MULTITASK SOFT OPTION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Multitask Soft Option Learning (MSOL), a hierarchical multitask framework based on Planning as Inference. MSOL extends the concept of options, using separate variational posteriors for each task, regularized by a shared prior. This allows fine-tuning of options for new tasks without forgetting their learned policies, leading to faster training without reducing the expressiveness of the hierarchical policy. MSOL avoids several instabilities during training in a multitask setting and provides a natural way to learn both intra-option policies and their terminations. We demonstrate empirically that MSOL significantly outperforms both hierarchical and flat transfer-learning baselines in challenging multi-task environments.

1 INTRODUCTION

A key challenge in Reinforcement Learning (RL) is to scale current approaches to higher complexity tasks without requiring a prohibitive number of environmental interactions. However, for many tasks, it is possible to construct or learn efficient exploration priors that allow to focus on more relevant parts of the state-action space, reducing the number of required interactions. These include, for example, reward shaping (Ng et al., 1999; Konidaris & Barto, 2006), curriculum learning (Bengio et al., 2009), some meta-learning algorithms (Wang et al., 2016; Duan et al., 2016; Gupta et al., 2018; Houthooft et al., 2018; Xu et al., 2018), and transfer learning (Caruana, 1997; Taylor & Stone, 2011; Bengio, 2012; Parisotto et al., 2015; Rusu et al., 2015; Teh et al., 2017).

One promising way to capture prior knowledge is to decompose policies into a hierarchy of sub-policies (or skills) that can be reused and combined in novel ways to solve new tasks (Dayan & Hinton, 1993; Thrun & Schwartz, 1995; Parr & Russell, 1998; Sutton et al., 1999; Barto & Mahadevan, 2003). The idea of Hierarchical RL (HRL) is also supported by findings that humans appear to employ a hierarchical mental structure when solving tasks (Botvinick et al., 2009; Collins & Frank, 2016). In such a hierarchical RL policy, lower-level, *temporally extended* skills yield directed behavior over multiple time steps. This has two advantages: i) it allows efficient exploration, as the target states of skills can be reached without having to explore much of the state space in between, and ii) directed behavior also reduces the variance of the future reward, which accelerates convergence of estimates thereof. On the other hand, while a hierarchical approach can therefore significantly speed up exploration and training, it can also severely limit the expressiveness of the final policy and lead to suboptimal performance when the temporally extended skills are not able to express the required policy for the task at hand (Mankowitz et al., 2014).

Many methods exist for constructing and/or learning skills for particular tasks (Dayan & Hinton, 1993; Sutton et al., 1999; McGovern & Barto, 2001; Menache et al., 2002; Şimşek & Barto, 2009; Gregor et al., 2016; Kulkarni et al., 2016; Bacon et al., 2017; Nachum et al., 2018a). Training on multiple tasks simultaneously is one promising approach to learn skills that are both relevant and generalise across tasks (Thrun & Schwartz, 1995; Pickett & Barto, 2002; Fox et al., 2016; Andreas et al., 2017; Frans et al., 2018). Ideally, the entire hierarchy can be trained end-to-end on the obtained return, obviating the need to specify proxy rewards for skills (Frans et al., 2018).

However, learning hierarchical policies end-to-end in a multitask setting poses two major challenges: i) because skills optimize environmental rewards directly, correctly updating them relies on already (nearly) converged master policies that use them similarly across all tasks, requiring complex training schedules (Frans et al., 2018), and ii) the end-to-end optimization is prone to local minima in

which multiple skills have learned similar behavior. This second points is explained in more detail in Appendix A.

In this paper, we propose Multitask Soft Option Learning (MSOL), a novel approach to learning hierarchical policies in a multi-task setting that extends Options (Sutton et al., 1999), a common definition for skills, and casts the concept into the Planning as Inference (PAI) framework (see, e.g., Levine, 2018, for a review). MSOL brings multiple advantages: i) it stabilizes end-to-end multitask training, removing the need for complex training schedules like in Frans et al. (2018), ii) it gives rise to coordination between master policies, avoiding local minima of the type described in Appendix A, iii) it allows fine-tuning of options, i.e. adapting them to new tasks at test-time without the risk of unlearning previously acquired useful behavior, thereby avoiding suboptimal performance due to restricted expressiveness, iv) and lastly, we show how the *soft option* framework gives rise to a natural solution to the challenging task of learning option-termination policies.

MSOL differentiates between a prior policy for each option, shared across all tasks, and a flexible task-specific posterior policy. The option prior can be fixed once it is fully trained, preventing unlearning of useful behavior even when the posteriors are updated. On new tasks, the option posteriors are initialized to the priors and regularized towards them, but are still adaptable to the specific task. This allows the same accelerated training as with 'hard' options, but can solve more tasks due to the adjustable posteriors. Furthermore, during option learning, we can train prior and posterior policies simultaneously (Teh et al., 2017), all without the need for complex training schedules (Frans et al., 2018): training is stabilized because only the priors are shared across tasks.

Our experiments demonstrate that MSOL outperforms previous hierarchical and transfer learning algorithms during transfer tasks in a multitask setting. MSOL only modifies the regularized reward and loss function, but does not require any specialized architecture. In particular, it also does not require artificial restrictions on the expressiveness of either the higher-level or intra-option policies.

2 PRELIMINARIES

An agent's task is formalized as a MDP $(\mathcal{S}, \mathcal{A}, \rho, P, r)$, consisting of the state space $\mathcal{S}$, the action space $\mathcal{A}$, the initial state distribution ρ, the transition probability $P(s_{t+1}|s_t, a_t)$ of reaching state s_{t+1} by executing action a_t in state s_t, and the reward $r(s_t, a_t)$ an agent receives for this transition.

2.1 PLANNING AS INFERENCE

Planning as inference (PAI) (Todorov, 2008; Toussaint, 2009; Kappen et al., 2012) frames RL as a probabilistic-inference problem (Levine, 2018). The agent learns a distribution $q_\phi(a|s)$ over actions a given states s, i.e., a policy, parameterized by ϕ, which induces a distribution over trajectories τ of length T, i.e., $\tau = (s_1, a_1, s_2, \ldots, a_T, s_{T+1})$:

$$q_\phi(\tau) \quad = \quad \rho(s_1) \prod_{t=1}^{T} q_\phi(a_t|s_t)\, P(s_{t+1}|s_t, a_t)\,. \tag{1}$$

This can be seen as a structured variational approximation of the optimal trajectory distribution. Note that the true initial state probability $\rho(s_1)$ and transition probability $P(s_{t+1}|s_t, a_t)$ are used in the variational posterior, as we can only control the policy, not the environment.

