COMPOSING ENTROPIC POLICIES USING DIVERGENCE CORRECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep reinforcement learning (RL) algorithms have made great strides in recent years. An important remaining challenge is the ability to quickly transfer existing skills to novel tasks, and to combine existing skills with newly acquired ones. In domains where tasks are solved by composing skills this capacity holds the promise of dramatically reducing the data requirements of deep RL algorithms, and hence of greatly increasing their applicability. Recent work has studied ways of composing behaviors represented in the form of action-value functions. We analyze these methods to highlight their strengths and weaknesses, and point out situations where each of them is susceptible to poor performance. To perform this analysis we extend generalized policy improvement to the max-entropy framework and introduce a method for the practical implementation of successor features in continuous action spaces. Then we propose a novel approach which achieves an approximately optimal result. This method works by explicitly learning the (discounted, future) divergence between policies. We study this approach in the tabular case and propose a scalable variant that is applicable in multi-dimensional continuous action spaces. We compare our approach with existing ones on a range of non-trivial continuous control problems with compositional structure, and demonstrate near-optimal performance despite requiring less information than competing approaches.

1 INTRODUCTION

Reinforcement learning algorithms coupled with powerful function approximators have recently achieved a series of successes playing video games (Mnih et al., 2015) or Go (Silver et al., 2016), as well as in simulated and real robotic control (Lillicrap et al., 2015; Kalashnikov et al., 2018). Unfortunately, while being extremely powerful deep reinforcement learning (DRL) algorithms often require a large number of interactions with the environment to achieve good results, partially because they are often applied "from scratch" rather than in settings where they can leverage existing experience. This reduces their applicability in domains where generating experience is expensive, or learning from scratch is challenging.

The data efficiency of DRL algorithms is affected by various factors and significant research effort has been directed at achieving improvements (e.g. Popov et al., 2017). At the same time the development of basic locomotor behavior in humans can, in fact, require large amounts of experience and practice (Adolph et al., 2012), and it can take significant effort and training to master complex, high-speed skills (Haith & Krakauer, 2013). Once such skills have been acquired, however, humans rapidly put them to work in new contexts and to solve new tasks, suggesting transfer learning as an important mechanism.

Transfer learning has been explored extensively in multiple fields of the machine learning community (see e.g. Weiss et al., 2016, for a recent review). For instance, in the computer vision literature it is common practice to transfer experience in the form of visual feature vectors(Huh et al., 2016; Kornblith et al., 2018).

In RL and robotics the transfer of knowledge from one task to another has been studied from a variety of angles. For the purpose of this paper we are interested in methods that are suitable for transfer in the context of high-dimensional motor control problems. We further focus on model-free approaches, which are evident in human motor control (Haith & Krakauer, 2013), and have recently

been used by a variety of scalable deep RL methods (e.g. Lillicrap et al., 2015; Mnih et al., 2015; Schulman et al., 2017; Kalashnikov et al., 2018).

Transfer may be especially valuable in domains where a small set of skills can be composed, in different combinations, to solve a variety of tasks. Different notions of compositionality have been considered in the RL and robotics literature. For instance, 'options' are associated with discrete units of behavior that can be sequenced, thus emphasizing composition in time (Precup et al., 1998). In this paper we are concerned with a rather distinct notion of compositionality, namely how to combine and blend potentially concurrent behaviors. This form of composition is particularly relevant in high-dimensional continuous action spaces, where it is possible to achieve more than one task simultaneously (e.g. walking somewhere while juggling).

One approach to this challenge is via the composition of task rewards. Specifically, we are interested in the following question: If we have previously solved a set of tasks with similar transition dynamics but different reward functions, how can we leverage this knowledge to solve new tasks which can be expressed as a convex combination of those rewards functions?

This question has recently been studied in two independent lines of work: by Barreto et al. (2017; 2018) in the context of successor feature (SF) representations used for Generalized Policy Improvement (GPI) with deterministic policies, and by Haarnoja et al. (2018a); van Niekerk et al. (2018) in the context of maximum entropy policies. These approaches operate in distinct frameworks but both achieve skill composition by combining the Q-functions associated with previously learned skills.

We clarify the relationship between the two approaches and show that both can perform well in some situations but achieve poor results in others, often in complementary ways. We introduce a novel method of behavior composition that that can consistently achieve good performance.

Our contributions are as follows:

1. We provide an analysis of when GPI, and "optimistic" compositions of entropy-regularized policies (Haarnoja et al., 2018a) transfer. We construct both tabular and continuous action tasks where both fail to transfer well.
2. We introduce succcessor features (SF) in the context of maximum entropy and extend GPI to this case (max-ent GPI).
3. We propose a correction term – which we call Divergence Correction (DC)– based on the Rényi divergence between policies which allows us, in principle, to recover the optimal policy for transfer for any convex combination of rewards.
4. We demonstrate a practical implementation of these methods in continuous action spaces using adaptive importance sampling and compare the approaches introduced here: max-ent GPI and DC with optimism(Haarnoja et al., 2018a) and Conditional Q functions (Schaul et al., 2015) in a variety of non-trivial continuous action transfer tasks.

2 BACKGROUND

2.1 MULTI-TASK RL

We consider Markov Decision Processes defined by the tuple $\mathcal{M}$ containing: a state space $\mathcal{S}$, action space $\mathcal{A}$, a start state distribution $p(s_1)$, a transition function $p(s_{t+1}|s_t, a_t)$, a discount $\gamma \in [0, 1)$ and a reward function $r(s_t, a_t, s_{t+1})$. The objective of RL is to find a policy $\pi(a|s) : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ which maximises the discounted expected return from any state $J(\pi) = \mathbb{E}_{\pi,\mathcal{M}} \left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau\right]$ where the expected reward is dependent on the policy π and the MDP $\mathcal{M}$.

We formalize transfer as in Barreto et al. (2017); Haarnoja et al. (2018a), as the desire to perform well across all tasks in a set $\mathcal{M} \in \mathcal{T}'$ after having learned policies for tasks $\mathcal{M} \in \mathcal{T}$, without additional experience. We assume that $\mathcal{T}$ and $\mathcal{T}'$ are related in two ways: all tasks share the same state transition function, and tasks in $\mathcal{T}'$ can be expressed as convex combinations of rewards associated with tasks in set $\mathcal{T}$. So if we write the reward functions for tasks in $\mathcal{T}$ as the vector $\boldsymbol{\phi} = (r_1, r_2, \dots)$, tasks in $\mathcal{T}'$ can be expressed as $r_{\mathbf{w}} = \boldsymbol{\phi} \cdot \mathbf{w}$. We focus on combinations of two policies $r_b = br_i + (1 - b)r_j$ but the methods can be extended to more than two tasks.

2.2 SUCCESSOR FEATURES

Successor Features (SF) (Dayan, 1993) and Generalised Policy Improvement (GPI) (Barreto et al., 2017; 2018) provide a principled solution to the transfer in the setting defined above. SF make the additional assumption that the reward feature $\boldsymbol{\phi}$ is fully observable, that is, the agent has access to the rewards of all tasks in $\mathcal{T}$ but not $\mathcal{T}'$ during training on each individual task.

The key observation of SF representations is that linearity of the reward $r_{\mathbf{w}}$ with respect to the features $\boldsymbol{\phi}$ implies the following decomposition of the value policy of π as

$$Q^{\pi}_{\mathbf{w}}(s_t, a_t) = \mathbb{E}_{\pi}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau-t} \boldsymbol{\phi}_{\tau} \cdot \mathbf{w} | a_t\right] = \mathbb{E}_{\pi}\left[\sum_{i=t}^{\infty} \gamma^{\tau-t} \boldsymbol{\phi}_{\tau} | a_t\right] \cdot \mathbf{w} \equiv \boldsymbol{\psi}^{\pi}(s_t, a_t) \cdot \mathbf{w}, \quad (1)$$

where $\boldsymbol{\psi}^{\pi}$ is the expected discounted sum of features $\boldsymbol{\phi}$ induced by policy π. This decomposition allows us to compute the action-value for π on any task $\mathbf{w}$ by learning $\boldsymbol{\psi}^{\pi}$.

