Entropic Policy Composition with Generalized Policy Improvement and Successor Features

Jonathan J Hunt[1], Andre Barreto[1], Timothy P Lillicrap[1, 2], Nicolas Heess[1]
jjhunt, andrebarreto, countzero, heess @ google.com
[1]DeepMind, [2]University College London

Abstract

Transferring and generalizing previously learned solutions to new tasks is a promising avenue for improving data efficiency in reinforcement learning (RL). The maximum entropy (max-ent) RL framework trades off task reward with trajectory entropy and it has been demonstrated that the resulting policies can be composed to obtain solutions to new tasks by summing their action-value functions. Here we introduce an alternative approach to compositionality in high-dimensional continuous action spaces. Successor features (SF) provide a mechanism for decomposing value functions into dynamics and reward and have previously been used for transfer in small discrete action spaces. We introduce a maximum entropy formulation of successor features (MESF), as well as of Generalized Policy Improvement (GPI), which provides a principled approach to combining multiple existing policies to solve a new task. In a tabular setting we show the similarities and differences between different policy composition schemes and show that GPI/MESF can perform well compared to the simple sum of action-value functions in certain circumstances. We also provide a practical algorithm for GPI/MESF for high-dimensional continuous action spaces and provide results on an 8-DOF ant.

1 Introduction

Transfer in RL has been formalized in many ways. Here, we focus on model-free motor control, a scalable approach with some notable recent successes[e.g. 14, 17, 11].

We restrict transfer to 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?

Transfer in this setting has recently been studied in two independent lines of work: by Barreto et al. [2, 3] with deterministic policies in small discrete action spaces, and by Haarnoja et al. [8] in the context of maximum entropy (max-ent) policies in continuous action spaces. These approaches operate in distinct frameworks but both achieve skill composition by combining the action-value functions associated with previously learned skills. Related work [18, 16, 20] demonstrated optimal composition of max-ent policies, but only by imposing strong assumptions on the class of MDPs.

We introduce max-ent Successor Features (MESF) and extend GPI to the max-ent framework, including providing a perspective on max-ent GPI as a form of approximate inference. We clarify the relationship between GPI/MESF and [8] in a tabular setting, and demonstrate a practical implementation of max-ent GPI in continuous action spaces using adaptive importance sampling.

Infer to Control: NIPS 2018 Workshop on Probabilistic Reinforcement Learning and Structured Control.

2 Background

2.1 Multi-task RL

We consider Markov Decision Processes $\mathcal{M}$ defined by 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 $r(s_t, a_t)$. The standard RL objective is to find a policy $\pi(a|s) : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ which maximises the discounted expected return from any state.

We formalize transfer as in [2, 8], as the desire to perform well across tasks in $\mathcal{M} \in \mathcal{T}'$ 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 $r_{\mathbf{w}} = \boldsymbol{\phi} \cdot \mathbf{w}$.

2.2 Successor Features

Successor Features (SF) [5] and Generalised Policy Improvement (GPI) [2, 3] provide a principled solution to transfer in the setting defined above. SF makes 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 $\mathcal{T}$ 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 $\boldsymbol{\phi}$ allows the action value Q^π of policy π to be decomposed 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 $\boldsymbol{\psi}^\pi$.

2.3 Maximum Entropy RL

The max-ent RL objective arises naturally from an RL as inference perspective [13][1]. In this framework, the RL problem is reframed as one of posterior inference by introducing binary random variables indicating optimality $p(\mathcal{O}_t = 1|s_t, a_t) = \exp(\frac{1}{\alpha} r(s_t, a_t))$ where α acts as a temperature.

The RL problem then becomes one of inference to find the posterior distribution $p(a_t|s_t, \mathcal{O}_{t:T} = 1)$ (see [13, fig 1b] for the graphical model). This inference problem can be solved with a single backwards pass of belief propagation with the messages

$$\beta_t(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t) = \textstyle\int_{\mathcal{S}} \beta_{t+1}(s_{t+1}) p(s_{t+1}|s_t, a_t) p(\mathcal{O}_t|s_t, a_t) ds_{t+1} \quad (2)$$

and $\beta_t(s_t) = p(\mathcal{O}_{t:T}|s_t) = \int_{\mathcal{A}} p(\mathcal{O}_{t:T}|s_t, a_t) p(a_t|s_t) da_t = \int_{\mathcal{A}} \beta_t(s_t, a_t) da_t$ where we've assumed an uninformative prior on a_t.