An advantage of this formulation is that we can incorporate information both from prior knowledge, in the form of a prior policy distribution, and the task at hand through a likelihood function that is defined in terms of the achieved reward. The prior policy $p(a_t|s_t)$ can be specified by hand or, as in our case, learned (see Section 3). To incorporate the reward, we introduce a binary *optimality variable* $\mathcal{O}_t$ (Levine, 2018), whose likelihood is highest along the optimal trajectory that maximizes return: $p(\mathcal{O}_t = 1|s_t, a_t) = \exp\big(r(s_t, a_t)/\beta\big)$. The constraint $r \in (-\infty, 0]$ can be relaxed without changing the inference procedure (Levine, 2018). For brevity, we denote $\mathcal{O}_t = 1$ as $\mathcal{O}_t \equiv (\mathcal{O}_t = 1)$. If a given prior policy $p(a_t|s_t)$ explores the state-action space sufficiently, then $p(\tau, \mathcal{O}_{1:T})$ is the distribution of desirable trajectories. PAI aims to find a policy such that the variational posterior in (1) approximates this distribution by minimizing the Kullback-Leibler (KL) divergence

$$\mathcal{L}(\phi) = \mathbb{D}_{\text{KL}}(q_\phi(\tau) \,\|\, p(\tau, \mathcal{O}_{1:T}))\,, \text{ with } p(\tau, \mathcal{O}_{1:T}) = \rho(s_1) \prod_{t=1}^{T} p(a_t|s_t) P(s_{t+1}|s_t, a_t) p(\mathcal{O}_t|s_t, a_t). \tag{2}$$

2.2 MULTI-TASK LEARNING

In a multi-task setting, we have a set of different tasks $i \in \mathcal{T}$, drawn from a task distribution with probability $\xi(i)$. All tasks share state space $\mathcal{S}$ and action space $\mathcal{A}$, but each task has its own initial-state distribution ρ_i, transition probability $P_i(s_{t+1}|s_t, a_t)$, and reward function r_i. Our goal is to learn n tasks concurrently, distilling common information that can be leveraged to learn faster on new tasks from $\mathcal{T}$. In this setting, the prior policy $p_\theta(a_t|s_t)$ can be learned jointly with the task-specific posterior policies $q_{\phi_i}(a_t|s_t)$ (Teh et al., 2017). To do so, we simply extend (2) to

$$\mathcal{L}(\{\phi_i\}, \theta) \quad = \quad \mathbb{E}_{i\sim\xi}\left[\mathbb{D}_{\text{KL}}(q_{\phi_i}(\tau) \,\|\, p_\theta(\tau, \mathcal{O}_{1:T}))\right] \quad = \quad -\frac{1}{\beta}\sum_{i=1}^{n}\sum_{t=1}^{T}\mathbb{E}_{\tau\sim q}\left[R^{\text{reg}}_{i,t}\right], \tag{3}$$

where $R^{\text{reg}}_{i,t} := r_i(s_t, a_t) - \beta \ln \frac{q_{\phi_i}(a_t|s_t)}{p_\theta(a_t|s_t)}$ is the regularised reward. Minimizing the loss in (3) is equivalent to maximizing the regularized reward $R^{\text{reg}}_{i,t}$. Moreover, minimizing the term $\mathbb{E}_{\tau\sim q}\left[\ln \frac{q_{\phi_i}(a_t|s_t)}{p_\theta(a_t|s_t)}\right]$ implicitly minimizes the expected KL-divergence $\mathbb{E}_{s_t\sim q}\left[\mathbb{D}_{\text{KL}}[q_{\phi_i}(\cdot|s_t)\|p_\theta(\cdot|s_t)]\right]$. In practise (see Appendix C.1) we will also make use of a discount factor $\gamma \in [0, 1]$. For details on how γ arises in the PAI framework we refer to Levine (2018).

2.3 OPTIONS

Options (Sutton et al., 1999) are skills that generalize primitive actions and consist of three components: i) an intra-option policy $p(a_t|s_t, z_t)$ selecting primitive actions according to the currently active option z_t, ii) a probability $p(b_t|s_t, z_{t-1})$ of terminating the *previously* active option z_{t-1}, and iii) an initiation set $\mathcal{I} \subseteq \mathcal{S}$, which we simply assume to be $\mathcal{S}$. Note that by construction, the higher-level (or master-) policy $p(z_t|z_{t-1}, s_t, b_t)$ can only select a new option z_t if the previous option z_{t-1} has terminated.

3 METHOD

We aim to learn a reusable set of options that allow for faster training on new tasks from a given distribution. Unlike prior work (Frans et al., 2018), we learn both intra-option and termination policies, while preventing multiple options from learning the same behavior.

To differentiate ourselves from classical 'hard' options, which, once learned, do not change during new tasks, we call our novel approach *soft-options* (this is further discussed in Appendix B). Each soft-option consists of an option *prior*, which is shared across all tasks, and a task-specific option *posterior*. The priors of both the intra-option policy and the termination policy capture how an option typically behaves and remain fixed once they are fully learned. At the beginning of training on a new task, they are used to initialize the task-specific posterior distribution. During training, the posterior is then regularized against the prior to prevent inadvertent unlearning. However, if maximizing the reward on certain tasks is not achievable with the prior policy, the posterior is free to deviate from it. We can thus speed up training using options, while remaining flexible enough to solve any task.

Additionally, this soft option framework also allows for learning good *priors* in a multitask setting while avoiding local minima in which several options learn the same behavior. See Figure 1 for an overview over the hierarchical prior-posterior architecture that we explain further below.

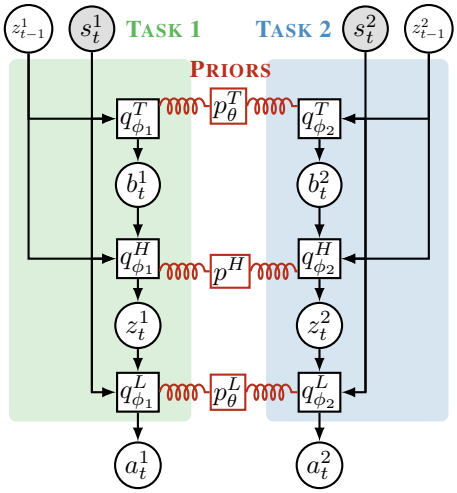


Figure 1: Two hierarchical posterior policies (left and right) with common priors (middle). For each task i, the policy conditions on the current state s^i_t and the last selected option z^i_{t-1}. It samples, in order, whether to terminate the last option (b^i_t), which option to execute next (z^i_t) and what primitive action (a^i_t) to execute in the environment.

