[2]Technically, we could only require that $\rho_i \geq 0$, as the likelihood need not be a valid probability density.

3.3 POLICY REUSE FOR EXPLORATION

The most straightforward approach for policy reuse is to sample a source policy π_t at random at each time step t according to $\mathbf{b}(s_t;\theta)$, and apply action $a = \pi_t(s_t)$ in the current state s_t. To allow for exploration, the agent only takes advice with probability $p_t \in [0,1]$, otherwise following a target exploration policy [Fernández and Veloso, 2006, Li and Zhang, 2018]. This resulting behaviour policy is suitable for any off-policy RL algorithm. We call this ***M**odel-**A**ware **P**olicy Reu**s**e for **E**xploration* (MAPSE), and present the corresponding pseudocode for sampling the modified behavior policy π^b in Algorithm 1, and the overall training loop for MAPSE in Algorithm 2.

Algorithm 1 `PolicyReuse`(s_t)

Require: s_t, $\mathbf{b}(s_t;\theta)$, p_t, $\Pi = \{\pi_1^*, \dots \pi_n^*\}$, π
  sample_source_policy $\sim$ Bernoulli(p_t)
  if sample_source_policy $=$ true **then**
    $i_t \sim$ Categorical$(\mathbf{b}(s_t;\theta))$; $a_t \leftarrow \pi_{i_t}^*(s_t)$
  else
    $a_t \sim \pi(s_t)$
  end if
  return a_t

Algorithm 2 `MAPSE`

Require: $\hat{P}_i(s'|s,a)$ (or $\hat{f}_i(s,a)$ and ρ_i) for $i = 1, 2 \dots n$, $\Pi = \{\pi_1^*, \dots \pi_n^*\}$, π, θ, λ, $\mathcal{D} = \emptyset$, p_t
  for episode $m = 1, 2, \dots$ **do**
    Sample $D \leftarrow (s_0, a'_0, r_0, s_1, a'_1, r_1, \dots)$ from the target task, where $a'_t \sim$ `PolicyReuse`(s_t)
    Store D in $\mathcal{D}$
    Sample a mini-batch $\mathcal{B} \subset \mathcal{D}$
    Update π and $\theta \leftarrow \theta - \lambda\nabla_\theta\mathcal{L}(\theta)$ on $\mathcal{B}$
  end for
  return π

However, this approach has several shortcomings. Firstly, it is not clear how to anneal p_t, since $\mathbf{b}(s;\theta)$ is learned over time and is non-stationary. Secondly, using the recommended actions too often can lead to poor test performance, since the agent may not observe sub-optimal actions enough times to generalize well to the target task. Finally, since efficient credit assignment is particularly difficult in sparse reward problems [Seo et al., 2019], the effectiveness of this behavior policy could be limited in practice.

3.4 POTENTIAL-BASED REWARD SHAPING

A more principled approach for incorporating policy advice is to reshape the original reward function using *potential-based reward shaping* (PBRS) [Ng et al., 1999, Brys et al., 2015]. More specifically, the original reward signal $R_t(s, a, s')$ at time t is modified to a new signal $R'_t(s, a, s') = R_t(s, a, s') + cF_t(s, a, s', a')$, where[3]

$$F_t(s, a, s', a') = \gamma\, \Phi_{t+1}(s', a') - \Phi_t(s, a). \tag{7}$$

Here, $c > 0$ defines the strength of the shaped reward signal, and can be tuned for each problem or empirically scaled to the magnitude of the rewards. The *potential* $\Phi_t(s, a)$ is chosen to be the posterior probability $\mathbb{P}(a|s, \mathcal{D}_t)$ that action a would be recommended by a source policy in state s at time t. By repeating the derivations leading to (1), we can derive an expression for $\Phi_t(s, a)$ in the Bayesian framework:

$$\begin{aligned}
&\Phi_t(s, a) = \mathbb{P}(a|s, \mathcal{D}_t) \\
&= \int \sum_{i=1}^{n} \mathbb{P}(a|s, \mathcal{D}_t, \mathbf{w}, i)\, \mathbb{P}(i|\mathbf{w}, s)\, \mathbb{P}(\mathbf{w}|s, \mathcal{D}_t)\, \mathrm{d}\mathbf{w} \\
&= \sum_{i=1}^{n} \mathbb{P}(a = \pi_i^*(s))\, b_i(s; \theta).
\end{aligned} \tag{8}$$

Note that (8) reduces to Brys et al. [2015] when $n = 1$ and source policies are deterministic. Unlike MAPSE, this approach can also be applied on-policy, and preserves policy optimality [Devlin and Kudenko, 2012]. It can also be applied for problems with continuous action spaces, by modelling $\mathbb{P}(a = \pi_i^*(s))$ using RBFs as discussed earlier. We call this approach ***M**odel-**A**ware **R**eward **S**haping* (MARS), and present the training loop in Algorithm 3.