If we have a set of policies $\pi_1, \pi_2, ..., \pi_n$ indexed by i, GPI provides a principled approach to transfer on task $\pi_{\mathbf{w}}$. Namely, GPI guarantees that the GPI policy $\pi^{GPI}_{\mathbf{w}}(a_t|s_t) \equiv \delta(a_t - \arg\max_{a'_t} Q^{GPI}_{\mathbf{w}}(s_t, a'_t))$ where

$$Q^{GPI}_{\mathbf{w}}(s_t, a_t) \equiv \max_i Q^{\pi_i}_{\mathbf{w}}(s_t, a_t) \quad (2)$$

has a return at least as good as any component policy, that is, $V^{\pi^{GPI}}_{\mathbf{w}}(s) \geq \max_i V^{\pi_i}_{\mathbf{w}}(s) \, \forall s \in \mathcal{S}$.

2.3 MAXIMUM ENTROPY RL

The maximum entropy (max-ent) RL objective augments the reward with an entropy term to favor entropic solutions

$$J(\pi) = \mathbb{E}_{\pi,\mathcal{M}}\left[\sum_{i=\tau}^{\infty} \gamma^{\tau-t}(r_{\tau} + \alpha H[\pi(\cdot|s_{\tau})])\right] \quad (3)$$

where α is a parameter that determines the relative importance of the entropy term.

This objective has been considered in a number of works including Kappen (2005); Todorov (2009); Haarnoja et al. (2017; 2018a); Ziebart et al. (2008); Fox et al. (2015).

We define the action-value Q^{π} associated with eq. 3 as

$$Q^{\pi}(s_t, a_t) \equiv r_t + \mathbb{E}_{\pi}\left[\sum_{\tau=t+1}^{\infty} \gamma^{\tau-t}(r_{\tau} + \alpha H[\pi(\cdot|s_{\tau})])\right] \quad (4)$$

(notice $Q^{\pi}(s_t, a_t)$ does not include any entropy terms for the state s_t). Soft Q iteration

$$Q(s_t, a_t) \leftarrow r(s_t, a_t, s_{t+1}) + \gamma \mathbb{E}_{p(s_{t+1}|s_t, a_t)}\left[V(s_{t+1})\right] \quad (5)$$

$$V(s_t) \leftarrow \mathbb{E}_{\pi}\left[Q(s_t, a_t)\right] + \alpha H[\pi(\cdot|s_t)] = \alpha \log \int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q(s_t, a_t)) da \equiv \alpha \log Z(s_t) \quad (6)$$

where $\pi(a_t|s_t) \propto \exp(\frac{1}{\alpha} Q(s_t, a_t))$ converges to the optimal policy with standard assumptions (Haarnoja et al., 2017). The general form of the Boltzmann policy π allows it to capture multi-modal solutions. However, it also makes sampling from π challenging.

3 COMPOSING POLICIES IN MAX-ENT REINFORCEMENT LEARNING

In this section we present two novel approaches for max-ent transfer learning. In section 4 we then outline a practical method for making use of these results.

3.1 MAX-ENT SUCCESSOR FEATURES AND GENERALIZED POLICY IMPROVEMENT

We introduce max-ent SF, which provide a practical method for computing the value of a maximum entropy policy under any convex combination of rewards. We then formulate GPI (Barreto et al., 2017) for maximum entropy policies.

We define the action-dependent SF to include the entropy of the policy, excluding the current state, analogous to the max-entropy definition of Q^π in (4):

$$\boldsymbol{\psi}^\pi(s_t, a_t) \equiv \boldsymbol{\phi}_t + \mathbb{E}_\pi\left[\sum_{i=\tau+1}^{\infty} \gamma^{\tau-t}(\boldsymbol{\phi}_\tau + \alpha\mathbf{1}\cdot H[\pi(\cdot|s)])\right] = \phi_t + \gamma\mathbb{E}_{p(s_{t+1}|s_t,a_t)}\left[\boldsymbol{\Upsilon}(s_{t+1})\right] \tag{7}$$

where $\mathbf{1}$ is a vector of ones of the same dimensionality as ϕ and we define the state-dependent successor features as the expected $\boldsymbol{\psi}^\pi$ in analogy with $V^\pi(s)$:

$$\boldsymbol{\Upsilon}^\pi(s) \equiv \mathbb{E}_{a\sim\pi(\cdot|s)}\left[\boldsymbol{\psi}^\pi(s,a)\right] + \alpha\mathbf{1}\cdot H[\pi(\cdot|s)]. \tag{8}$$

The max-entropy action-value of π for any convex combination of rewards $\mathbf{w}$ is then given by $Q^\pi_{\mathbf{w}}(s,a) = \boldsymbol{\psi}^\pi(s,a)\cdot\mathbf{w}$. Max-ent SF allow us to estimate the action-value of previous policies on a new task. We show that, as in the deterministic case, there is a principled way to combine these estimates.

Theorem 3.1 (Max-Ent Generalized Policy Improvement) *Let $\pi_1, \pi_2, ..., \pi_n$ be n policies with α-max-ent action-value functions $Q^1, Q^2, ..., Q^n$ and value functions $V^1, V^2, ..., V^n$. Define*

$$\pi(a|s) \propto \exp\left(\tfrac{1}{\alpha}\max_i Q^i(s,a)\right).$$

Then,

$$Q^\pi(s,a) \geq \max_i Q^i(s,a) \text{ for all } s\in\mathcal{S} \text{ and all } a\in\mathcal{A}, \tag{9}$$

$$V^\pi(s) \geq \max_i V^i(s) \text{ for all } s\in\mathcal{S}, \tag{10}$$

where $Q^\pi(s,a)$ is the α-max-ent action-value function of π, and $V^\pi(s)$ is the max-ent state-value function of π.

Proof: See appendix A.1. Once the agent has learned $\boldsymbol{\psi}^{\pi_i}(s,a)$, the SFs of policies π_i, we define the max-ent GPI policy for task $\mathbf{w}$ as $\pi^{GPI}_{\mathbf{w}}(a|s) \propto \exp(\frac{1}{\alpha}\max_i Q^{\pi_i}_{\mathbf{w}}(s,a)) = \exp(\frac{1}{\alpha}\max_i \boldsymbol{\psi}^{\pi_i}(s,a)\cdot\mathbf{w})$.

3.2 Divergence Correction (DC)

Haarnoja et al. (2018a) introduced a simple approach to policy combination by estimating the action-value for the joint task $r_b = br_i + (1-b)r_j$ from the optimal action-values of the component tasks Q^i and Q^j

$$Q^{Opt}_b(s,a) \equiv bQ^i(s,a) + (1-b)Q^j(s,a). \tag{11}$$

When using Boltzmann policies defined by Q, the resulting policy, $\pi^{Opt}_b(a|s) \propto \exp(\frac{1}{\alpha}Q^{Opt}_b(s,a))$, is the product distribution of the two component policies. We refer to π^{Opt}_b as the "optimistic" policy, as it acts according to the optimistic assumption that the optimal returns of Q^i and Q^j will be, simultaneously, achievable. Haarnoja et al. (2018a) focus on the case when $b = \frac{1}{2}$ and prove that the regret of this estimator can be bounded by the discounted sum of Rényi divergence of the two component policies. Intuitively, if the two policies have large areas of agreement in the actions, the composed policy will perform well. As we analyse later, both max-ent GPI we presented above, and the optimistic policy can, in different ways, fail to transfer well in some situations.

Here we show, at the cost of learning a function conditional on the task combination, it is in principle possible to recover the optimal policy for the convex transfer tasks, without direct experience on those tasks, by correcting for the optimistic bias in Q^{Opt}_b. For simplicity, as in Haarnoja et al. (2018a), we restrict this to the case with only 2 tasks, but it can be extended to multiple tasks.