Given this message the optimal action distribution is

$$p(a_t|s_t, \mathcal{O}_{t:T}) \propto p(\mathcal{O}_{t:T}|s_t, a_t) p(a_t|s_t) = \beta(s_t, a_t) \quad (3)$$

Again using the assumption that the action prior is uninformative. This policy can be shown to maximise the RL objective augmented with an entropy term, which favors more entropic trajectories and has been considered in a number of works [e.g. 12, 18, 7, 8, 22, 6].

The messages have intuitive interpretations as the optimal soft Q action-value $Q^*(s_t, a_t) = \alpha \log \beta_t(s_t, a_t)$ and value $V^*(s_t) = \alpha \log \beta_t(s_t)$.

2.4 Optimistic Policy Composition

Haarnoja et al. [8] introduced a simple approach to policy composition with max-ent policies for transfer to a new task $r_{\mathbf{w}}$ by adding the action-values of the component tasks

$$Q^{Opt}_{\mathbf{w}}(s, a) \equiv \textstyle\sum_i \mathbf{w}_i Q^i(s, a) \quad (4)$$

[1] As in the initial derivation in [13], we focus on deterministic dynamics, without discounting and with negative rewards. All of these restrictions can be removed without significantly modifying the conclusions.

The Boltzmann policy defined by Q^{Opt}, $\pi^{Opt}(a|s) \propto \exp(\frac{1}{\alpha} Q^{Opt}_{\mathbf{w}}(s,a))$, is the product distribution of the component policies. We refer to π^{Opt} as the "optimistic" policy, as it acts according to the optimistic assumption that the optimal returns of all subtasks will be, simultaneously, achievable.

3 Max-Ent Policy Composition with GPI

3.1 Max-Ent Successor Features

We introduce max-ent SF, which provides a practical method for computing the value of a max-ent policy under any convex combination of rewards. We then formulate GPI [2] for max-ent 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}:

$$\boldsymbol{\psi}^{\pi}(s_t, a_t) \equiv \boldsymbol{\phi}_t + \gamma \mathbb{E}_{p(s_{t+1}|s_t,a_t)} \left[\boldsymbol{\Upsilon}^{\pi}(s_{t+1}) \right] \tag{5}$$

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 $V^{\pi}(s)$:

$$\boldsymbol{\Upsilon}^{\pi}(s) \equiv \mathbb{E}_{a \sim \pi(\cdot|s)} \left[\boldsymbol{\psi}^{\pi}(s,a) \right] + \alpha \mathbf{1} \cdot H[\pi(\cdot|s)]. \tag{6}$$

Max-ent SF allow us to compute the soft action-value of π on any task $r_{\mathbf{w}}$[2] of rewards $\mathbf{w}$ as $Q^{\pi}_{\mathbf{w}}(s,a) = \boldsymbol{\psi}^{\pi}(s,a) \cdot \mathbf{w}$.

3.2 Max-Ent Generalized Policy Improvement

We first outline Max-Ent GPI from an approximate inference perspective. In the exact inference case we can obtain the optimal policy from the message $p(a_t|s_t, \mathcal{O}_{t:T}) \propto \beta(s_t, a_t)$. Now, we consider the case where we do not have access to the message $\beta(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t)$ but only a set of approximate messages $\{\beta^i(s_t, a_t)\}$. These messages convey the probability of achieving an optimal return while following policy π^i for all future actions $\beta^i(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t, \pi^i)$. One interpretation is to consider performing approximating inference using messages from a related, but not identical graphical model[3]. Importantly, $\beta(s_t, a_t) \geq \beta^i(s_t, a_t)$ since $\beta(s_t, a_t)$ is the probability of an optimal trajectory under the optimal policy. The max-ent successor features described above describe one practical method for computing $\beta^i(s_t, a_t)$ for a new task from a set of existing policies.

Since each $\beta^i(s_t, a_t)$ is a lower-bound on $p(\mathcal{O}_{t:T}|s_t, a_t)$, we use the tightest lower bound to estimate the posterior $p(a_t|s_t, \mathcal{O}_{t:T}) \propto p(\mathcal{O})_{t:T}|s_t, a_t) \approx \max_i \beta^i(s_t, a_t)$.