3.1 HIERARCHICAL POSTERIOR POLICIES

To express options in the PAI framework, we introduce two additional variables at each time step t: *option selections* z_t, representing the currently selected option, and decisions b_t to *terminate* them and allow the higher-level (master) policy to choose a new option. The agent's behavior depends on the currently selected option z_t, by drawing actions a_t from the *intra-option posterior policy* $q^L_{\phi_i}(a_t|s_t, z_t)$. The selection z_t itself is drawn from a *master policy* $q^H_{\phi_i}(z_t|s_t, z_{t-1}, b_t) = (1 - b_t)\,\delta(z_t - z_{t-1}) + b_t\, q^H_{\phi_i}(z_t|s_t)$, which conditions on $b_t \in \{0, 1\}$, drawn by the *termination posterior policy* $q^T_{\phi_i}(b_t|s_t, z_{t-1})$. The master policy either continues with the previous z_{t-1} or draws a new option, where we set $b_1 = 1$ at the beginning of each episode. The joint posterior policy is

$$q_{\phi_i}(a_t, z_t, b_t|s_t, z_{t-1}) \quad = \quad q^T_{\phi_i}(b_t|s_t, z_{t-1})\, q^H_{\phi_i}(z_t|s_t, z_{t-1}, b_t)\, q^L_{\phi_i}(a_t|s_t, z_t)\,. \tag{4}$$

While z_t can be a continuous variable, we consider only $z_t \in \{1 \dots m\}$, where m is the number of available options. The induced distribution $q_{\phi_i}(\tau)$ over trajectories of task i, $\tau = (s_1, b_1, z_1, a_1, s_2, \dots, s_T, b_T, z_T, a_T, s_{T+1})$, is then

$$q_{\phi_i}(\tau) \quad = \quad \rho_i(s_1) \prod_{t=1}^{T} q_{\phi_i}(a_t, z_t, b_t|s_t, z_{t-1}) P_i(s_{t+1}|s_t, a_t). \tag{5}$$

3.2 HIERARCHICAL PRIOR POLICY

Our framework transfers knowledge between tasks by a shared prior $p_\theta(a_t, z_t, b_t|s_t, z_{t-1})$ over all joint policies (4): $p_\theta(a_t, z_t, b_t|s_t, z_{t-1}) = p^T_\theta(b_t|s_t, z_{t-1})\, p^H(z_t|z_{t-1}, b_t)\, p^L_\theta(a_t|s_t, z_t)$. By choosing p^T_θ, p^H, and p^L_θ correctly, we can learn useful temporally extended options. The parameterized priors $p^T_\theta(b_t|s_t, z_{t-1})$ and $p^L_\theta(a_t|s_t, z_t)$ are structurally equivalent to the posterior policies $q^T_{\phi_i}$ and $q^L_{\phi_i}$ so that they can be used as initialization for the latter. Optimizing the regularized return (see next section) w.r.t. θ distills the common behavior into the prior policy and softly enforces similarity across posterior distributions of each option amongst all tasks i.

The prior $p^H(z_t|z_{t-1}, b_t) = (1 - b_t)\,\delta(z_t - z_{t-1}) + b_t \frac{1}{m}$ selects the previous option z_{t-1} if $b_t = 0$, and otherwise draws options uniformly to ensure exploration. Because the posterior master policy is different on each task, there is no need to distill common behavior into a joint prior.

3.3 OBJECTIVE

We extend the multitask objective in (3) by substituting $p_\theta(\tau, \mathcal{O}_{1:T})$ and $p_{\phi_i}(\tau)$ with those induced by our hierarchical posterior policy in (4) and the corresponding prior. The resulting objective has the same form but with a new regularized reward that is maximized:

$$R^{\text{reg}}_{i,t} \quad := \quad r_i(s_t, a_t) - \underbrace{\beta \ln \frac{q^H_{\phi_i}(z_t|s_t, z_{t-1}, b_t)}{p^H(z_t|z_{t-1}, b_t)}}_{①} - \underbrace{\beta \ln \frac{q^L_{\phi_i}(a_t|s_t, z_t)}{p^L_\theta(a_t|s_t, z_t)}}_{②} - \underbrace{\beta \ln \frac{q^T_{\phi_i}(b_t|s_t, z_{t-1})}{p^T_\theta(b_t|s_t, z_{t-1})}}_{③}\,. \tag{6}$$

As we maximize $\mathbb{E}_q[R^{\text{reg}}_{i,t}]$, this corresponds to maximizing the expectation over $r_i(s_t, a_t) - \beta\big[\mathbb{D}_{\text{KL}}(q^H_{\phi_i} \| p^H) + \mathbb{D}_{\text{KL}}(q^L_{\phi_i} \| p^L_\theta) + \mathbb{D}_{\text{KL}}(q^T_{\phi_i} \| p^T_\theta)\big]$, along the on-policy trajectories drawn from $q_{\phi_i}(\tau)$. Term ① of the regularization encourages exploration in the space of options. It can *also* be seen as a form of deliberation cost (Harb et al., 2017) as it is only nonzero whenever we terminate an option and the master policy needs to select another to execute.

Term ② softly enforces similarity between option posteriors across tasks and updates the prior towards the 'average' posterior. It also encourages the master policy to pick the most *specialized* option whose posteriors across all tasks are most similar. In other words, the master policy $q^H_{\phi_i}$ is encouraged to pick option z_t which maximizes r_i, but minimizes term ② by picking the option z_t for which prior p^L_θ and posterior $q^L_{\phi_i}$ are the most similar. Because the prior is the average posterior, this rewards the master policy to pick the most specialized option (that still achieves high reward). As discussed in more detail in Appendix A, this allows us to escape the local optimization minima that hard options face in multitask learning, while still having fully specialized options after training.

Lastly, we can use ③ to also encourage temporal abstraction of options. To do so, *during option learning*, we fix the termination prior p^T to a Bernoulli distribution $p^T(b) = (1-\alpha)^b \alpha^{1-b}$. Choosing

a large α encourages prolonged execution of one option, but allows switching whenever necessary. This is also similar to deliberation costs (Harb et al., 2017) but with a more flexible cost model.

Additionally, we can still distill a termination prior p_θ^T which can be used on future tasks. Instead of learning p_θ^T by minimizing the KL against the posterior termination policies, we can get more decisive terminations by minimizing

$$\min_\theta \sum_{i=1}^{n} \mathbb{E}_{\tau\sim q^i}\left[\mathbb{D}_{\text{KL}}\big(\hat{q}_{\phi_i}(\cdot|s_t, z_{t-1}) \| \, p_\theta^T(\cdot|s_t, z_{t-1})\big)\right], \tag{7}$$

and $\hat{q}_{\phi_i}(b = 1|s_t, z_{t-1}) = \sum_{z_t \neq z_{t-1}} q_{\phi_i}^H(z_t|s_t, z_{t-1}, b_t = 1)$ i.e., the learned termination prior distills the probability that the tasks' master policies would change the active option if they had the opportunity. Details on how we optimiszed the MSOL objective are given in Appendix C.

4 Related Work

Most hierarchical approaches rely on proxy rewards to train the lower level components. Some of them aim to reach pre-specified subgoals (Sutton et al., 1999; Kulkarni et al., 2016), which are often found by analyzing the structure of the MDP (McGovern & Barto, 2001; Menache et al., 2002; Şimşek et al., 2005; Şimşek & Barto, 2009) or previously learned policies (Goel & Huber, 2003). Those methods typically require knowledge, or a sufficient approximation, of the transition model, both of which are often infeasible.

Recently, several authors have proposed unsupervised training objectives for learning diverse skills based on their distinctiveness (Gregor et al., 2016; Florensa et al., 2017; Achiam et al., 2018; Eysenbach et al., 2019). However, those approaches don't learn termination functions and cannot guarantee that the required behavior on the downstream task is included in the set of learned skills. Hausman et al. (2018) also incorporate reward information, but do not learn termination policies and are therefore restricted to learning multiple solutions to the provided task instead of learning a *decomposition* of the task solutions which can be re-composed to solve new tasks.