Theorem 3.2 (DC Optimality) *Let π_i, π_j be α max-ent optimal policies for tasks with rewards r_i and r_j with max-ent action-value functions Q^i, Q^j. Define $C^\infty_b(s_t,a_t)$ as the fixed point of*

$$C^{(k+1)}_b(s_t,a_t) = -\alpha\gamma\mathbb{E}_{p(s_{t+1}|s_t,a_t)}\left[\log\int_{\mathcal{A}}\pi_i(a_{t+1}|s_{t+1})^b\pi_j(a_{t+1}|s_{t+1})^{(1-b)}\exp(-\tfrac{1}{\alpha}C^{(k)}_b(s_{t+1},a_{t+1}))da_{t+1}\right]$$

Given the conditions for Soft Q convergence, the max-ent optimal $Q^*_b(s,a)$ for $r_b = br_i + (1-b)r_j$ is

$$Q^*_b(s,a) = bQ^i(s,a) + (1-b)Q^j(s,a) - C^\infty_b(s,a)\ \forall s\in\mathcal{S}, a\in\mathcal{A}, b\in[0,1].$$

Method	**Optimal**	**Bounded loss**	**Requires ϕ**	**Requires $f(s, a\|b)$**
Optimistic				
CondQ	✓	na	✓	✓
GPI		✓	✓	
DC	✓	na		✓

Table 1: Theoretical properties of different approaches to max-ent transfer. The methods compared are: Optimistic, CondQ, max-ent GPI (over a fixed, finite set of policies), and DC. The columns indicate whether the transfer policy is optimal, the regret of the transfer policy is bounded, whether rewards for all tasks $\boldsymbol{\phi}$ need to be observed simultaneously during training and whether the method requires learning a function conditional on the transfer task b. DC is the only method that both recovers (in principle) the optimal policy and does not require observing ϕ during training.

Proof: See appendix A.2. We call this Divergence Correction (DC) as the quantity C_b^∞ is related to the Rényi divergence between policies (see appendix A.2 for details). Learning C_b^∞ does not require any additional information (in principle) than that required to learn policies π_1 and π_2. Unlike with SF, it is not necessary to observe other task features while training the policies. On the other hand, unlike with GPI, which can be used to naturally combine any number of tasks with arbitrary weight vectors $\mathbf{w}$, in order to apply DC one must estimate $C_b^\infty(s, a)$ for all values of b. so the complexity of learning C^∞ increases significantly if more than 2 tasks are combined.

Table 1 provides a comparison on the properties of the methods we consider here. We also compare with simply learning a conditional Q function $Q(s, a|b)$ (CondQ) (e.g. Schaul et al., 2015; Andrychowicz et al., 2017). As with GPI, this requires observing the full set of task features $\boldsymbol{\phi}$, in order to compute r_b for arbitrary b.

4 Implementation using Adaptive Importance Sampling

Compositional transfer appears to be the most useful in high-dimensional, continuous action spaces, since in such action spaces it is often possible to simultaneously work towards achieving multiple objectives. Sampling the Boltzmann policies defined by Q in continuous action spaces is challenging, particularly during transfer when we need to sample from joint distributions of existing policies. We now present an approach using adaptive importance sampling with mixture proposal distributions to learn and transfer with max-ent policies.

Our approach is motivated by two observations: (i) batch computation in modern GPUs and CPUs makes importance sampling with a relatively large number of samples efficient (since the samples can be treated as a batch and computed efficiently), (ii) the product distribution for a mixture of normal distributions can be tractably sampled from. This motivates us to use adaptive importance sampling with a mixture of (truncated) normal distributions for the proposal distribution.

4.1 Learning Max-Ent Boltzmann policies with Adaptive Importance Sampling

The basic method we use is to learn proposal distributions for each policy and use this to importance sample the policy for estimating the other terms. Specifically, for each policy we learn an action-value $Q_{\theta_Q}(s, a)$, and value $V_{\theta_V}(s)$ network, and a proposal distribution $q_{\theta_q}(a|s)$.

The proposal distribution is learned by minimizing the forward KL divergence with the Boltzmann policy distribution $\pi(a|s) \propto \exp \frac{1}{\alpha} Q_{\theta_Q}(s, a)$. We use the forward divergence because it is "zero avoiding" and will typically over-estimate the support of π (Murphy, 2012) which is desirable when for the use as proposal distribution. The loss on the proposal distribution is then

$$\mathcal{L}(\theta_q) = \mathbb{E}_{\rho^\beta}\left[\mathbb{E}_{a\sim\pi(\cdot|s)}[\log \pi(a|s_t) - \log q_{\theta_q}(a|s_t)]\right] \tag{12}$$

where the expectation is over some off-policy state density ρ^β generated by exploration policy β.

However, this proposal loss itself requires we sample from π, so we importance sample. We use self-normalized importance weighting so that we do not need to compute the partition function for π. An obvious choice for the proposal distribution would be $q_{\theta_q}(a|s)$, however, we found this to

be unstable so we use a target proposal distribution $q_{\theta'_q}$ and use a mixture distribution of the target proposal distributions for all policies. The estimator for all losses are given in Appendix B.

With the learned proposal distribution we can then estimate the losses for Soft Q updates of the action-value and value.

$$\mathcal{L}(\theta_Q) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(Q_{\theta_Q}(s_t, a_t) - (r(s_t, a_t, s_{t+1}) + \gamma V_{\theta'_V}(s_{t+1})))^2\right] \tag{13}$$

$$\mathcal{L}(\theta_V) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(V_{\theta_V}(s_t) - \alpha \log \int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q_{\theta_Q}(s_t, a))da)^2\right] \tag{14}$$

The target value $\alpha \log \int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q(s, a))da$ is estimated using self-normalized importance sampling with the proposal distribution $q_{\theta_q}(a|s)$. This introduces two sources of bias, due to the self-normalization and because we apply a concave function $\log$ to the expectation. In practise, we found this estimator sufficient provided the proposal distribution was close enough to π. For acting we use the the learned proposal distribution to importance sample the policy. We use two common tricks for DeepRL to make our algorithm more stable: target networks for V'_{θ_V} and q'_{θ_q} (Mnih et al., 2015; Lillicrap et al., 2015) and parameterizing Q with an advantage function $Q_{\theta_Q}(s, a) = V_{\theta_V}(s) + A_{\theta_A}(s, a)$ (Baird, 1994; Wang et al., 2015; Harmon et al., 1995) which is more stable when the advantage is small compared with the value.

All of these updates are off-policy, and we learned using a replay buffer of experience. This algorithm can be straightforwardly extended to learn more than one policy by using estimators for each policy π_i. Because our estimators are off-policy, we can store all experience in a single replay buffer, and make use of all experience when training our policies. We implemented this in a distributed framework with actors and learner working in parallel, see supplementary algorithm 1.

4.2 Importance Sampled Max-Ent GPI

We extended our importance sampled approach to estimate max-ent SF. Max-ent GPI requires us to learn, for each policy π_i, the expected (maximum entropy) features $\boldsymbol{\psi}_i$, in order to estimate the (entropic) value of each policy under a new convex combination task $\mathbf{w}$.

This requires that our experience tuple contain the full feature vector $\boldsymbol{\phi}$, rather than just the reward for the policy which generated the experience r_i. We learn the base policies for each reward r_i as in algorithm 1. For learning the max-ent successor features, as with the action-value estimate, we learn a state-dependent target $\boldsymbol{\Upsilon}^i_{\theta_V}$ along with $\boldsymbol{\psi}^i_{\theta_Q}$.

$$\mathcal{L}(\theta_{\boldsymbol{\Upsilon}}) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(\boldsymbol{\Upsilon}_{\theta_{\boldsymbol{\Upsilon}}}(s_t) - \mathbb{E}_{a_t \sim \pi(a_t|s_t)}[\boldsymbol{\psi}_{\theta_{\boldsymbol{\psi}}}(s_t, a_t) + \alpha \mathbf{1}(-Q_{\theta_Q}(s_t, a_t) + \alpha \log Z(s_t))])^2\right]$$

To learn these quantities we use an analogous approach to the action-values (using the relationship in eq (8) and the proposal distribution $q_{\theta_q}(a|s)$, full details in appendix B). We use a target network for $\boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}$ and found that, because these updates when using experience shared between tasks is far off-policy, it is necessary to have a longer target update period than for V. We parametrized $\boldsymbol{\psi}$ as a "psi-vantage" $\boldsymbol{\psi}(s, a) = \boldsymbol{\Upsilon}(s) + \boldsymbol{\psi}^A(s, a)$.