We restate this in RL terms and provide a proof to support the intuition that the max-ent value of this estimate is at least as good as following any single estimate $p^i(a_t|s_t, \mathcal{O}_{t:T}) \propto \beta^i(s_t, a_t)$.

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,

$$Q^{\pi}(s,a) \geq \max_i Q^i(s,a) \text{ for all } s \in \mathcal{S} \text{ and all } a \in \mathcal{A}, \tag{7}$$

$$V^{\pi}(s) \geq \max_i V^i(s) \text{ for all } s \in \mathcal{S}, \tag{8}$$

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. Practically, GPI/MESF provides a principled approach to composing entropic policies for transfer. During training, we learn the successor features for $\boldsymbol{\psi}^{\pi_i}(s,a)$ for each policy. For transfer this provides us the value of existing policies on a new task, and we use max-ent GPI to combine all existing policies. In contrast to the optimistic transfer, GPI acts conservatively by estimating the future value as a lower-bound on its true value. Note that although GPI values the future under existing policies, the GPI policy is not a mixture of existing policies.

[2] $\mathbf{w}$ must be convex. The constraint on convexity arises to ensure $\mathbf{1}H[\pi(\cdot|s)] \cdot \mathbf{w} = H[\pi(\cdot|s)$, so the temperature of all policies is identical.

[3] This model has the same dynamics but different $\mathcal{O}_t$

4 Implementation and Experiments

Sampling the Boltzmann policies defined by Q is challenging, particularly during transfer. We use importance sampling with learned proposal distributions to approximate our policies. We use a mixture of normal distributions for the proposal distribution so that the product of proposal distributions can be sampled efficiently. When comparing with optimistic composition we use the same base policies and train on the same experience for both approaches. Details of the algorithm are in appendix B.

We first consider some illustrative tabular cases of compositional transfer. These highlight the complementary performance of GPI/MESF and optimistic transfer (figure 1).

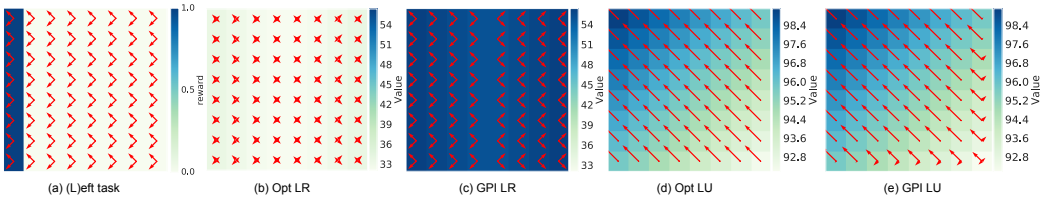

Figure 1: **Tabular policy composition** Tasks are in an infinite-horizon tabular 8x8 world. The action space is the 4 diagonal movements (actions at the boundary transition back to the same state) (**a**) The left task r_l (color indicates reward) with arrows showing the optimal max-ent policy. (**b**-**c**) Transfer to task LR $r_{lr} = \frac{1}{2}r_l + \frac{1}{2}r_r$. These tasks are not compatible so optimistic transfers results in an indecisive policy (arrows show policy, color shows value) while GPI performs well. Right r_r and up r_u tasks are defined similarly. (**d-e**) For left-up the policies are compatible, so optimistic transfer performs well, GPI acts conservatively according to the lower bound and performs marginally worse.

We demonstrate our algorithm and transfer on an 8 DOF ant with a continous action space (figure 2) [4]. In this task, the policies are not compatible, so optimistic transfer performs poorly, while GPI/MESF acts conservatively and performs well.

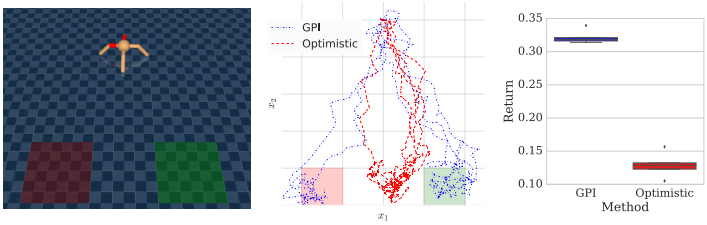

Figure 2: (**a**) Transfer on an 8-DOF ant. The two base tasks consist of rewards for reaching the red and green squares. We compared GPI/MASF against optimistic for the composed task. (**b**) Trajectories from the composed task. Optimistic attempts to "get the best of both worlds" while GPI is more conservative. (**c**) Box-plot of the returns for the compositional task from 5 seeds.