A third usage of proxy rewards is by training lower level policies to move towards goals defined by the higher levels. When those goals are set in the original state space (Nachum et al., 2018a), this approach has difficulty scaling to high dimensional state spaces like images. Setting the goals in a learned embedding space (Dayan & Hinton, 1993; Vezhnevets et al., 2017; Nachum et al., 2018b) can be difficult to train, though. In both cases, the temporal extension of the learned skills are set manually. On the other hand, Goyal et al. (2019) also learn a hierarchical agent, but not to transfer skills, but to find decisions states based on how much information is encoded in the latent layer.

HiREPS Daniel et al. (2012) also take an inference motivated approach to learning options. In particular Daniel et al. (2016) propose a similarly structured hierarchical policy, albeit in a single task setting. However, they do not utilize learned prior *and* posterior distributions, but instead use expectation maximization to iteratively infer a hierarchical policy to explain the current reward-weighted trajectory distribution.

Several previous works try to overcome the restrictive nature of options that can lead to sub-optimal solutions by allowing the higher-level actions to modulate the behavior of the lower-level policies Schaul et al. (2015); Heess et al. (2016); Haarnoja et al. (2018). However, this significantly increases the required complexity of the higher-level policy and therefore the learning time.

The multitask- and transfer-learning setup used in this work is inspired by Thrun & Schwartz (1995) and Pickett & Barto (2002) who suggest extracting options by using commonalities between solutions to multiple tasks. Andreas et al. (2017) extend this to deep policies but require additional human supervision in the form of policy sketches. In contrast to our work, which focusses on learning transferrable option priors, Tirumala et al. (2019) investigate the use of different priors for the higher-level policy. Closest to our work is Meta Learning of Shared Hierarchies (MLSH) (Frans et al., 2018) which, however, shares the lower-level policies across all tasks without distinguishing between prior and posterior. As discussed, this leads to local minima and insufficient diversity in the learned options. Similarly to us, Fox et al. (2016) differentiate between prior and posterior policies on multiple tasks and utilize a KL-divergence between them for training. However, they do not consider termination probabilities and instead only choose one option per task.

Our approach is closely related to DISTRAL (Teh et al., 2017) with which we share the multitask learning of prior and posterior policies. However, DISTRAL has no hierarchical structure and applies the same prior distribution over primitive actions, independent of the task. As a necessary hierarchical heuristic, the authors propose to also condition on the last primitive action taken. This works well when the last action is indicative of future behavior; however, in Section 5 we show several failure cases where a *learned* hierarchy is needed.

5 EXPERIMENTS

We conduct a series of experiments to show: i) when learning hierarchies in a multitask setting, MSOL successfully overcomes the local minimum of insufficient option diversity, as described in Appendix A; ii) MSOL can learn useful termination policies; iii) MSOL is equally applicable to discrete as well as continuous domains; and iv) using soft options yields fast transfer learning while still reaching optimal performance, even on new, out-of-distribution tasks.

All architectural details and hyper-parameters can be found in the appendix. For all experiments, we first train the exploration priors and options on n tasks from the available task distribution $\mathcal{T}$ (training phase is plotted in Appendix E). Subsequently, we test how quickly we can learn new tasks from $\mathcal{T}$ (or another distribution $\mathcal{T}'$).

We compare the following algorithms: MSOL is our proposed method that utilizes soft options both during option learning and transfer. MSOL(frozen) uses the soft options framework during learning to find more diverse skills, but does not allow fine-tuning the posterior sub-policies after transfer. DISTRAL (Teh et al., 2017) is a strong non-hierarchical transfer learning algorithm that also utilizes prior and posterior distributions. DISTRAL(+action) utilizes the last action as option-heuristic which works well in some tasks but fails when the last action is not sufficiently informative. MLSH (Frans et al., 2018) is a multitask option learning algorithm like MSOL, but utilizes 'hard' options for both learning and transfer, i.e., sub-policies that are shared exactly across tasks. It also relies on fixed option durations and requires a complex training schedule between master and intra-option policies to stabilize training. We use the author's MLSH implementation. Lastly, we compare against Option Critic (OC) (Bacon et al., 2017), which takes the task-id as additional input in order to apply it to a multitask setting.

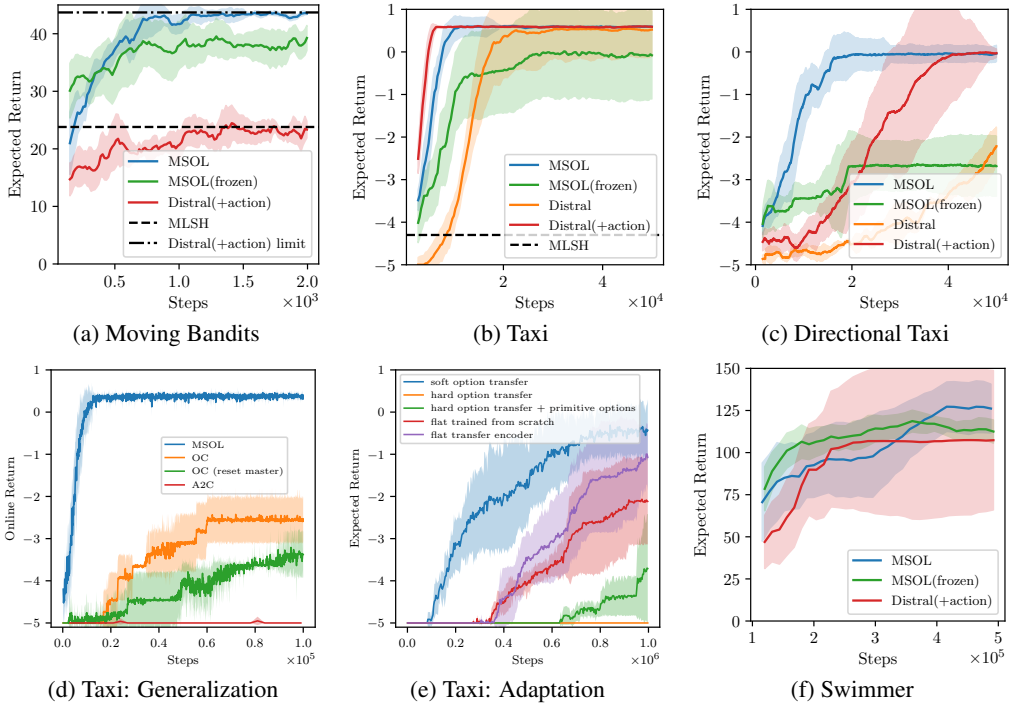
5.1 MOVING BANDITS

We start with the 2D Moving Bandits environment proposed and implemented by Frans et al. (2018), which is similar to the example in Appendix A. In each episode, the agent receives a reward of 1 for each time step it is sufficiently close to one of two randomly sampled, distinguishable, marked positions in the environment. The agent can take actions that move it in one of the four cardinal directions. Which position is not signaled in the observation. Each episode lasts 50 time steps.