4.3 Divergence Correction

All that is required for transfer using optimistic policy combination (eq. 11, Haarnoja et al. (2018a)) is the max-ent action values of each task. We showed in section 3.2 that if we can learn the fixed point of $C^\infty_b(s, a)$ we can correct this optimism and recover the optimal action-value $Q^*_b(s, a)$.

We use the recursive relationship in $C^\infty_b(s, a)$ to estimate it with a TD-0 estimator. However, this requires learning a conditional estimator for any value of b, so as to support arbitrary task combinations. Fortunately, since C^∞_b depends only on the policies and transition function, it is possible to estimate C^∞_b for different values of b without directly generating roll-outs of this value by sampling b at each update. We learn C^∞_b as $C_{\theta_C}(s, a, b)$, for each pair of policies π_i, π_j with loss

$$\mathcal{L}(\theta_C) = \mathbb{E}_{s \sim \rho^\beta, b \sim U(0,1)}[\tfrac{1}{2}(C_{\theta_C}(s, a, b) + \alpha\gamma \mathbb{E}_{p(s'|s,a)}[\log \int_{\mathcal{A}} \exp(b \log \pi_i(a'|s') + (1-b)\pi_j(a'|s') - \frac{1}{\alpha} C_{\theta_{C'}}(s', a', b))da'])^2]. \tag{15}$$

We approximate this loss with an importance sample using a mixture distribution $p(a'|s')$ of q_i, q_i and the product distribution $q_{prod} \propto q_i(a'|s')q_j(a'|s')$ and sample $b \in U(0,1)$ at each update (the full estimator is given in appendix B). As before with functions we use target networks and an advantage parametrization $C(s,a,b) = C^A(s,a,b) + C^B(s,b)$. Note that, unlike GPI and CondQ (next section), learning C_b^∞ does not require observer ϕ.

We also considered a heuristic approach where we learned C only for $b = \frac{1}{2}$ (this is typically approximately the largest divergence). This avoids the complexity of a conditional estimator and we estimate C_b^∞ as $\hat{C}_b^\infty(s,a) \sim 4b(1-b)C_{1/2}^\infty(s,a)$. This heuristic can be justified by considering Gaussian policies with similar variance (see appendix C) and often worked well in practise. We call this heuristic DC-Cheap, and we can also make use of the max-ent GPI information to correct for over-estimates of C_b^∞, $Q^{DC-Cheap+GPI}(s,a) = \max(Q^{OPT}(s,a) - \hat{C}_b^\infty(s,a), Q^{GPI}(s,a))$.

4.4 Cond Q

As a baseline, we consider instead of learning $C(s,a,b)$, just directly learning a conditional Q function using a similar approach of sampling b each update $Q(s,a,b)$ (Schaul et al., 2015). This, like GPI but unlike DC, requires observing ϕ during training so the reward on task b can be estimated. We provide the full details of this in appendix B.

4.5 Sampling Compositional Policies

Above we have defined different approaches to estimating the action-value $Q_b(s,a)$ for the transfer task r_b from the existing policies. Sampling from the Boltzmann policy defined by Q_b is non-trivial. We wish to avoid offline training of a sampler, as required in Haarnoja et al. (2018a) during transfer.

We take advantage of the product distribution of proposals being approximately tractable. So we can sample from $q_{ij}(a|s) \propto q_i(a|s)q_j(a|s)$. If the proposal distributions fit the policies well, the product distribution will be a good proposal distribution for when the optimistic assumption is valid. However, when $C_b^\infty(s,a)$ is large, then the optimal action during transfer may be unlikely under the base policies and this proposal will be poor. To deal with this case, we take advantage of the efficiency of computing a large importance sample in the same state space (since it can be efficiently batched) and we sample from a mixture distribution of all policies, all policy products and the uniform distribution. Empirically, we find this is sufficient to approximately recover the optimal policy in our experiments. We use the same proposal distribution for all transfer approaches. Alternatively, we could iteratively adapt the proposal distribution online (Kalashnikov et al. (2018) did this for deterministic policies).

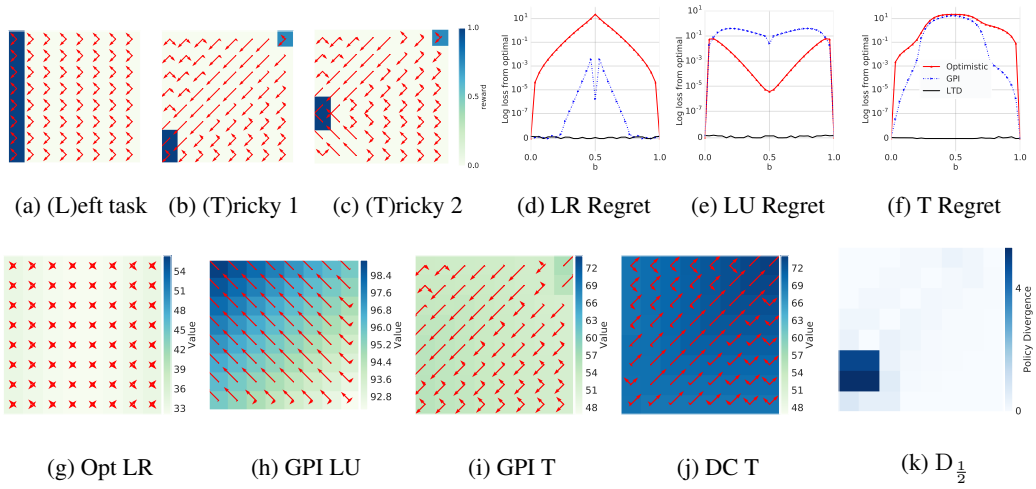
5 Experiments

5.1 Discrete, tabular environment

We first consider some illustrative tabular cases of compositional transfer. These highlight where GPI and Optimistic transfer can have high regret (figure 1). As expected, we find that GPI performs well when the optimal policy is close to one of the existing policies; the optimistic composition performs well when both subtask policies are compatible. The "tricky" task is illustrative of the sorts of task we will focus on in the next section: It provides non-overlapping rewards for each task in one corner, while lower value overlapping rewards are provided in the other corner of the grid world (cf. 1). This means that the optimal policy for the joint task does not resemble either existing policy and both GPI and optimism perform poorly. DC performs well in all cases.

5.2 Continuous action spaces

We next compare the different approaches in more challenging continuous control tasks. We train max-ent policies to solve individual tasks using the importance sampling approach from section 4 and then assess transfer on convex combinations of the rewards. We train all approaches using the same experience and proposal distribution.



(a) (L)eft task (b) (T)ricky 1 (c) (T)ricky 2 (d) LR Regret (e) LU Regret (f) T Regret

(g) Opt LR (h) GPI LU (i) GPI T (j) DC T (k) $D_{\frac{1}{2}}$

Figure 1: **Policy combination in the tabular case**. All tasks are in an infinite-horizon tabular 8x8 world. The action space is the 4 diagonal movements (actions at the boundary transition back to the same state) (**a**-**c**) shows 3 reward functions (color indicates reward, dark blue $r = +1$, light blue $r = 0.75$). The arrows indicate the action likelihoods for the max-ent optimal policy for each task. (**d**-**f**) The log regret for the max-ent returns for 3 qualitatively distinct compositional tasks $r_b = br_i + (1-b)r_j$, using different approaches to transfer from the base policies. The compositional tasks we consider are left-right (LR), left-up (LU) and the "tricky" tasks (T).
(**d**) GPI performs well when the subtasks are incompatible, meaning the optimal policy is near one of the component policies. (**g**) The optimistic policy performs poorly in these situations, resulting in indecision about which subtask to commit to.
(**e**) Conversely, when the subpolicies are similar, such as on the LU task, optimistic transfers well while the GPI policy (**h**) does not consistently take advantage of the compatibility of the two tasks to simultaneously achieve both subgoals. (**i**) Neither GPI nor the optimistic policies (**j** shows GPI, by the Optimistic policy is similar) perform well when the optimal policy is dissimilar to either existing task policy. The two tricky task policies are compatible in many states but have a high-divergence in the bottom-left corner since the rewards are non-overlapping there (**k**), thus the optimal policy on the composed task is to move to the top right corner where there are overlapping rewards. By learning, and correcting for, this future divergence between policies, DC results in optimal policies for all task combinations including tricky (**k**).