Algorithm 3 `MARS`

Require: $\hat{P}_i(s'|s, a)$ (or $\hat{f}_i(s, a)$ and ρ_i) for $i = 1, 2 \ldots n$, $\Pi = \{\pi_1^*, \ldots \pi_n^*\}$, π, θ, λ, $\mathcal{D} = \emptyset$, c
  for episode $m = 1, 2, \ldots$ **do**
    Sample $D \leftarrow (s_0, a_0, r'_0, s_1, a_1, r'_1, \ldots)$ from the target task, where $a_t \sim \pi(s_t)$ and r'_t is computed according to (7) and (8)
    Store D in $\mathcal{D}$
    Sample a mini-batch $\mathcal{B} \subset \mathcal{D}$
    Update π and $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta)$ on $\mathcal{B}$
  end for
  return π

[3] Our implementation used look-back shaping rather than the look-ahead form presented here, in which s, a correspond to the state-action pair at time $t - 1$ and s', a' correspond to time t rather than t and $t + 1$. While policy invariance is not necessarily guaranteed for look-back shaping in off-policy learning [Wiewiora et al., 2003], we found it to work well and easy to implement.

Remarks The proposed framework is general and modular and can be combined with most RL algorithms and even model-based RL. Furthermore, the computational cost of processing each sample is a linear function of the cost of evaluating the source task dynamics and policies. Many extensions to this framework are also possible. For instance, to reduce the effect of negative transfer, it is possible to estimate the target task dynamics $\hat{P}_{target}(s'|s, a)$ or $\hat{f}_{target}(s, a)$, and include it as an additional $(n + 1)$-st component in the mixture (2). If this model can be estimated accurately, it can also be used to update the agent directly. Further improvements for MARS could be obtained by learning a secondary Q-value function for the potentials [Harutyunyan et al., 2015]. We do not investigate these extensions in this paper, which can form interesting topics for future study.

4 EMPIRICAL EVALUATION

We evaluate the empirical performance of MAPSE and MARS in a typical reinforcement learning setting (see Appendix for details).

Research Questions In particular, our aim is to answer the following questions:

1. Does $\mathbf{b}(s; \theta)$ learn to select the most relevant source task(s) in each state?
2. Does MARS (and possibly MAPSE) achieve better test performance, relative to the number of transitions observed, over existing baselines?
3. How interpretable is the learned task similarity, as measured by $\mathbf{b}(s; \theta)$?

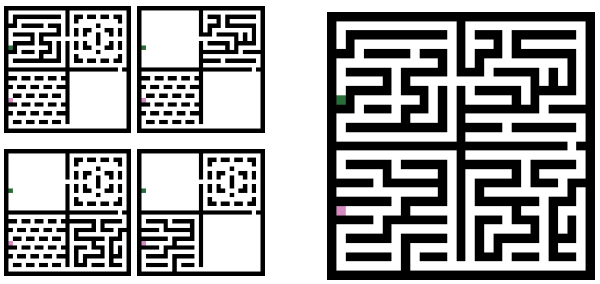
Baseline Algorithms In order to answer these questions, we consider tabular **Q-learning** [Watkins and Dayan, 1992] and **DQN** [Mnih et al., 2015] with MAPSE and MARS, in which source models and policies are learned separately. Please note that our approach could also be applied with model-based approaches. To ensure fair comparison with relevant baselines, we include one recently published context-free and one contextual policy reuse algorithm:

1. **CAPS:** a contextual option-based algorithm for hierarchical transfer as proposed in Li et al. [2019];
2. **UCB:** a multi-armed bandit algorithm for policy selection as proposed in Li and Zhang [2018];
3. **Φ1, Φ2. . . :** reward shaping for transferring each source policy i individually as proposed in Brys et al. [2015];
4. **Q:** Q-learning [Watkins and Dayan, 1992] and DQN [Mnih et al., 2015] trained without transfer.

Benchmark Domains To help us compare these baselines, we consider three variants of existing problems, **Transfer-Maze**, **Transfer-CartPole**, and **Transfer-SparseLunarLander**, that are explained in the subsequent subsections. These domains are designed to demonstrate the effectiveness of MAPSE/MARS in discrete and continuous state spaces and a realistic problem with sparse rewards and complex physics. The quality of the estimated models also varies considerably between them, and serves to assess the stability of $\mathbf{b}(s;\theta)$ in the face of imperfect $\hat{P}_i$.