We compare against MLSH and DISTRAL to highlight challenges that arise in multitask training. We allow MLSH and MSOL to learn two options. During transfer, optimal performance can only be achieved when both options successfully learned to reach *different* marked locations, i.e., when they are diverse. In Figure 2(a) we can see that MSOL is able to do so but the options learned by MLSH are not sufficiently diverse, for the reason explain in Appendix A. DISTRAL, even with the last action provided as additional input, is not able to quickly utilize the prior knowledge. The last action only conveys meaningful information when taking the goal locations into account: DISTRAL agents need to *infer* the intention based on the last action and the relative goal positions. While this is possible, in practice the agent was not able to do so, even with a much larger network. However, longer training allows DISTRAL to perform as well as MSOL, since its posterior is flexible, denoted by "DISTRAL(+action) limit". Lastly, MSOL(frozen) also outperforms DISTRAL(+action) and MLSH, but performs worse that MSOL. This highlights the utility of making options soft, i.e. adaptable.

5.2 TAXI

Next, we use a slightly modified version of the original Taxi domain (Dietterich, 1998) to show learning of termination functions as well as transfer- and generalization capabilities. To solve the task, the agent must pick up a passenger on one of four possible locations by moving to their location



(a) Moving Bandits (b) Taxi (c) Directional Taxi

(d) Taxi: Generalization (e) Taxi: Adaptation (f) Swimmer

Figure 2: Performance during testing of the learned options and exploration priors. Each line is the median over 5 random seeds (2 for MLSH) and shaded areas indicate standard deviations. More details in text.

and executing a special 'pickup/drop-off' action. Then, the passenger must be dropped off at one of the other three locations, again using the same action executed at the corresponding location. The domain has a discrete state space with 30 locations arranged on a grid and a flag indicating whether the passenger was already picked up. The observation is a one-hot encoding of the discrete state, excluding passenger- and goal location. This introduces an information-asymmetry between the task-specific master policy, and the shared options, allowing them to generalize well (Galashov et al., 2018). Walls (see Figure 3) limit the movement of the agent and invalid actions.

We investigate two versions of Taxi. In the original (Dietterich, 1998, just called *Taxi*), the action space consists of one no-op, one 'pickup/drop-off' action and four actions to move in all cardinal directions. In *Directional Taxi*, we extend this setup: the agent faces in one of the cardinal directions and the available movements are to move forward or rotate either clockwise or counter-clockwise.

In both environments the set of tasks $\mathcal{T}$ are the 12 different combinations of pickup/drop-off locations. Episodes last at most 50 steps and there is a reward of 2 for delivering the passenger to its goal and a penalty of -0.1 for each time step. During training, the agent is initialized to any valid state. During testing, the agent is always initialized without the passenger on board.

We allow four learnable options in MLSH and MSOL. This necessitates the options to be diverse, i.e., one option to reach each of the four pickup/drop-off locations. Importantly, it also requires the options to learn to terminate when a passenger is picked up. As one can see in Figure 2(b), MLSH struggles both with option-diversity and due to its fixed option duration which is not flexible enough for this environment. DISTRAL(+action) performs well in the original *Taxi* environment, as seen in Figure 2(b), since here the last action is a good indicator for the agent's intention. However, in the directional case shown in Figure 2(c), the actions are less informative and make it much harder for DISTRAL to use prior knowledge. By contrast, MSOL performs well in both taxi environments. Comparing its performance with MSOL(frozen) shows the utility of adaptable soft options during transfer.

Figure 3, which visualizes the options learned by MSOL, shows that it successfully learns useful movement primitives and termination functions. The same soft option represents different behavior depending on whether it already picked up the passenger. This is expected as this behavior does not need to terminate the current option on three of the 12 tasks.

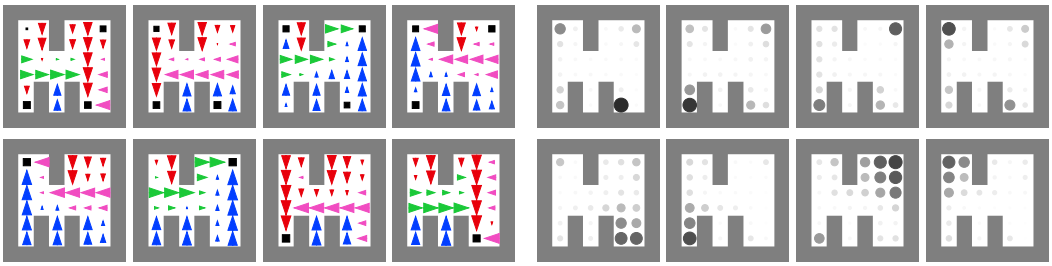


Figure 3: Options learned with MSOL on the taxi domain, before (top) and after pickup (bottom). The light gray area indicates walls. The left plots show the intra-option policies: arrows and colors indicated direction of most likely action, the size indicates its probability. A square indicates the pickup/dropoff action. The right plots show the termination policies: intensity and size of the circles indicate termination probability.

Next we show how learning information-asymmetric soft options can help with transfer to unseen tasks. In Figure 2(d) we show learning on four tasks from $\mathcal{T}$ using options that were trained on the remaining eight, comparing against A2C and OC. Note that in OC, there is no information-asymmetry: We share the same networks across all tasks and provide the task-id as additional input, including to the option-policies. This prevents them from generalizing well to unseen tasks. On the other hand, withholding the task-information from them would be similar to MLSH, which we already showed to struggle with local minima. The strong performance of MSOL on this task shows that we need *soft* options to be able to train information-asymmetric options that generalize well.

We also investigate the utility of flexible soft options: In Figure 2(e) we show learning performance on twelve *changed* tasks in which the pickup/dropoff locations where moved by one cell while the options were trained with the original locations. As expected, hard options are not able to solve this tasks. Even with additional access to primitive actions, exploration is inefficient (Jong et al., 2008). On the other hand, MSOL is able to quickly learn this new task by adapting the previously learned options, outperforming hard options and flat policies learned from scratch.

5.3 SWIMMER

Lastly, we show that MSOL can also be applied to continuous multitask domains. In particular, we investigate the MuJoCo environment 'Swimmer' (Todorov et al., 2012; Brockman et al., 2016). Instead of rewarding forward movement as in the original implementation, now the rewarded movement direction depends on the task from $\mathcal{T} = \{up, down, left, right\}$. We also include a small amount of additive action noise (details in the Appendix). We show that MSOL performs competitive even in the absence of known failure cases of DISTRAL (see Figure 2(f)).

6 DISCUSSION

Multitask Soft Option Learning (MSOL) proposes reformulating options using the perspective of prior and posterior distributions. This offers several key advantages.

First, during transfer, it allows us to distinguish between fixed, and therefore knowledge-preserving option *priors*, and flexible option *posteriors* that can adjust to the reward structure of the task at hand. This effects a similar speed-up in learning as the original options framework, while avoiding sub-optimal performance when the available options are not perfectly aligned to the task. Second, utilizing this 'soft' version of options in a multitask learning setup increases optimization stability and removes the need for complex training schedules between master and lower level policies. Furthermore, this framework naturally allows master policies to coordinate across tasks and avoid local minima of insufficient option diversity. It also allows for autonomously learning option-termination policies, a very challenging task which is often avoided by fixing option durations manually.