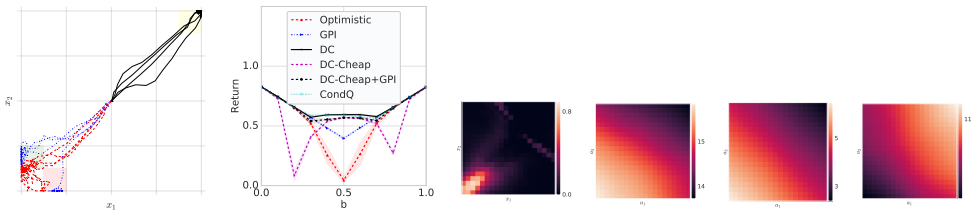
We focus on the "tricky" tasks with multi-modal solutions. Figure 2 examines the transfer policies in detail in a simple point-mass task and shows how the estimated C_b^∞ corrects the optimistic Q and dramatically changes the policy.

We then examine conceptually similar tasks in more difficult domains: a 5 DOF planar manipulator reaching task (figure 3), 3 DOF jumping ball and 8 DOF ant (figure 4). We see that DC recovers a near optimal policy in all case. The performance of GPI depends noticeably on the choice of α. DC-Cheap, which is a simpler heuristic, performs almost as well as DC in the tasks we consider except for the point mass task. When bounded by GPI (DC-Cheap+GPI) it performs well for the point mass task as well, suggesting simple approximations of C_b^∞ may be sufficient in some cases.[1]

6 Discussion

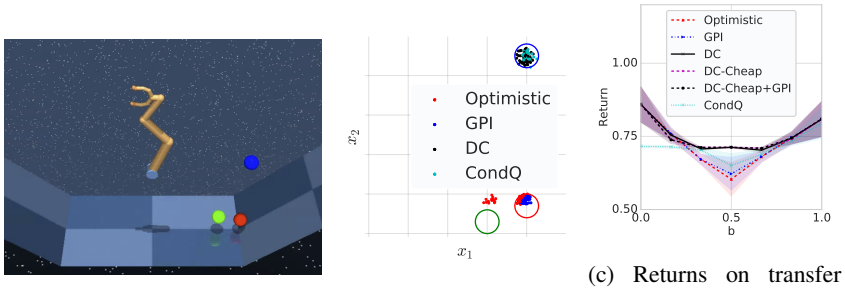
We have presented two approaches to transfer learning via convex combinations of rewards in the maximum entropy framework: max-ent GPI and DC. We have shown that, under standard assumptions, the max-ent GPI policy performs at least as well as its component policies, and that DC recovers the optimal policy. Todorov (2009) and (Saxe et al., 2017; van Niekerk et al., 2018) previously considered optimal composition of max-ent policies. However, these approaches require stronger assumptions than max-ent SF or DC, namely that reward states are absorbing and that the

[1] We provide videos of the more interesting tasks at `https://tinyurl.com/yaplfwaq`.



(a) Trajectories (b) Returns (c) $D_{\frac{1}{2}}$ (d) Q^{Opt} (e) $C^{\infty}_{1/2}$ (f) Q^{LTD}

Figure 2: **Tricky point mass**. The "tricky" tasks in a simple, continous action space (2-D velocity controlled pointed mass). (**a**) Environment and example trajectories. The rewards are $(r_1 = 1, r_2 = 0)$, $(0, 1)$ and $(0.75, 0.75)$ for the green, red and yellow squares. Lines show sampled trajectories (starting in the center) for the compositional task $r_{1/2}$ with Optimistic (red), GPI (blue) and DC (black). Only DC, DC heuristics and CondQ (not shown) find the optimal transfer policy of navigating to yellow reward area for the joint task which is the optimal solution for the combined task. (**b**) The returns for each transfer method. DC and CondQ methods recover significantly better performance than GPI, and the optimistic policy performs poorly due to the non-overlapping rewards. (**c**) The Rényi divergence of the two base policies as a function of position. As expected, the two policies are compatible until near the bottom left corner where the rewards are non-overlapping. (**d**) Q^{Opt} at the center position for the combined task. As both policies prefer moving left and down, most of the energy is on these actions. (**e**) However, the future divergence $C^{\infty}_{1/2}$ under these actions is high, which results in the (**f**) DC Q policy different significantly from Optimistic.

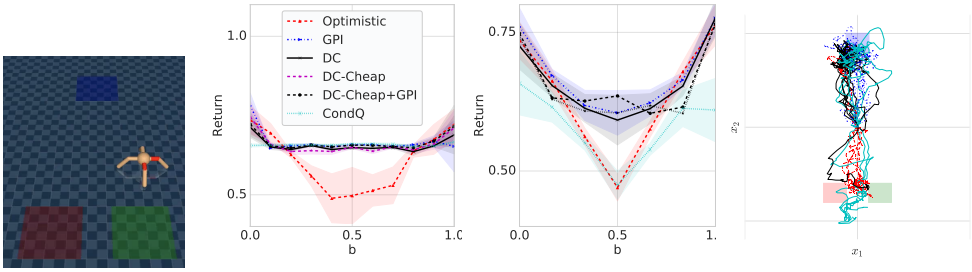


(a) Planar manipulator tricky (b) Finger Position (c) Returns on transfer task

Figure 3: **"Tricky" task with planar manipulator**. The "tricky" tasks with a 5D torque-controlled planar manipulator. The training tasks consist of (mutually exclusive) rewards of $(1, 0)$, $(0, 1)$ when the finger is at the green and red targets respectively, reward $(0.75, 0.75)$ at the blue target. (**b**) Finger position at the en (of the trajectoriesstard ting from randomly sampled start states) for the transfer task with circles indicating the rewards. DC and Cond-Q trajectories reach towards the blue target (the optimal solution) while optimistic and GPI trajectories primarily reach towards one of the suboptimal partial solutions. (**c**) The returns on the transfer tasks (shaded bars show SEM, 5 seeds).

joint reward is restricted to the softmax of the component rewards (soft OR). By contrast, DC does not restrict the class of MDPs and *learns* how compatible policies are, allowing approximate recovery of optimal transfer policies both when the component rewards are jointly achievable (AND), and when only one sub-goal can be achieved (OR).

We have compared our methods with conditional action-value functions (CondQ) (Schaul et al., 2015, e.g.) and optimistic policy combination (Haarnoja et al., 2018a). Further, we have presented a practical algorithm for training DC and max-ent GPI models in continous action spaces using adaptive importance sampling. We have compared these approaches, along with heuristic approximations of DC, and demonstrated that DC recovers an approximately optimal policy during transfer across a variety of high-dimensional control tasks. Empirically we have found CondQ may be harder to learn than DC, and it requires additional observation of ϕ during training.



(a) Ant (b) JB Returns (c) Ant returns (d) Trajectories

Figure 4: **"Tricky" task with mobile bodies.** "Tricky" task with two bodies: a 3 DOF jumping ball (supplementary figure 5) and (**a**) 8 DOF ant (both torque controlled). The task has rewards $(1, 0), (0, 1)$ in the green and red boxes respectively and $(0.75, 0.75)$ in the blue square. (**b-c**) Returns for both walkers when started in the center position. The optimistic approach does not recover the optimal policy for the combinatorial task while the other approaches largely do, although CondQ does not learn a good policy on the ant (shaded bars show SEM, 3 seeds for jumping ball, 5 seeds for ant). (**e**) Sampled trajectories of the ant on the transfer task starting from a neutral position for $b = \frac{1}{2}$. GPI and DC consistently go to the blue square (optimal), CondQ and Optimistic do not.