5 Conclusion

Here we have introduced GPI/MESF. We provided an inference perspective of max-ent GPI, as a variational approximation of the optimal posterior. We demonstrated a practical algorithm for making use of these ideas, to our knowledge the first examples of SF and GPI in continuous action spaces. We compared with a previous approach [8] in a simple tabular case and an 8-DOF ant.

[4] Videos of the base and transfer policies can be viewed at `https://tinyurl.com/y89g4dsd`

References

[1] L. 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, pages 2448–2453. IEEE, 1994.

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

[3] A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *Proceedings of the International Conference on Machine Learning*, pages 501–510, 2018.

[4] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

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

[6] R. Fox, A. Pakman, and N. Tishby. Taming the noise in reinforcement learning via soft updates. *arXiv preprint arXiv:1512.08562*, 2015.

[7] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

[8] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine. Composable deep reinforcement learning for robotic manipulation. *arXiv preprint arXiv:1803.06773*, 2018.

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

[10] M. E. Harmon, L. C. Baird III, and A. H. Klopf. Advantage updating applied to a differential game. In *Advances in neural information processing systems*, pages 353–360, 1995.

[11] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

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

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

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[15] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020, 9780262018029.

[16] A. M. Saxe, A. C. Earle, and B. S. Rosman. Hierarchy through composition with multitask lmdps. 2017.

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

[18] E. Todorov. Compositionality of optimal control laws. In *Advances in Neural Information Processing Systems*, pages 1856–1864, 2009.

[19] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[20] B. van Niekerk, S. James, A. Earle, and B. Rosman. Will it blend? composing value functions in reinforcement learning. *arXiv preprint arXiv:1807.04439*, 2018.

[21] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[22] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 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,

$$Q^\pi(s,a) \geq \max_i Q^i(s,a) \text{ for all } s \in \mathcal{S} \text{ and all } a \in \mathcal{A}, \tag{7}$$

$$V^\pi(s) \geq \max_i V^i(s) \text{ for all } s \in \mathcal{S}, \tag{8}$$

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. [9] 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. Applying $\mathcal{T}^\pi$ 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

$$\begin{aligned}
\mathcal{T}^\pi Q^{\max}(s,a) - \mathcal{T}^{\pi_i} Q^{\max}(s,a) &= \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\alpha \log Z^\pi(s') - \mathbb{E}_{a' \sim \pi_i(\cdot|s')}[-\alpha \log \pi_i(a'|s') + Q^{\max}(s',a')]\right] \\
&= \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\mathbb{E}_{a' \sim \pi_i(\cdot|s')}[\alpha \log Z^\pi(s') + \alpha \log \pi_i(a'|s') - Q^{\max}(s',a')]\right] \\
&= \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)}\left[\alpha D_{\mathrm{KL}}(\pi_i(\cdot|s') \| \pi(\cdot|s'))\right] \\
&\geq 0.
\end{aligned} \tag{9}$$

From (9) 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 (7). In order to show (8), we note that

$$\begin{aligned}
V^\pi(s) &\equiv \alpha H[\pi(\cdot|s)] + \mathbb{E}_{a \sim \pi}\left[Q^\pi(s,a)\right] \\
&\geq \alpha H[\pi(\cdot|s)] + \mathbb{E}_{a \sim \pi}\left[Q^{\max}(s,a)\right] \\
&= \alpha \log Z^\pi(s).
\end{aligned} \tag{10}$$

Similarly, we have, for all i,

$$\begin{aligned} V^i(s) &= \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q^i(s,a) - \alpha \log \pi_i(a|s)\right] \\ &\leq \mathbb{E}_{a\sim\pi_i(\cdot|s)}\left[Q^{\max}(s,a) - \alpha \log \pi_i(a|s)\right] \\ &= \alpha \log Z^\pi(s) - \alpha D_{\mathrm{KL}}(\pi_i(\cdot|s)\|\pi(\cdot|s)) \\ &\leq \alpha \log Z^\pi(s). \end{aligned} \tag{11}$$

The bound (8) follows from (10) and (11).