4.1 TRANSFER-MAZE

This domain is a two-dimensional region consisting of cells arranged in a 30-by-30 grid, and divided into four rooms as shown in Figure 3. Obstacles are placed to form the rooms and also scattered throughout to make the task challenging and assess the ability of $\mathbf{b}(s)$ to match local dynamics. The four possible actions $\{$`left, up, right, down`$\}$ move the agent to the adjacent cell in the corresponding direction, but have no effect if the destination is an obstacle or boundary. The agent incurs a penalty of 0.02 for hitting a wall, and otherwise incurs a penalty of 0.01. The goal is to reach a fixed goal cell in the least number of steps, at which point the agent receives a reward of 1. The source tasks are designed to correctly model the interior of only one of the four rooms, so that only a context-aware algorithm could learn to utilize the source task knowledge correctly.



(a) Source tasks. (b) Target tasks.

Figure 3: The Transfer-Maze domain. The agent always begins in a fixed cell (green). The goal of the agent is to navigate to the fixed target cell (red).

We use Q-learning to learn optimal source and target policies, while the dynamics $\hat{f}_i(s,a)$ are trained using lookup tables. The agent's policy is periodically tested on separate episodes and the number of steps taken to reach the goal is recorded. The averaged performance is then reported in Figure 2a. Note that we have omitted the plot for $\Phi 1, \Phi 2 \ldots$ since convergence could not be obtained after 200,000 steps using only single policies. Figure 4 plots the belief $\mathbf{b}(s;\theta)$ learned and assigned to each of the source tasks over time.

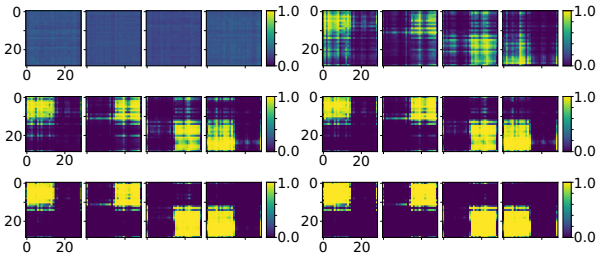
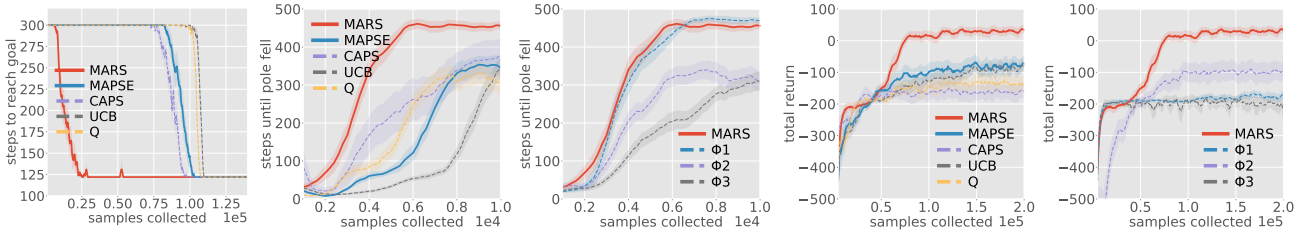


Figure 4: Each group of 4 plots shows the learned belief $\mathbf{b}_t(s;\theta)$ assigned to each source task for the Transfer-Maze experiment, after training on (respectively, left to right, top to bottom) $t = 0$, 5K, 10K, 20K, 50K and 100K target samples. The color indicates the probability, while x- and y-axes represent the x and y position of the agent.

4.2 TRANSFER-CARTPOLE

The CartPole control problem involves balancing a pole upright on top of a moving cart [Brockman et al., 2016]. The state $(x, \dot{x}, \theta, \dot{\theta})$ consists of the position and velocity of the cart, and the angle and angular velocity of the pole. The two actions correspond to applying an external force to the left or right side of the cart. To make the problem more difficult, we allow the external force $F(x)$ applied to the cart to vary with position, $F(x) = 35\sqrt{\frac{1+36}{1+36\cos^2(5x)}}\cos(5x)+40$. One way to interpret this is that the surface is not friction-less, but contains slippery (force of 75) and rough (force of 5) patches. To learn better policies, the agent can apply half or full force to the cart in either direction (4 possible actions). As a result, the correct source policy to transfer in each state depends on the surface. The problem is made even more difficult by uniformly initializing $x \in [-1.5, +1.5]$, to require the agent to generalize to both surfaces. In the first two source tasks, the agent balances the pole only on rough and slippery surfaces, respectively. For the third source task, the pole length is doubled and $F(x)$ is set uniformly to 20.