REFERENCES

Karen E Adolph, Whitney G Cole, Meghana Komati, Jessie S Garciaguirre, Daryaneh Badaly, Jesse M Lingeman, Gladys LY Chan, and Rachel B Sotsky. How do you learn to walk? thousands of steps and dozens of falls per day. *Psychological science*, 23(11):1387–1394, 2012.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Leemon C Baird. Reinforcement learning in continuous time: Advantage updating. In *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, volume 4, pp. 2448–2453. IEEE, 1994.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pp. 4055–4065, 2017.

Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the International Conference on Machine Learning*, pp. 501–510, 2018.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

Manuel Gil, Fady Alajaji, and Tamas Linder. Rényi divergence measures for commonly used univariate continuous distributions. *Information Sciences*, 249:124–131, 2013.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1803.06773*, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018b.

Adrian M Haith and John W Krakauer. Model-based and model-free mechanisms of human motor learning. In *Progress in motor control*, pp. 1–21. Springer, 2013.

Mance E Harmon, Leemon C Baird III, and A Harry Klopf. Advantage updating applied to a differential game. In *Advances in neural information processing systems*, pp. 353–360, 1995.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

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

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073*, 2017.

Doina Precup, Richard S Sutton, and Satinder Singh. Theoretical results on reinforcement learning with temporally abstract options. In *European conference on machine learning*, pp. 382–393. Springer, 1998.

Andrew M Saxe, Adam Christopher Earle, and Benjamin S Rosman. Hierarchy through composition with multitask lmdps. 2017.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

Emanuel Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pp. 1856–1864, 2009.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Benjamin van Niekerk, Steven James, Adam Earle, and Benjamin Rosman. Will it blend? composing value functions in reinforcement learning. *arXiv preprint arXiv:1807.04439*, 2018.

Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. A survey of transfer learning. *Journal of Big Data*, 3:1–40, 2016.

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A PROOFS

A.1 MAX-ENT GENERALIZED POLICY IMPROVEMENT

Theorem 3.1 (Max-Ent Generalized Policy Improvement) *Let $\pi_1, \pi_2, ..., \pi_n$ be n policies with α-max-ent action-value functions $Q^1, Q^2, ..., Q^n$ and value functions $V^1, V^2, ..., V^n$. Define*

$$\pi(a|s) \propto \exp\left(\tfrac{1}{\alpha}\max_i Q^i(s,a)\right).$$

Then,

$$Q^\pi(s,a) \geq \max_i Q^i(s,a) \text{ for all } s \in \mathcal{S} \text{ and all } a \in \mathcal{A}, \tag{9}$$

$$V^\pi(s) \geq \max_i V^i(s) \text{ for all } s \in \mathcal{S}, \tag{10}$$

where $Q^\pi(s,a)$ is the α-max-ent action-value function of π, and $V^\pi(s)$ is the max-ent state-value function of π.

For brevity we denote $Q^{\max} \equiv \max_i Q^i$. Define the soft Bellman operator associated with policy π as

$$\mathcal{T}^\pi Q(s,a) \equiv r(s,a,s') + \gamma \mathbb{E}_{p(s'|s,a)}\left[\alpha H[\pi(\cdot|s')] + \mathbb{E}_{a'\sim\pi(\cdot|s')}\left[Q(s',a')\right]\right].$$

Haarnoja et al. (2018b) have pointed out that the soft Bellman operator $\mathcal{T}^\pi$ corresponds to a conventional, "hard", Bellman operator defined over the same MDP but with reward $r_\pi(s,a,s') = r(s,a,s') + \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[H[\pi(\cdot|s')]\right]$. Thus, as long as $r(s,a,s')$ and $H[\pi(\cdot|s')]$ are bounded, $\mathcal{T}^\pi$ is a contraction with Q^π as its fixed point. Appplying $\mathcal{T}^\pi$ to $Q^{\max}(s,a)$ we have:

$$\begin{aligned}
\mathcal{T}^\pi Q^{\max}(s,a) &= r(s,a,s') + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[-\alpha\log\pi(a'|s') + Q^{\max}(s',a')\right] \\
&= r(s,a,s') + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a), a'\sim\pi(\cdot|s')}\left[-\alpha\log\frac{\exp(\alpha^{-1}Q^{\max}(s',a'))}{Z^\pi(s')} + Q^{\max}(s',a')\right] \\
&= r(s,a,s') + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\alpha\log Z^\pi(s')\right].
\end{aligned}$$

Similarly, if we apply $\mathcal{T}^{\pi_i}$, the soft Bellman operator induced by policy π_i, to $Q^{\max}(s,a)$, we obtain:

$$\mathcal{T}^{\pi_i} Q^{\max}(s,a) = r(s,a,s') + \gamma\mathbb{E}_{s'\sim p(\cdot|s,a), a'\sim\pi_i(\cdot|s')}\left[-\alpha\log\pi_i(a'|s') + Q^{\max}(s',a')\right].$$

We now note that the Kullback-Leibler divergence between π_i and π can be written as

$$\begin{aligned}
D_{\mathrm{KL}}(\pi_i(\cdot|s)\|\pi(\cdot|s)) &= \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[\log\pi_i(a|s) - \log\pi(a|s)\right] \\
&= \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[\log\pi_i(a|s) - \frac{1}{\alpha}Q^{\max}(s,a) + \log Z^\pi(s)\right].
\end{aligned}$$

The quantity above, which is always nonnegative, will be useful in the subsequent derivations. Next we write

$$\begin{aligned}
\mathcal{T}^\pi Q^{\max}(s,a) - \mathcal{T}^{\pi_i} Q^{\max}(s,a) &= \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\alpha\log Z^\pi(s') - \mathbb{E}_{a'\sim\pi_i(\cdot|s')}[-\alpha\log\pi_i(a'|s') + Q^{\max}(s',a')]\right] \\
&= \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\mathbb{E}_{a'\sim\pi_i(\cdot|s')}[\alpha\log Z^\pi(s') + \alpha\log\pi_i(a'|s') - Q^{\max}(s',a')]\right] \\
&= \gamma\mathbb{E}_{s'\sim p(\cdot|s,a)}\left[\alpha D_{\mathrm{KL}}(\pi_i(\cdot|s')\|\pi(\cdot|s'))\right] \\
&\geq 0.
\end{aligned} \tag{16}$$

From (16) we have that

$$\mathcal{T}^\pi Q^{\max}(s,a) \geq \mathcal{T}^{\pi_i} Q^{\max}(s,a) \geq \mathcal{T}^{\pi_i} Q^i(s,a) = Q^i(s,a) \text{ for all } i.$$

Using the contraction and monotonicity of the soft Bellman operator $\mathcal{T}^\pi$ we have

$$Q^\pi(s,a) = \lim_{k\to\infty}(\mathcal{T}^\pi)^k Q^{\max}(s,a) \geq Q^i(s,a) \text{ for all } i.$$

We have just showed (9). In order to show (10), we note that

$$\begin{aligned}
V^\pi(s) &\equiv \alpha H[\pi(\cdot|s)] + \mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a)\right] \\
&\geq \alpha H[\pi(\cdot|s)] + \mathbb{E}_{a\sim\pi}\left[Q^{\max}(s,a)\right] \\
&= \alpha\log Z^\pi(s).
\end{aligned} \tag{17}$$

Similarly, we have, for all i,

$$\begin{aligned} V^i(s) &= \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q^i(s,a) - \alpha\log\pi_i(a|s)\right] \\ &\leq \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q^{\max}(s,a) - \alpha\log\pi_i(a|s)\right] \\ &= \alpha\log Z^\pi(s) - \alpha D_{\mathrm{KL}}(\pi_i(\cdot|s)\|\pi(\cdot|s)) \\ &\leq \alpha\log Z^\pi(s). \end{aligned} \tag{18}$$

The bound (10) follows from (17) and (18).