B Algorithm details

Learning and sampling the Boltzmann policies defined by Q in continuous action spaces is challenging, particularly during transfer.

B.1 Motivation

Our algorithm is motivated by two observations: (i) batch computation in modern architectures makes importance sampling with a relatively large number of samples efficient, (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.

We use neural networks to parametrise all quantities. For each policy we learn an action-value $Q_{\theta_Q}(s,a)$, value $V_{\theta_V}(s)$ and proposal distribution $q_{\theta_q}(a|s)$, each of which is described by a neural net with parameters θ.

We learn everything using off-policy temporal difference learning (TD(0)). The use of an off-policy algorithm allows us to make the most of all experience, by storing all experience, generated across all policies in a replay buffer R and improving all policies using this shared experience.

We implement this in a distributed RL framework with actors and a learner operating in parallel. Algorithm 1 outlines the basic algorithm.

We use two common tricks for DeepRL to make our algorithm more stable: target networks and parameterizing Q with an advantage function [1, 21, 10] which is more stable when the advantage is small compared with the value.

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
```

B.2 Losses and Estimators

Here we enumerate all of the losses and their estimators.

We learn by sampling minibatches of SARS tuples from the replay buffer 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 action sample k for state l.

B.2.1 Proposal loss

The proposal distribution is optimized by minimizing the forward KL divergence with the Boltzmann policy $\pi(a|s) \propto \exp \frac{1}{\alpha} Q_{\theta_Q}(s, a)$. This KL is "zero avoiding" and will typically over-estimate the support of π [15] which is desirable for a proposal distribution, particularly during transfer.

This leads to the following loss for θ_q

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

where the expectation is over some off-policy state density ρ^β generated by exploration policy β (in our case, all experience in the replay buffer).

Since this objective is itself defined in terms of samples from π we use importance sampling to obtain approximate samples. 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 mixture distribution $p(a|s)$ containing equally weighted components consisting of the target proposal distribution $q_{\theta'_q}(a|s)$ for all policies and the uniform distribution.

$$p(a|s) = \frac{1}{n+1}\left(\frac{1}{V^{\mathcal{A}}} + \sum_{i=1}^{n} q^i_{\theta'_q}(a|s)\right) \tag{13}$$

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

With this proposal this loss estimator is

$$\mathcal{L}(\theta_q) \approx -\frac{1}{B}\sum_{k=1}^{B}\sum_{l=1}^{N} w_{kl} \log q_{\theta_q}(a|s_t), \tag{14}$$

$$w'_{kl} = \frac{\frac{1}{\alpha}(Q_{\theta_Q}(s_k, a_{kl}))}{p(a_{kl}|s_k)};\;\; w_{kl} = \frac{w_{kl'}}{\sum_{m=1}^{N} w'_{km}}. \tag{15}$$

B.2.2 Value loss

The soft value loss is

$$\mathcal{L}(\theta_V) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(V_{\theta_V}(s_t) - \alpha \log \int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q_{\theta_Q}(s_t, a))da)^2\right] \tag{16}$$

We estimate this using importance sampling with the proposal distribution $q_{\theta_q}(a|s)$

$$\mathcal{L}(\theta_V) \approx \frac{1}{2B}\sum_{l=1}^{B}\left(V_{\theta_V}(s_l) - \alpha \log Z\right)^2 \tag{17}$$

$$Z = \left[\frac{1}{N}\sum_{k=1}^{N} \frac{\exp(\frac{1}{\alpha} Q_{\theta_Q}(s_l, a_{lk}))}{q_{\theta_q}(a_{lk}|s_l)}\right] \tag{18}$$

This estimator introduces bias because we apply a concave function $\log$ to the estimate of Z. In practice we found this estimator sufficient, provided the proposal distribution was close enough to π. We also used importance sampling to estimate π while acting.

B.2.3 Action-value loss

The TD(0) loss for Q_{θ_Q} is