Lastly, using this formulation also allows inclusion of prior information in a principled manner without imposing too rigid a structure on the resulting hierarchy. We utilize this advantage to explicitly incorporate the bias that good options should be temporally extended. In future research, other types of information can be explored. As an example, one could investigate sets of tasks which would benefit from a learned master prior, like walking on different types of terrain.

REFERENCES

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pp. 166–175, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pp. 17–36, 2012.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Matthew M Botvinick, Yael Niv, and Andrew C Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

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

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Anne Gabrielle Eva Collins and Michael Joshua Frank. Neural signature of hierarchically structured expectations predicts clustering and transfer of rule sets in reinforcement learning. *Cognition*, 152:160–169, 2016.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pp. 273–281, 2012.

Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2-3):337–357, 2016.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Thomas G Dietterich. The maxq method for hierarchical reinforcement learning. In *ICML*, volume 98, pp. 118–126. Citeseer, 1998.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=SJx63jRqFm`.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Roy Fox, Michal Moshkovitz, and Naftali Tishby. Principled option learning in markov decision processes. *arXiv preprint arXiv:1609.05524*, 2016.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. META LEARNING SHARED HIERARCHIES. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=SyX0IeWAW`.

Alexandre Galashov, Siddhant Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojtek M Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in kl-regularized rl. 2018.

Sandeep Goel and Manfred Huber. Subgoal discovery for hierarchical reinforcement learning using learned policies. In *FLAIRS conference*, pp. 346–350, 2003.

Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJg8yhAqKm.

Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *arXiv preprint arXiv:1802.07245*, 2018.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018.

Jean Harb, Pierre-Luc Bacon, Martin Klissarov, and Doina Precup. When waiting is not an option: Learning options with a deliberation cost. *arXiv preprint arXiv:1709.04571*, 2017.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Rein Houthooft, Richard Y Chen, Phillip Isola, Bradly C Stadie, Filip Wolski, Jonathan Ho, and Pieter Abbeel. Evolved policy gradients. *arXiv preprint arXiv:1802.04821*, 2018.

Nicholas K Jong, Todd Hester, and Peter Stone. The utility of temporal abstraction in reinforcement learning. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 1*, pp. 299–306. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 489–496. ACM, 2006.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr, 2018.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pp. 3675–3683, 2016.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. Time regularized interrupting options. In *Internation Conference on Machine Learning*, 2014.

Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pp. 361–368, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-778-1. URL http://dl.acm.org/citation.cfm?id=645530.655681.

Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-cutdynamic discovery of sub-goals in reinforcement learning. In *European Conference on Machine Learning*, pp. 295–306. Springer, 2002.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Ofir Nachum, Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018a.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018b.

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

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pp. 1043–1049, 1998.

Marc Pickett and Andrew G Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *ICML*, pp. 506–513, 2002.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

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, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Özgür Şimşek and Andrew G Barto. Skill characterization based on betweenness. In *Advances in neural information processing systems*, pp. 1497–1504, 2009.

Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 816–823. ACM, 2005.

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

Matthew E Taylor and Peter Stone. An introduction to intertask transfer for reinforcement learning. *Ai Magazine*, 32(1):15, 2011.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4499–4509, 2017.

Sebastian Thrun and Anton Schwartz. Finding structure in reinforcement learning. In *Advances in neural information processing systems*, pp. 385–392, 1995.

Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

Emanuel Todorov. General duality between optimal control and estimation. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pp. 4286–4292. IEEE, 2008.

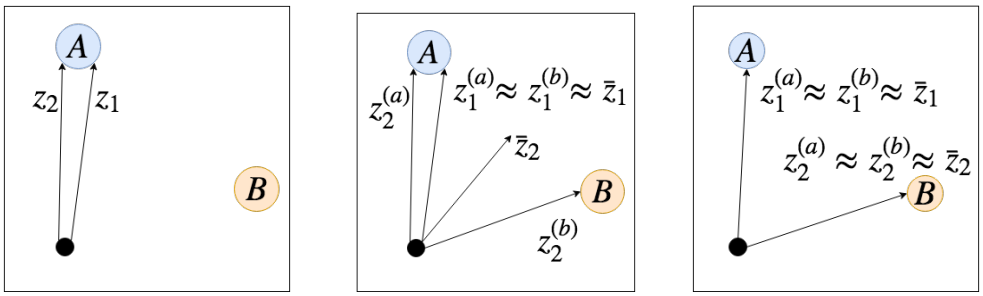
Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th annual international conference on machine learning*, pp. 1049–1056. ACM, 2009.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549, 2017.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Tianbing Xu, Qiang Liu, Liang Zhao, and Jian Peng. Learning to explore via meta-policy gradient. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5463–5472, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR.



(a) Local minimum (b) Using skill priors (c) After training

Figure 4: Hierarchical learning of two concurrent tasks (a and b) using two options (z_1 and z_2) to reach two relevant targets (A and B). a) Local minimum when simply sharing options across tasks. b) Escaping the local minimum by using prior ($\bar{z}_i$) and posterior ($z_i^{(j)}$) policies. c) Learned options after training. Details are given in the text in Appendix A.

Appendix

A Local Minima in Multitask Option Learning

Here our aim is to provide an intuitive explanation of why learning hard options in a multitask setting can lead to local minima. In this local minimum, multiple options have learned the same behavior and are unable to change it, even if doing so would ultimately lead to a higher reward. We use the Moving Bandits experiment schematically depicted in Figure 4 as example: The agent (black dot) observes two target locations A and B but does not know which one is the correct one that has to be reached in order to generate a reward.

The state- and action-spaces are continuous, requiring multiple actions to reach either A or B from the starting position. Consequently, having access to two options, one for each location, can accelerate learning. We denote the options by z_1 and z_2.

Assume that due to random initialization or later discovery of target B, both skills currently reach A. Furthermore, assume that we are training on multiple task *simultaneously* as often the case in RL. In this situation, the master policies on tasks in which the correct goal is A are indifferent between using z_1 and z_2 and will consequently use *both* with high probability.

Now, changing one skill, e.g. z_2, towards B in order to solve tasks in which B is the correct target, decreases the performance on all tasks that currently use z_2 to reach target A because for hard options the skills are shared exactly across tasks. Due to small, stochastic gradient updates changing z_2 towards B only slightly increases the probability of reaching B but significantly decreases the probability of reaching A (because now the option z_2 will at first point to neither A or B). Averaged across all tasks, this would at first *decrease* the average reward, consequently preventing any option from changing away from A, leaving B unreachable.

To "free up" z_2 (or any other option) and learn a new skill, all master policies would need to coordinate and refrain from using it to reach A and instead use the equally useful skill z_1 exclusively. Importantly, using soft options makes this possible.