A.2 DC Proof

Theorem 3.2 (DC Optimality) *Let π_i, π_j be α max-ent optimal policies for tasks with rewards r_i and r_j with max-ent action-value functions Q^i, Q^j. Define $C_b^\infty(s_t, a_t)$ as the fixed point of*

$$C_b^{(k+1)}(s_t,a_t) = -\alpha\gamma\mathbb{E}_{p(s_{t+1}|s_t,a_t)}\left[\log\int_{\mathcal{A}}\pi_i(a_{t+1}|s_{t+1})^b\pi_j(a_{t+1}|s_{t+1})^{(1-b)}\exp(-\tfrac{1}{\alpha}C_b^{(k)}(s_{t+1},a_{t+1}))da_{t+1}\right]$$

Given the conditions for Soft Q convergence, the max-ent optimal $Q_b^*(s,a)$ for $r_b = br_i + (1-b)r_j$ is

$$Q_b^*(s,a) = bQ^i(s,a) + (1-b)Q^j(s,a) - C_b^\infty(s,a)\,\forall s\in\mathcal{S}, a\in\mathcal{A}, b\in[0,1].$$

We follow a similar approach to Haarnoja et al. (2018a) but without making approximations and generalizing to all convex combinations.

First note that since π_i and π_j are optimal then $\pi_i(a|s) = \exp(\frac{1}{\alpha}(Q^i(s,a) - V^i(s)))$.

For brevity we use s and s' notation rather than writing the time index.

Define

$$Q_b^{(0)}(s,a) \equiv bQ^i(s,a) + (1-b)Q^j(s,a) \tag{19}$$

$$C^{(0)}(s,a) \equiv 0 \tag{20}$$

and consider soft Q-iteration on r_b starting from $Q_b^{(0)}$. We prove, inductively, that at each iteration $Q_b^{(k)} = bQ^i(s,a) + (1-b)Q^j(s,a) - C^{(k)}(s,a)$.

This is true by definition for $k=0$.

$$Q_b^{(k+1)}(s,a) = r_b(s,a) + \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp\frac{1}{\alpha}Q_b^{(k)}(s',a')da'\right] \tag{21}$$

$$= r_b(s,a) + \tag{22}$$

$$\gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp(\frac{1}{\alpha}(bQ^i(s',a') + (1-b)Q^j(s',a') - C^{(k)}(s',a')))da'\right]$$

$$= r_b(s,a) + \tag{23}$$

$$\mathbb{E}_{p(s'|s,a)}\left[bV^i(s') + (1-b)V^j(s') + \alpha\log\int_{\mathcal{A}}\exp(b\log\pi_i(a'|s') + (1-b)\log\pi_j(a'|s') - \frac{1}{\alpha}C^{(k)}(s',a'))da'\right]$$

$$= bQ^i(s,a) + (1-b)Q^j(s,a) + \tag{24}$$

$$\alpha\gamma\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\exp(b\log\pi_1(a'|s') + (1-b)\log\pi_2(a'|s') - \frac{1}{\alpha}C^{(k)}(s',a'))da'\right]$$

$$= bQ^i(s,a) + (1-b)Q^i(s,a) - C_b^{(k+1)}(s,a). \tag{25}$$

Since soft Q-iteration converges to the α max-ent optimal soft Q then equation 19 holds at the limit.

One can get an intuition for $C_b^\infty(s,a)$ by noting that

$$C_b^{(1)}(s,a) = \gamma\alpha\mathbb{E}_{p(s'|s,a)}\left[(1-b)\,\mathrm{D}_b\left(\pi_1(\cdot|s)\|\pi_2(\cdot|s)\right)\right] \tag{26}$$

where D_b is the Rényi divergence of order b. $C_b^\infty(s,a)$ can be seen as the discount sum of divergences, weighted by the unnormalized product distribution $\pi_1(a|s)^b\pi_2(a|s)^{1-b}$.

B ALGORITHM DETAILS

B.1 IMPORTANCE SAMPLED ESTIMATORS

For clarity we enumerate the various importance-sampled estimators used in our approach.

All of these updates are off-policy. We use a replay buffer R and learn by sampling minibatches of SARS tuples of size B, we index over the batch dimension with l and use s'_l to denote the state following s_l, so the tuple consists of (s_l, a_l, r_l, s'_l). For importance sampled estimators we sample N actions for each state s_l and use a_{lk} to denote sample k for state l.

Proposal loss The proposal loss equation 12 is sampled using a mixture distribution $p(a|s)$ as described in the main text.

$$p(a|s) = \frac{1}{n+1}\left(\frac{1}{V^{\mathcal{A}}} + \sum_{i=1}^{n} q^i_{\theta'_q}(a|s)\right) \tag{27}$$

where $V^{\mathcal{A}}$ is the volume of the action space (which is always bounded in our case).

$$\mathcal{L}(\theta_q) \approx -\frac{1}{B}\sum_{k=1}^{B}\sum_{l=1}^{N} w_{kl} \log q_{\theta_q}(a|s_t), \tag{28}$$

$$w'_{kl} = \frac{\frac{1}{\alpha}(Q_{\theta_Q}(s_k, a_{kl}))}{p(a_{kl}|s_k)};\ \ w_{kl} = \frac{w_{kl'}}{\sum_{m=1}^{N} w'_{km}}. \tag{29}$$

Value loss The value estimator for equation 15

$$\mathcal{L}(\theta_V) \approx \frac{1}{B}\sum_{l=1}^{B}\left(V_{\theta_V}(s_l) - \alpha \log\left[\frac{1}{N}\sum_{k=1}^{N}\frac{\exp(\frac{1}{\alpha}Q_{\theta_Q}(s_l, a_{lk}))}{q_{\theta_q}(a_{lk}|s_l)}\right]\right)^2. \tag{30}$$

Action-value loss The action-value estimator for equation 34

$$\mathcal{L}(\theta_Q) \approx \frac{1}{2B}\sum_{l=1}^{B}(Q_{\theta_Q}(s_l, a_l) - (r_l + \gamma V_{\theta'_V}(s')))^2. \tag{31}$$

Successor Feature Value Estimator

$$\mathcal{L}(\theta_{\boldsymbol{\Upsilon}^i}) \approx \frac{1}{2B}\sum_{l=1}^{B}\sum_{k=1}^{N} w_{lk}\left[(\boldsymbol{\psi}^i_{\theta_{\boldsymbol{\psi}_i}}(s_l, a_{lk}) - Q^i_{\theta_{Q_i}}(s_l, a_{lk}) + \alpha \log Z^i(s_l))^2\right], \tag{32}$$

$$w_{lk} \propto \frac{\exp(\frac{1}{\alpha}Q^i(s_l, a_l k))}{q^i_{\theta_q}(a_{lk}|s_l)}. \tag{33}$$

We use the importance sampled estimate of $\alpha \log Z$ from eq 35, rather than the value network which may be lagging the true partition function. As with the value estimate, we use self-normalized importance sampling to avoid the weights depending on $\alpha \log Z(s_l)$ (this introduces a bias, but in practise appears to work well).

Successor Features Action Value Loss

$$\mathcal{L}(\theta_{\boldsymbol{\psi}}) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(\boldsymbol{\psi}_{\theta_Q}(s_t, a_t) - (\boldsymbol{\phi}(s_t, a_t, s_{t+1}) + \gamma \boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}(s_{t+1})))^2\right]. \tag{34}$$

Successor Feature Action-Value estimator

$$\mathcal{L}(\theta_{\boldsymbol{\psi}^i}) \approx \frac{1}{2B}\sum_{l=1}^{B}(\boldsymbol{\psi}^i_{\theta_{\boldsymbol{\psi}}}(s_l, a_l) - (\boldsymbol{\phi}_l + \gamma \boldsymbol{\Upsilon}_{\theta'_{\boldsymbol{\Upsilon}}}(s'_l)))^2. \tag{35}$$

$C^{\infty}_{\theta_C}(s, a, b)$ **estimator**

$$\mathcal{L}(\theta_C) \approx \frac{1}{N}\sum_{l=1}^{B}\left(C_{\theta_C}(s_l, a_l, b_l) - \alpha\gamma \log\left[\frac{1}{N}\sum_{k=1}^{N}\frac{C^{target}_{\theta'_C}(s'_l, a'_{lk}, b_l)}{p(a_{lk'}|s_l)}\right]\right)^2, \quad (36)$$

$$C^{target}(s'_l, a'_{lk}, b_l)_{\theta'_C} \equiv \exp(\frac{1}{\alpha}(b_l Q^i_{\theta_Q}(s'_l, a'_{lk}) + (1-b_l)Q^j_{\theta_Q}(s'_l, a'_{lk}) - C_{\theta'_C}(s'_l, a'_{lk}, b_l)). \quad (37)$$

CondQ We also consider, as a control, learning the action-value function conditional on b directly (Schaul et al., 2015). We learn both a conditional value $V_{\theta_V}(s, b)$ and $Q_{\theta_Q}(s, a, b)$, again by sampling b uniformly each update.

$$\mathcal{L}(\theta_V) = \mathbb{E}_{\rho^\beta, b\sim U(0,1)}\left[\frac{1}{2}(\alpha \log \int \exp(\frac{1}{\alpha}Q(s, a, b)))^2\right], \quad (38)$$

$$\mathcal{L}\theta_Q = \mathbb{E}_{\rho^\beta, b\sim U(0,1)}\left[\frac{1}{2}(Q_{\theta_Q}(s, a, b) - (r_b + \gamma V(s', b)))^2\right], \quad (39)$$

where computing r_b for arbitrary b requires ϕ to have been observed.