$$\mathcal{L}(\theta_Q) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(Q_{\theta_Q}(s_t, a_t) - (r(s_t, a_t, s_{t+1}) + \gamma V_{\theta'_V}(s_{t+1})))^2\right] \tag{19}$$

this loss does not require sampling from π. For stability we use a target value function $V_{\theta_{V'}}$ for estimating the value of s'

$$\mathcal{L}(\theta_Q) \approx \frac{1}{2B}\sum_{l=1}^{B}(Q_{\theta_Q}(s_l, a_l) - (r_l + \gamma V_{\theta'_V}(s')))^2. \tag{20}$$

The action-value is parametrized as an advantage function $Q_{\theta_Q}(s, a) = V_{\theta'_V}(s) + A_{\theta_A}(s, a)$.

B.3 Importance Sampled Max-Ent GPI

We extended our importance sampled approach to estimate max-ent SF. This requires us to learn, for each policy π_i, the expected (max-ent) 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. Then $\boldsymbol{\psi}_{\theta_\psi}$ and $\boldsymbol{\Upsilon}_{\theta_\Upsilon}$ are learned in an analogous way to V and Q.

As with V_{θ_V}, we use a target network for $\boldsymbol{\Upsilon}_{\theta'_\Upsilon}$ and advantage parametrization. We found it is necessary to have a longer target update period than for V.

B.3.1 State Dependent Successor Features loss

The state-dependent SF loss is

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

This loss is estimated using importance sampling with proposal q_{θ_q}

$$\mathcal{L}(\theta_\Upsilon) \approx \frac{1}{2B}\sum_{l=1}^{B}\sum_{k=1}^{N} w_{lk}\left[(\boldsymbol{\psi}^i_{\theta_\psi}(s_l, a_{lk}) - Q^i_{\theta_Q}(s_l, a_{lk}) + \alpha\log Z(s_l))^2\right], \tag{21}$$

$$w_{lk} \propto \frac{\exp(\frac{1}{\alpha}Q^i(s_l, a_l k))}{q^i_{\theta_q}(a_{lk}|s_l)}. \tag{22}$$

We use the importance sampled estimate of Z from eq 18, rather than the value network which may be lagging the true partition function. As with the value estimate, the $\log$ introduces a bias, but in practise appears to work well.

B.3.2 State-Action Dependent Successor Features loss

The state-action dependent successor feature loss is

$$\mathcal{L}(\theta_\psi) = \mathbb{E}_{\rho^\beta}\left[\frac{1}{2}(\boldsymbol{\psi}_{\theta_Q}(s_t, a_t) - (\boldsymbol{\phi}(s_t, a_t, s_{t+1}) + \gamma\boldsymbol{\Upsilon}_{\theta'_\Upsilon}(s_{t+1})))^2\right]. \tag{23}$$

for which we use the following estimator

$$\mathcal{L}(\theta_{\psi^i}) \approx \frac{1}{2B}\sum_{l=1}^{B}(\boldsymbol{\psi}^i_{\theta_\psi}(s_l, a_l) - (\boldsymbol{\phi}_l + \gamma\boldsymbol{\Upsilon}_{\theta'_\Upsilon}(s'_l)))^2. \tag{24}$$

$\boldsymbol{\psi}_{\theta_\psi}$ is parametrized as a "psi-vantage" $\boldsymbol{\psi}_{\theta_\psi}(s, a) = \boldsymbol{\Upsilon}_{\theta'_\Upsilon}(s) + \boldsymbol{\psi}^A_{\theta_A}(s, a)$.

C Experiment details

The ant task was simulated using the MuJoCo physics simulator [19]. The rewards were sparse as described in the main text.

During training the ant started episodes at randomly sampled positions and orientations. For testing, episodes began at the center position with randomly sampled starting orientations. All policies were learned with a discounted infinite time horizon.

Transfer is made challenging by the need for good exploration. 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.

Below we list the hyper-parameters used in the ant experiments.

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
α	0.05
γ	0.95
Number of units in each network layer	252

Table 1: Hyper-parameters

The network architecture used shared weights across all policies (but no weight sharing between different networks, e.g. proposal, value, action-value and SF networks all had separate weights). The state vector was preprocessed by a linear projection of $3\times$ its dimension and then a $\tanh$ non-linearity. The action-state networks (Q, $\boldsymbol{\psi}$) consisted of 3 hidden layers with elu non-linearities [4], with both action and the preprocessed state projected by linear layers to be of the same dimensionality and concatenated as input to the network. All value networks and proposal networks consisted of 2 layers with elu non-linearities.