Following Mnih et al. [2015], the Q-values are approximated using a feed-forward neural network, and randomized experience replay and target networks are employed for stability. State dynamics $\hat{f}_i(s,a)$ are also parameterized as feed-forward neural networks $f_{\phi_i}(s,a)$ with parameters ϕ_i and trained using the MSE loss $\frac{1}{|\mathcal{B}|}\sum_{(s,a,s')\in\mathcal{B}}\|s'-f_{\phi_i}(s,a)\|^2$ on batches $\mathcal{B}$ drawn at random from the buffer. To learn $\mathbf{b}(s;\theta)$, we use the Gaussian kernel with fixed ν. For CAPS, we follow Li et al. [2019] and only train the last layer when learning the options; the learning rate for options is chosen from $\{10^{-2}, 10^{-3}, 10^{-4}\}$ that led to the best performance on the task. The test performance is defined as the number of steps that the agent can balance the pole in each episode before tipping over during testing. The average performance of all algorithms is illustrated in Figure 2b. Figure 5 plots the state-dependent belief $\mathbf{b}(s;\theta)$ learned and assigned to each source task over time as training progresses.



(a) Transfer-Maze (b) Transfer-CartPole (c) Tr.-SparseLunarLander

Figure 2: Test performance using the greedy policy of the target agent, as a function of the number of target samples: (a) number of steps to reach goal (b) number of steps balanced (c) cumulative reward. Curves are averaged over 20 trials for Transfer-Maze and Transfer-CartPole and 10 trials for Transfer-SparseLunarLander.

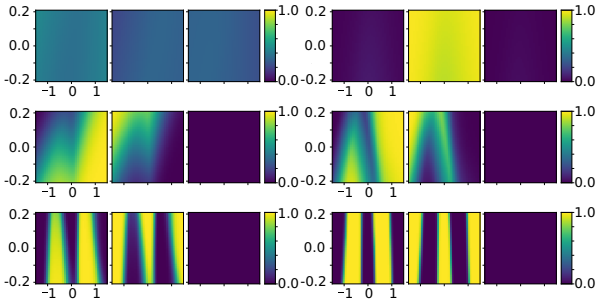


Figure 5: Each group of 3 plots shows the learned $\mathbf{b}_t(s, \theta)$ assigned to each source task for Transfer-CartPole, after training on (left to right, top to bottom) $t = 0$, 100, 500, 1K, 2.5K and 5K target samples. The color indicates the probability, while the x- and y-axes represent respectively the cart position (x) and pole angle (θ). Values are averaged over the other two state components over $[-0.5, +0.5]$.

4.3 TRANSFER-SPARSE LUNAR LANDER

This problem involves landing a spacecraft safely on a lunar surface [Brockman et al., 2016]. The state is 8-dimensional, consisting of position, orientation, velocities, angular velocities of the craft, and whether it has come into contact with the ground. The task is made much more challenging by deferring all rewards until the end of the episode, making this a sparse-reward problem. We were unable to learn accurate dynamics in this setting nor learn a suitable model-based controller, due to the complexity of the problem. In particular, the lunar surface terrain is randomly generated in each episode, making it difficult to predict ground contact. The first source task teaches the lander to hover above the landing pad at a fixed region in space ($x \in [-0.1, +0.1]$, $y \in [0.3, 0.5]$), and fails if the lander gets too close to the ground. The second source task places the lander at a random location ($x \in [-0.5, +0.5]$, $y = 0.4$) above the landing pad, and the agent learns to land the craft safely. The third source task is equivalent to the original LunarLander-v2, except the mass of the craft is reduced to 10% of the original. A successful transfer learning experiment, therefore, should learn to transfer skills from the hover and land source tasks depending on altitude, and avoid the risky policy for landing the lighter craft.

To solve this problem, we use a similar setup as Transfer-CartPole. State components are clipped to $[-1, +1]$, $\tanh$ output activations are used to predict the position and velocity components of the dynamics, and $\mathrm{sigmoid}$ activations are used to predict binary variables for leg-ground contact. Furthermore, source dynamics are learned offline on data collected during policy training to avoid the moving target problem and improve stability. We obtained an MSE of order 10^{-3} for the land and low-mass source task dynamics, highlighting the difficulty of learning accurate dynamics for ground contact. The test performance is defined as the total reward accumulated on each test episode and shown in Figure 2c. Figure 6 illustrates the learned belief $\mathbf{b}(s; \theta)$ on state trajectories obtained during training.

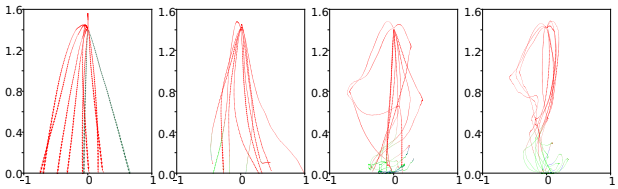


Figure 6: Lander's position on 10 training episodes, collected after $t = 0$, 10K, 25K and 50K target samples (left to right). Colors indicate source tasks assigned the largest beliefs in $\mathbf{b}_t(s; \theta)$ in each state s of the trajectories, where red corresponds to Hover, green to Land and blue to Low-Mass.