We estimate Cond-Q with the same importance samples as C. We use target networks for $V_{\theta'_V}(s, b)$ and parametrize $Q_{\theta_Q}(s, a, b) = V_{\theta_V}(s, b) + A_{\theta_A}(s, a, b)$.

Algorithm 1 Distributed single task adaptive importance sampled algorithm

```
Initialize proposal network θ_q, value network parameters θ_V and action-value network parameters
θ_Q and replay R
Copy to target nets θ'
while training do                                          ▷ in parallel on each actor
    Obtain parameters θ from learner
    Roll out episode
    Add experience to replay R
end while
while training do                                          ▷ in parallel on the learner
    Sample SARS tuple from R
    Minimize L(θ_q)
    Minimize L(θ_V)
    Minimize L(θ_Q)
    if target update period then
        Update target network parameters θ_Q' ← θ_Q, θ_q' ← θ_q
    end if
end while
```

C JUSTIFICATION FOR THE DC-CHEAP HEURISTIC

We wish to estimate $C^{\infty}_b(s, a)$ (defined in Theorem 3.2) while avoiding learning a conditional function of b. We make two (substantial) assumptions to arrive at this approximation.

Firstly, we assume policies $\pi_i(a|s), \pi_j(a|s)$ are Gaussian

$$\pi_i(a|s) = \exp\left(-\frac{(a - \mu_i(s))^2}{2\sigma(s)^2}\right) \quad (40)$$

and the variance $\sigma(s)$ is the same for both policies given a state (it may vary across states).

Secondly, we assume $C^{(k)}_b(s, a) = C^{(k)}_b(s)$ is independent of action. This is approximately correct when nearby states have similar Rényi divergences between policies.

We make use of a result by Gil et al. (2013) that states that the Rényi divergence of order b for two Gaussians of the same variance is

$$\mathrm{D}_b\left(\mathcal{N}(\mu_1, \sigma)\|\mathcal{N}(\mu_2, \sigma)\right) = \frac{1}{2}\frac{b(\mu_1 - \mu_2)^2}{\sigma^2}. \quad (41)$$

We first define

$$G_b(s) \equiv (1-b)\,\mathrm{D}_b\left(\pi_i(\cdot|s)\|\pi_j(\cdot|s)\right) = -\log\int \pi_i(a|s)^b\pi_j(a|s)^{(1-b)}da. \tag{42}$$

From equation 40

$$G_b(s) = 4b(1-b)G_{\frac{1}{2}}(s). \tag{43}$$

Given these assumptions we show inductively that $C_b^{(k)}(s,a) = 4b(1-b)C_{1/2}^{(k)}(s,a)\ \forall k, b \in [0,1]$.

Since $C_b^{(0)}(s,a) = 0\ \forall b \in [0,1], a \in \mathcal{A}, s \in \mathcal{S}$ this is true for $k=0$. We show it holds inductively

$$C_b^{(k+1)}(s,a) = -\alpha\gamma\mathbb{E}_{p(s'|s,a)}\left[\log\int_{\mathcal{A}}\pi_i(a'|s')^b\pi_j(a'|s')^{(1-b)}\exp(-\frac{1}{\alpha}C_b^{(k)}(s',a'))da'\right] \tag{44}$$

$$= \gamma\mathbb{E}_{p(s'|s,a)}\left[\alpha G_b(s') + C_b^{(k)}(s')\right] \tag{45}$$

$$= 4(1-b)bC_{\frac{1}{2}}^{(k+1)}(s,a). \tag{46}$$

Obviously these assumptions are not justified. However, note that we estimate the true divergence for $C_{1/2}^{\infty}$, i.e. without any assumptions of Gaussian policies and this heuristic is used to estimate C_b^{∞} from $C_{1/2}^{\infty}$. In practise, we find this heuristic works in many situations where the policies have similar variance, particulary when bounded by GPI.

D EXPERIMENT DETAILS

All control tasks were simulated using the MuJoCo physics simulator and constructed using the DM control suite (Tassa et al., 2018) which uses the MuJoCo physics simulator (Todorov et al., 2012).

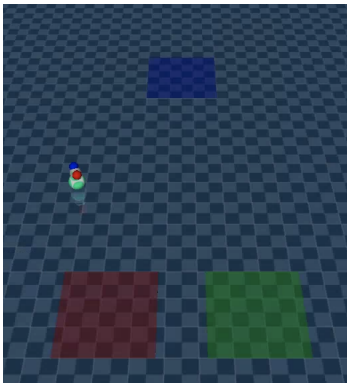


Figure 5: Jumping ball tricky task

The point mass was velocity controlled, all other tasks were torque controlled. The planar manipulator task was based off the planar manipulator in the DM control suite. The reward in all tasks was sparse as described in the main text.

During training for all tasks we start states from the randomly sampled positions and orientations. For the point mass, jumping ball and ant we evaluated transfer starting from the center (in the walker environments, the starting orientation was randomly sampled during transfer, the point mass does not have an orientation). For the planar manipulator transfer was tested from same random distribution as in training. For all tasks were learned infinite time horizon policies.

Transfer is made challenging by the need for good exploration. That was not the focus on this work. We aided exploration in several ways: during training we acted according to a higher-temperature policy $\alpha_e = 2\alpha$. We also sampled actions uniformly in an ϵ-greedy fashion with $\epsilon = 0.1$ and added Gaussian exploration noise during training. This was sufficient to explore the state space for most

tasks. For the planar manipulator and the jumping ball, we found it necessary to induce behavior tasks by learning tasks for reaching the blue target. This behavior policy was, of course, only used for experience and not during transfer.

Below we list the hyper-parameters and networks use for all experiment. The discount γ and α were the only sensitive parameters that we needed to vary between tasks to adjust for the differing magnitudes of returns and sensitivity of the action space between bodies. If α is too small then the policies often only find one solution and all transfer approaches behave similarly, while for large α the resulting policies are too stochastic and do not perform well.

Proposal learning rate	10^{-3}
All other learning rates	10^{-4}
Value target update period	200
Proposal target update period	200
$\boldsymbol{\Upsilon}$ target update period	500
Number of importance samples for all estimators during learning	200
Number of importance samples for acting during training	50
Number of importance samples for acting during transfer	1000

Table 2: Parameters the same across all experiments

The state vector was preprocessed by a linear projection of $3\times$ its dimension and then a $\tanh$ non-linearity. All action-state networks (Q, $\boldsymbol{\psi}$, C) consisted of 3 hidden layers with elu non-linearities (Clevert et al., 2015), with both action and preprocessed state projected by linear layers to be of the same dimensionality and used for input the first layer. All value networks and proposal networks consisted of 2 layers with elu non-linearities. The number of neurons in each layer was varied between environments, but was kept the same in all networks and layers (we did not sweep over this parameter, but choose a reasonable number based on our prior on the complexity of the task).

Below we list the per task hyper-parameters

Task	**Number of units**	α	γ
Point mass	22	1	0.99
Planar Manipulator	192	0.05	0.99
Jumping Ball	192	0.2	0.9
Ant	252	0.1	0.95

Table 3: Parameters varied between experiments