In Figures 4(b) and 4(c) we depict this schematically: The key difference is that in MSOL we have separate *task-specific posteriors* $z_i^{(a)}$ and $z_i^{(b)}$ for tasks a and b and soft options $i \in \{1, 2\}$ (for simplicity, we assume that the correct target is A for task a and B for task b). This allows us, in a first step, to solve *all* tasks (Figure 4(b)). Later, as discussed in Section 3.3, maximizing the regularized reward (6) will lead to specialized skills, i.e. option *priors* (Figure 4(c)).

The corresponding shared priors $\bar{z}_i$ (one per option) are updated to effectively yield the 'average' (as measured by the KL-divergence) between the task-specific posteriors $z_i^{(a)}$ and $z_i^{(b)}$. Allowing the posteriors of an option, for example $z_2^{(a)}$ and $z_2^{(b)}$, to deviate from the prior $\bar{z}_2$ allows that option, in a first step, to solve *both* tasks, irrespective of the misaligned priors. This is shown in Figure 4(b)

where all posteriors of z_1 are moving to A, but for z_2, the posteriors on different tasks a and b move to different targets.

Crucially, to maximize the regularized reward (6), the KL-divergences between priors and posteriors should be minimized along they trajectories, i.e. *weighted by how likely they are to occur*. Consequently, the higher $\mathbb{D}_{\text{KL}}[z_2^{(a)} \| \bar{z}_2] > 0$ can be avoided on task a by picking the more specialized option z_1 for which $\mathbb{D}_{\text{KL}}[z_1^{(a)} \| \bar{z}_1] = 0$. This, in turn, "frees up" z_2 to now also specialize to B as shown in Figure 4(c). After this implicit coordination, $z_1^{(a)}$ is always used to reach A and $z_2(b)$ is always used to reach B, allowing both $\bar{z}_1$ and $\bar{z}_2$ to also lead to A and B respectively.

B Relationship between MSOL and classical options

Assume we are faced with a new task and are given some prior knowledge in the form of a set of skills that we can use. Using those skills and their termination probabilities as prior policies p^T and p^L in the soft option framework, we can see β as a temperature parameter determining how closely we are restricted to following them. For $\beta \to \infty$ we recover the classical 'hard' option case and our posterior option policies are restricted to the prior.[1] For $\beta = 0$ the priors only initialize the otherwise unconstrained policy, quickly unlearning behavior that may be useful down the line. Lastly, for $0 < \beta < \infty$ we use the prior information to guide exploration but are only softly restricted to the given skills and can also explore and use policies 'close' to them.

C MSOL training details

C.1 Optimization

Even though R_i^{reg} depends on ϕ_i, its gradient w.r.t. ϕ_i vanishes.[2] Consequently, we can treat the regularized reward as a classical RL reward and use any RL algorithm to find the optimal hierarchical policy parameters ϕ_i. In the following, we explain how to adapt A2C (Mnih et al., 2016) to soft options. The extension to PPO (Schulman et al., 2017) is straightforward.[3]

The joint posterior policy in (4) depends on the current state s_t and the previously selected option z_{t-1}. The expected sum of regularized future rewards of task i, the value function V_i, must therefore also condition on this pair:

$$V_i(s_t, z_{t-1}) \;:=\; \mathbb{E}_{\tau \sim q}\left[\sum_{t'=t}^{T} \gamma^{t'-t} R_{i,t'}^{\text{reg}} \,\middle|\, s_t, z_{t-1} \right]. \tag{8}$$

As $V_i(s_t, z_{t-1})$ cannot be directly observed, we approximate it with a parametrized model $V_{\phi_i}(s_t, z_{t-1})$. The k-step advantage estimation at time t of trajectory τ is given by

$$A_{\phi_i}(\tau_{t:(t+k)}) \quad := \quad \sum_{j=0}^{k-1} \gamma^j R_{t+j}^{\text{reg}} + \gamma^k V_{\phi_i}^{-}(s_{t+k}, z_{t+k-1}) - V_{\phi_i}(s_t, z_{t-1}), \tag{9}$$

where the superscript '$-$' indicates treating the term as a constant. The approximate value function V_{ϕ_i} can be optimized towards its bootstrapped k-step target by minimizing $\mathcal{L}_V(\phi_i, \tau_{1:T}) := \sum_{t=1}^{T} (A_{\phi_i}(\tau_{t:(t+k)}))^2$. As per A2C, $k \in [1 \dots n_s]$ depending on the state (Mnih et al., 2016). The corresponding policy gradient loss is

$$\mathcal{L}_A(\phi_i, \tau_{1:T}) := \sum_{t=1}^{T} A_{\phi_i}^{-}(\tau_{t:(t+k)}) \ln q_{\phi_i}(a_t, z_t, b_t | s_t, z_{t-1}).$$

The gradient w.r.t. the prior parameters θ is[4]

$$\nabla_\theta \mathcal{L}_P(\theta, \tau_{1:T}, \tilde{b}_{1:T}) \quad := \quad -\sum_{t=1}^{T} \Big(\nabla_\theta \ln p_\theta^L(a_t | s_t, z_t) + \nabla_\theta \ln p_\theta^T(\tilde{b}_t | s_t, z_{t-1}) \Big), \tag{10}$$

[1] However, in this limiting case optimization using the regularized reward is not possible.

[2] $\int p(x) \nabla \ln p(x)\, dx = \int \nabla p(x)\, dx = \nabla \int p(x)\, dx = 0$.

[3] However, for PAI frameworks like ours, unlike in the original PPO implementation, the advantage function must be updated after each epoch.

[4] Here we ignore β as it is folded into λ_P later.

where $\tilde{b}_t = \delta_{z_{t-1}}(z'_t)$ and $z'_t \sim q^H(z'_t|s_t, z_{t-1}, b_t = 1)$. To encourage exploration in all policies of the hierarchy, we also include an entropy maximization loss:

$$\mathcal{L}_H(\phi_i, \tau_{1:T}) \quad := \quad \sum_{t=1}^{T} \Big(\ln q^H_{\phi_i}(z_t|s_t, z_{t-1}, b_t) + \ln q^L_{\phi_i}(a_t|s_t, z_t) + \ln q^T_{\phi_i}(b_t|s_t, z_{t-1}) \Big) . \quad (11)$$

Note that term ① in (6) already encourages maximizing $\mathcal{L}_H(\phi_i, \tau)$ for the master policy, since we chose a uniform prior $p^H(z_t|b_t = 1)$. As both terms serve the same purpose, we are free to drop either one of them. In our experiments, we chose to drop the term for q^H in R^{reg}_t, which proved slightly more stable to optimize that the alternative.

We can optimize all parameters jointly with a combined loss over all tasks i, based on sampled trajectories $\tau^i := \tau^i_{1:T} \sim q_{\phi_i}$ and corresponding sampled values of $\tilde{b}^i := \tilde{b}^i_{1:T}$:

$$\mathcal{L}(\{\phi_i\}, \theta, \{\tau^i\}, \{\tilde{b}^i\}) = \sum_{i=1}^{n} \Big(\mathcal{L}_A(\phi_i, \tau^i) + \lambda_V \mathcal{L}_V(\phi_i, \tau^i) + \lambda_P \mathcal{L}_P(\theta, \tau^i, \tilde{b}^i) + \lambda_H \mathcal{L}_H(\phi_i, \tau^i) \Big) .$$

C.2 Training schedule

For faster training, it is important to prevent the master policies q^H from converging too quickly to allow sufficient updating of all options. On the other hand, a lower exploration rate leads to more clearly defined options. We consequently anneal the exploration bonus λ_H with a linear schedule during training.