4.4 DISCUSSION

In all three experiments, we can see that MARS consistently outperforms all other baselines including MAPSE, in terms of sample efficiency and solution quality. Also, MAPSE consistently outperforms UCB, as shown in Figure 2. Figure 6 provide one possible explanation for this, namely the ability of the mixture model to converge to a correct representation even when presented with imperfect source dynamics as in Transfer-SparseLunarLander. Fur-

thermore, on all three domains, MARS achieves asymptotic performance comparable to, or better, than the best single potential function $\Phi 1$, $\Phi 2$ etc. This reaffirms our hypothesis that reward shaping can improve generalization on test data with little tuning. Furthermore, we conjecture that the inconsistent performance of CAPS is due to its reliance on fluctuating Q-values, that can be especially difficult to estimate in sparse-reward settings, as evidenced in Figure 2c. This is mitigated in MARS and MAPSE by their reliance instead on more stable samples of the dynamics, that can be learned offline prior to target task training. On the other hand, UCB does not perform well, converging asymptotically to a single source policy uniformly across the state space and demonstrating the need for contextual transfer.

Imperfect Transfer MAPSE and MARS can be most effective when the target task naturally breaks up into simpler subtasks as in Transfer-Maze. However, in states s where *none* of the source policies is relevant for transfer, there isn't necessary an optimal $\mathbf{b}(s;\theta)$. To demonstrate the potential limitations of MAPSE/MARS in such settings, we rerun the Transfer-Maze experiment but omit one of the 4 source policies. The training then proceeds as described previously 4 times (once for each possible omitted source policy), and the averaged performance is illustrated in Figure 7.

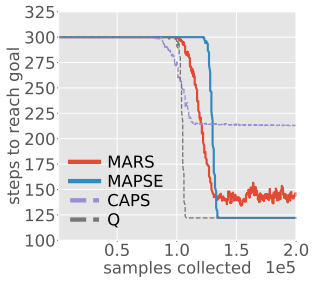


Figure 7: Average number of steps to reach the goal, after one source policy is omitted in Transfer-Maze.

Interestingly, all baselines perform worse than standard Q-learning. Furthermore, unlike the perfect transfer setting, MAPSE now learns a better policy than MARS. One possible explanation for this is that RL is highly sensitive to uninformative rewards now arising from (7), whereas MAPSE anneals p_m quickly to zero and eventually learns the right policy without relying on bad advice. The idea of annealing could potentially be applied to anneal the c hyper-parameter for MARS in a similar manner. Finally, CAPS failed to find a meaningful policy in 2 out of the 4 cases, while performing well on the remaining 2. Once again, the reliance of MAPSE and MARS on stable samples of the dynamics could provide some degree of robustness against negative transfer, which could be improved further by learning a model of target dynamics as discussed at the end of Section 3.4.

5 RELATED WORK

State-dependent contextual transfer with multiple source policies is an emerging topic in transfer learning. Different approaches for solving this problem have been proposed in different settings, including soft attention [Rajendran et al., 2017], bi-level optimization [Li and Kudenko, 2018], and options [Li et al., 2019]. Bayesian approaches include the *AC-Teach* framework of Kurenkov et al. [2019], which used Bayesian DDPG to learn probability distributions over Q-values corresponding to student and teacher actions. However, this approach was specific to DDPG and continuous control problems. Our paper complements existing work by using source task dynamics rather than value functions to reason about task similarity, and is compatible with most model-based and model-free RL algorithms. More generally, our work is related to *hierarchical reinforcement learning* [Goyal et al., 2020, Li et al., 2019, Peng et al., 2019, Yang et al., 2020], in which policies can be decomposed and transferred as low-level primitives.

Potential-based reward shaping (PBRS) was first introduced in Ng et al. [1999] for constructing dense reward signals without changing the optimal policies. Later, Wiewiora et al. [2003] and Devlin and Kudenko [2012] extended this to action- and time-dependent shaping, respectively. More recently, Harutyunyan et al. [2015] combined these two extensions into one framework and used it to incorporate arbitrary reward functions. Brys et al. [2015] made the connection between PBRS and policy reuse, by turning a single source policy into a binary reward signal and applying Harutyunyan et al. [2015]. Later, Suay et al. [2016] recovered a potential function from demonstrations directly using inverse RL. Our paper extends Brys et al. [2015] by reusing *multiple* source policies in a *contextual* way compatible with modern deep RL. Thus, our paper contributes to the expanding body of research on policy transfer and reward shaping.

6 CONCLUSION

We investigated transfer of policies from multiple source tasks with common sub-goals. We showed theoretically how errors in dynamics are related to errors in policy values. We then used estimates of source task dynamics to contextually measure similarity between source and target tasks using a deep mixture model. We introduced MARS and MAPSE to use this information to transfer policies from the source tasks to the target task. Experiments showed strong performance of MARS and thus the advantages of leveraging more stable dynamics as a novel means of deep contextual transfer.

Acknowledgements

We would like to thank the reviewers, who provided constructive comments that significantly improved the paper.

References

Panagiotis Aivaliotis, Konstantinos Georgoulias, and George Chryssolouris. The use of digital twin for predictive maintenance in manufacturing. *International Journal of Computer Integrated Manufacturing*, 32(11):1067–1080, 2019.

Christophe Andrieu, Nando De Freitas, Arnaud Doucet, and Michael I Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1-2):5–43, 2003.

Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

David S Broomhead and David Lowe. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, Royal Signals and Radar Establishment Malvern (United Kingdom), 1988.

Tim Brys, Anna Harutyunyan, Matthew E Taylor, and Ann Nowé. Policy transfer using reward shaping. In *AAMAS*, pages 181–188, 2015.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, pages 465–472, 2011.

Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *AAMAS*, pages 433–440, 2012.

Jacob D Durrant and J Andrew McCammon. Molecular dynamics simulations and drug discovery. *BMC biology*, 9(1):71, 2011.

Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. In *ICLR*, 2021.

Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *AAMAS*, pages 720–727, 2006.

Michael Gimelfarb, Scott Sanner, and Chi-Guhn Lee. Reinforcement learning with multiple experts: A bayesian model combination approach. *NeurIPS*, 31:9528–9538, 2018.

Michael Gimelfarb, Scott Sanner, and Chi-Guhn Lee. Epsilon-bmc: A bayesian ensemble approach to epsilon-greedy exploration in model-free reinforcement learning. In *UAI*, pages 476–485. PMLR, 2020.

Anirudh Goyal, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, and Yoshua Bengio. Reinforcement learning with competitive ensembles of information-constrained primitives. In *ICLR*, 2020.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *ICRA*, pages 3389–3396. IEEE, 2017.

Anna Harutyunyan, Sam Devlin, Peter Vrancx, and Ann Nowé. Expressing arbitrary reward functions as potential-based advice. In *AAAI*, 2015.

Fengxiang He and Dacheng Tao. Recent advances in deep learning theory. *arXiv preprint arXiv:2012.10931*, 2020.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, 2018.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *ICLR*, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012.

Andrey Kurenkov, Ajay Mandlekar, Roberto Martin-Martin, Silvio Savarese, Animesh Garg, Michael Danielczuk, Ashwin Balakrishna, Matthew Matl, David Wang, Ken Goldberg, et al. Ac-teach: A bayesian actor-critic method for policy learning with an ensemble of suboptimal teachers. In *ICRA*, 2019.

Alessandro Lazaric. Transfer in reinforcement learning: a framework and a survey. In *Reinforcement Learning*, pages 143–173. Springer, 2012.

Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NeurIPS*, pages 1071–1079, 2014.

Mao Li and D Kudenko. Reinforcement learning from multiple experts demonstrations. In *ALA*, volume 18, 2018.

Siyuan Li and Chongjie Zhang. An optimal online method of selecting source policies for reinforcement learning. In *AAAI*, 2018.

Siyuan Li, Fangda Gu, Guangxiang Zhu, and Chongjie Zhang. Context-aware policy reuse. In *AAMAS*, pages 989–997, 2019.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, page 361–368, 2001.

Thomas P Minka. Bayesian model averaging is not model combination. *Available electronically at http://www. stat. cmu. edu/minka/papers/bma. html*, pages 1–2, 2000.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Kristine Monteith, James L Carroll, Kevin Seppi, and Tony Martinez. Turning bayesian model averaging into bayesian model combination. In *IJCNN*, pages 2657–2663. IEEE, 2011.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, pages 7559–7566, 2018.

Andrew Y Ng and Stuart J Russell. Algorithms for inverse reinforcement learning. In *ICML*, pages 663–670, 2000.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pages 278–287, 1999.

Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. In *NeurIPS*, pages 3686–3697, 2019.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Janarthanan Rajendran, Aravind S Lakshminarayanan, Mitesh M Khapra, P Prasanna, and Balaraman Ravindran. Attend, adapt and transfer: Attentive deep architecture for adaptive transfer from multiple sources in the same domain. In *ICLR*, 2017.