Similarly, a high value of β leads to better options but can prevent finding the extrinsic reward $r_i(s_t, a_t)$ early on in training. Consequently, we increase β over the course of training, also using a linear schedule.

D Architecture

All policies and value functions share the same encoder network with two fully connected hidden layers of size 64 for the Moving Bandits environment and three hidden layers of sizes 512, 256, and 512 for the Taxi environments. Distral was tested with both model sizes on the Moving Bandits task to make sure that limited capacity is not the problem. Both models resulted in similar performance, the results shown in the paper are for the larger model. On swimmer the encoder model size is $1024 \times 256 \times 512$. Master-policies, as well as all prior- and posterior policies and value functions consist of only one layer which takes the latent embedding produced by the encoder as input. Furthermore, the encoder is shared across tasks, allowing for much faster training since observations can be batched together.

Options are specified as an additional one-hot encoded input to the corresponding network that is passed through a single 128 dimensional fully connected layer and concatenated to the state embedding before the last hidden layer. We implement the single-column architecture of Distral as a hierarchical policy with just one option and with a modified loss function that does not include terms for the master and termination policies. Our implementation builds on the A2C/PPO implementation by Kostrikov (2018), and we use the implementation for MLSH that is provided by the authors (`https://github.com/openai/mlsh`).

E Hyper-parameters and additional environment details

We use $2\lambda_V = \lambda_A = \lambda_P = 1$ in all experiments. Furthermore, we train on all tasks from the task distribution, regularly resetting individual tasks by resetting the corresponding master and re-initializing the posterior policies. Optimizing β for MSOL and Distral was done over $\{0.01, 0.02, 0.04, 0.1, 0.2, 0.4\}$. We use $\gamma = 0.95$ for Moving Bandits and Taxi and $\gamma = 0.995$ for Swimmer.

E.1 Moving Bandits

For MLSH, we use the original hyper-parameters (Frans et al., 2018). The duration of each option is fixed to 10. The required warm-up duration is set to 9 and the training duration set to 1. We also

use 30 parallel environments split between 10 tasks. This and the training duration are the main differences to the original paper. Originally, MLSH was trained on 120 parallel environments which we were unable to do due to hardware constraints. Training is done over 6 million frames per task.

For MSOL and Distral we use the same number of 10 tasks and 30 processes. The duration of options are learned and we do not require a warm-up period. We set the learning rate to 0.01 and $\beta = 0.2$, $\alpha = 0.95$, $\lambda_H = 0.05$. Training is done over 0.6 million frames per task. For Distral we use $\beta = 0.04$, $\lambda_H = 0.05$ and also 0.6 million frames per task.

E.2 TAXI

For MSOL we anneal β from 0.02 to 0.1 and λ_H from 0.1 to 0.05. For Distral we use $\beta = 0.04$. We use 3 processes per task to collect experience for a batch size of 15 per task. Training is done over 1.4 million frames per task for *Taxi* and 4 million frames per task for *Directional Taxi*. MLSH was trained on 0.6 million frames for *Taxi* as due to it's long runtime of several days, using more frames was infeasible. Training was already converged.

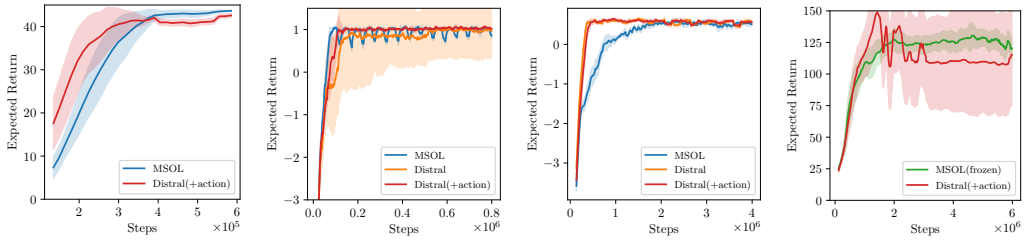
E.3 TAXI: ADAPTATION TO CHANGED TASKS

As shown in the main experiments, soft options can still be useful, even when the task distribution changes. It is unsurprising that hard options are not able to solve the task. However, interestingly, providing hard options *and* primitive actions to the master policy performs much worse than just learning from scratch. This phenomenon was investigated previously by Jong et al. (2008) and further justifies using soft options for transfer to out-of-distribution tasks.

Whether training from scratch or re-using misspecified options that were trained on a different set of tasks is learning faster mainly depends on i) how strongly the options are misspecified and ii) how difficult the exploration problem is in the environment. This tradeoff is shown in Figure 6: On a smaller 8x8 grid (left), learning from scratch performs competitively because exploration is sufficiently simple. On the other hand, on a 10x10 grid (right, results from the main paper), exploration is harder and soft options allow for significantly faster learning because they can guide the exploration in a helpful way.

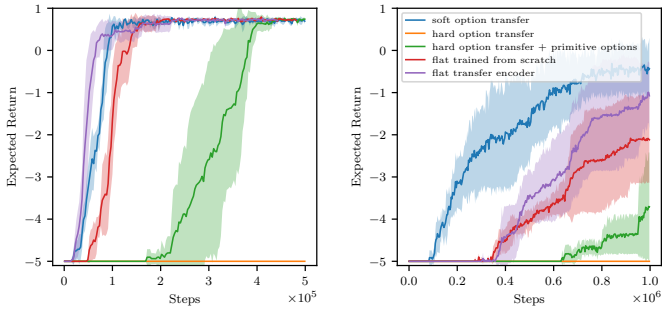
E.4 SWIMMER

For training Distral and MSOL we use PPO instead of A2C as it generally achieves better performance on continuous tasks. We have $\lambda_H = 0.0004$ for both MSOL and Distral for primitive actions and $\lambda'_H = 0.1$ for the master- and termination policies in MSOL. We use a learning rate of 0.0002, GAE (Schulman et al., 2015) with $\tau = 0.98$. We collect 2000 steps in parallel on 6 processes per task, resulting in a batchsize of $12,000$ per task. Training is done over 6 million frames with a linearly scheduled increase of β from 0 to 0.04 for MSOL and 0.01 for Distral. We set $\alpha = 0.98$.



(a) Moving Bandits (b) Taxi (c) Directional Taxi (d) Swimmer

Figure 5: Performance during training phase. Note that MSOL and MSOL(frozen) share the same training as they only differ during testing. Further, note that the highest achievable performance for Taxi and Directional Taxi is higher during training as they can be initialized closer to the final goal (i.e. with the passenger on board).



(a) Grid of size 8x8

(b) Grid of size 10x10

Figure 6: Tradeoff between exploration and misspecification: For smaller grid sizes, exploration is easier and learning from scratch performs competitively. For larger grid sizes, soft options can accelerate training through faster exploration, even if they are misspecified because they were trained on a different set of tasks.