Benjamin Rosman, Majd Hawasly, and Subramanian Ramamoorthy. Bayesian policy reuse. *Machine Learning*, 104(1):99–127, 2016.

M. Seo, L. F. Vecchietti, S. Lee, and D. Har. Rewards prediction-based credit assignment for reinforcement learning with sparse binary rewards. *IEEE Access*, 7: 118776–118791, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2936863.

Guodong Shao and Moneer Helu. Framework for a digital twin in manufacturing: Scope and requirements. *Manufacturing Letters*, 24:105–107, 2020.

Halit Bener Suay, Tim Brys, Matthew E Taylor, and Sonia Chernova. Learning from demonstration for shaping through inverse reinforcement learning. In *AAMAS*, pages 429–437, 2016.

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

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

Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots. In *Robotics: Science and Systems*, 2018.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *JMLR*, 10 (Jul):1633–1685, 2009.

Andrea Tirinzoni, Mattia Salvini, and Marcello Restelli. Transfer of samples in policy search via multiple importance sampling. In *ICML*, pages 6264–6274, 2019.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Eric Wiewiora, Garrison W Cottrell, and Charles Elkan. Principled methods for advising reinforcement learning agents. In *ICML*, pages 792–799, 2003.

Tianpei Yang, Jianye Hao, Zhaopeng Meng, Zongzhang Zhang, Yujing Hu, Yingfeng Chen, Changjie Fan, Weixun Wang, Wulong Liu, Zhaodong Wang, and Jiajie Peng. Efficient deep reinforcement learning via adaptive policy transfer. In *IJCAI*, 2020.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

Contextual Policy Transfer in Reinforcement Learning Domains via Deep Mixtures-of-Experts (Supplementary Material)

Michael Gimelfarb* **Scott Sanner*** **Chi-Guhn Lee**

Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada

Abstract

In this Appendix, we include and provide discussion for additional plots that had to be left out of the main paper due to space limitations. We also describe all hyper-parameters chosen to reproduce the experiments in the main paper.

ADDITIONAL PLOTS

Figure 8 illustrates the test performance on all three transfer learning experiments (Transfer-Maze, Transfer-CartPole and Transfer-SparseLunarLander) using different values of the reuse parameter p_t for the MAPSE algorithm. Figure 9 demonstrates the test performance on all transfer learning experiments using different values of the reuse parameter p_t for the UCB-based policy reuse algorithm. Figure 10 illustrates the test performance on all transfer learning experiments using different learning rates β for the last layer of the option-value network for the option-based context-aware policy reuse algorithm CAPS.

FURTHER IMPLEMENTATION DETAILS

All code was written and executed using Eclipse PyDev running Python 3.7. All neural networks were initialized and trained using Keras with TensorFlow backend (version 1.14), and weights were initialized using the default setting. The Adam optimizer was used to train all neural networks. Experiments were run on an Intel 6700-HQ Quad-Core processor with 8 GB RAM running on the Windows 10 operating system. The hyper-parameter settings used in the experiments are listed in Table 1.

* we had to decrease the learning rate for MARS and reward shaping using a single policy to avoid instability

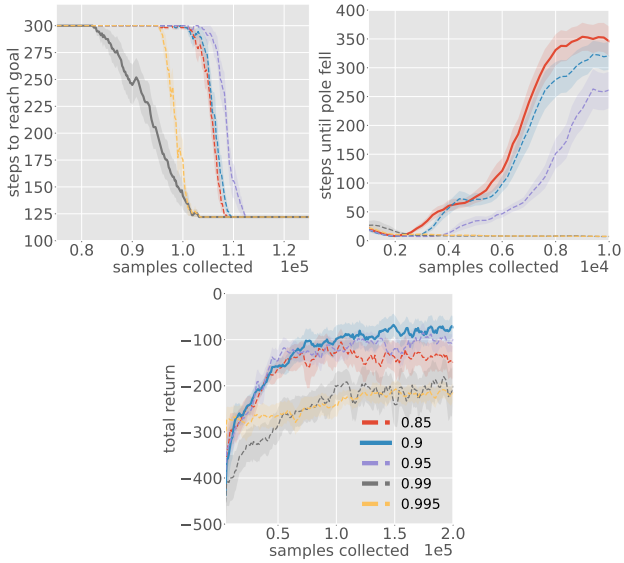


Figure 8: Smoothed mean test performance using the greedy policy for different value of p_t for MAPSE, on Transfer-Maze, Transfer-CartPole, and Transfer-SparseLunarLander (left to right). We ran 20 trials for Transfer-Maze and Transfer-CartPole and 10 trials for Transfer-SparseLunarLander.

** we report the best value found in $\{0.01, 0.001, 0.0001\}$ in the deep learning case
*** $p_t = p^t$ where t is the episode number; we report the best $p \in \{0.85, 0.9, 0.95, 0.99, 0.995\}$

Also, please note that for the imperfect transfer setting, we set $c = 0.001$ for MARS and reduce the learning rate to 0.6, since the algorithm converges much slower for larger values of c or becomes unstable.

*Affiliate to Vector Institute, Toronto, Canada.

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

Parameters				
Name	Description	Transfer-Maze	Transfer-CartPole	Transfer-SparseLunarLander
T	maximum roll-out length	300	500	1000
γ	discount factor	0.95	0.98	0.99
ε_t	exploration probability	0.12	$\max\{0.01, 0.99^t\}$	$\max\{0.01, 0.9925^t\}$
	learning rate of Q-learning*	0.08, 0.8		
	replay buffer capacity		5000	20000
B	batch size		32	64
Q	topology of DQN		4-40-40-4	8-120-100-4
	hidden activation of DQN		ReLU	ReLU
	learning rate of DQN*		0.0002, 0.0005	0.0002, 0.0005
	learning rate for termination function weights**	0.4	0.01	0.0001
	target network update frequency (in batches)		500	100
	L2 penalty of DQN		10^{-6}	10^{-6}
$\hat{f}_i$	topology of dynamics model		8-50-50-4	12-100-100-8
	hidden activation of dynamics model		ReLU	ReLU
	learning rate of dynamics model		0.001	0.001
	L2 penalty of dynamics model		10^{-6}	10^{-6}
ν_i	Gaussian kernel precision		5×10^5	5×10^5
$\mathbf{a}$	topology of mixture model	58-30-30-4	4-30-30-3	8-30-30-3
	hidden activation of mixture	ReLU	ReLU	ReLU
λ	learning rate of mixture	0.001	0.001	0.001
	training epochs/batch for mixture	4	3	1
c	PBRS scaling factor	1.0	2.0	20.0
p_t	probability of following source policies***	0.99^t (MAPSE), 0.85^t (UCB)	0.85^t (MAPSE), 0.85^t (UCB)	0.9^t (MAPSE), 0.95^t (UCB)

Table 1: Hyper-parameter settings.

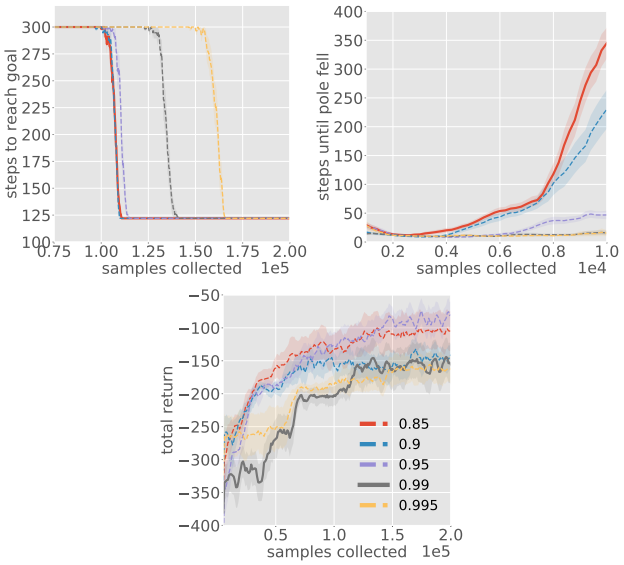


Figure 9: Smoothed mean test performance using the greedy policy for different value of p_t for UCB, on Transfer-Maze, Transfer-CartPole, and Transfer-SparseLunarLander (left to right). We ran 20 trials for Transfer-Maze and Transfer-CartPole and 10 trials for Transfer-SparseLunarLander.

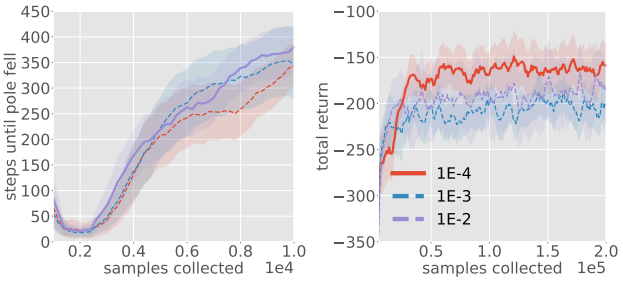


Figure 10: Smoothed mean test performance using the greedy policy for different value of termination learning rate for CAPS. From left to right: (1) number of steps balanced on Transfer-CartPole, and (2) total return on Transfer-SparseLunarLander. We ran 20 trials for Transfer-CartPole and 10 trials for Transfer-SparseLunarLander.