Hindsight Task Relabelling: Experience Replay for Sparse Reward Meta-RL

Anonymous Author(s)
Affiliation
Address
email

Abstract

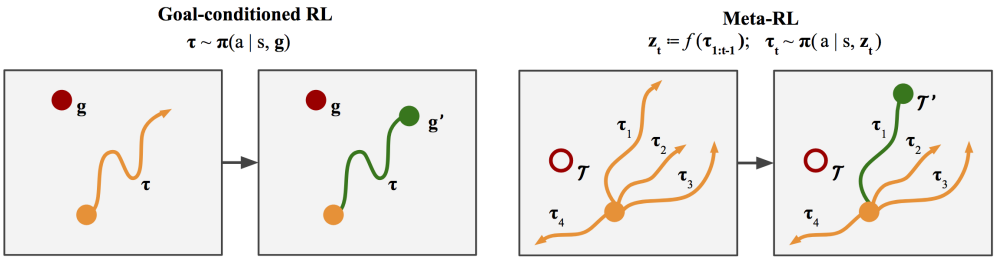
Meta-reinforcement learning (meta-RL) has proven to be a successful framework for leveraging experience from prior tasks to rapidly learn new related tasks, however, current meta-RL approaches struggle to learn in sparse reward environments. Although existing meta-RL algorithms can learn strategies for adapting to new sparse reward tasks, the actual adaptation strategies are learned using hand-shaped reward functions, or require simple environments where random exploration is sufficient to encounter sparse reward. In this paper, we present a formulation of hindsight relabeling for meta-RL, which relabels experience during meta-training to enable learning to learn entirely using sparse reward. We demonstrate the effectiveness of our approach on a suite of challenging sparse reward goal-reaching environments that previously required dense reward during meta-training to solve. Our approach solves these environments using the true sparse reward function, with performance comparable to training with a separate dense reward function.

Reinforcement learning (RL) has seen tremendous success applied to challenging games (Mnih et al., 2015; Silver et al., 2017) and robotic control (Lillicrap et al., 2015; Levine et al., 2016), driven by advances in compute and the use of deep neural networks as powerful function approximators in RL algorithms. However, agents trained using deep RL often struggle to meaningfully utilize past experience to learn new tasks, even if the new tasks differ only slightly from tasks seen during training (Zhang et al., 2018; Packer et al., 2018; Cobbe et al., 2019). In contrast, humans are adept at utilizing prior experience to rapidly acquire new skills and adapt to unseen environments.

Meta-reinforcement learning (meta-RL) aims to address this limitation by extending the RL framework to explicitly consider structured distributions of tasks (Schmidhuber, 1987; Bengio et al., 1990; Thrun & Pratt, 1998). Whereas conventional RL is concerned with learning a single task, meta-RL is concerned with *learning to learn*, i.e., learning how to quickly learn a new task by leveraging prior experience on related tasks. Meta-RL methods generally utilize a limited amount of experience in a new environment to estimate a latent task embedding which conditions the policy, or to compute a policy gradient which is used to directly update the parameters of the policy.

A major challenge in both RL and meta-RL is learning with sparse rewards. When rewards are sparse or delayed, the adaptation stage in meta-RL becomes extremely difficult: inferring the task at hand requires receiving reward signal from the environment, which in the sparse reward setting only happens after successfully completing the task. Due to this inherent incompatibility between meta-RL and sparse rewards, existing meta-RL algorithms that consider the sparse reward setting either only work in simple environments that do not require temporally-extended exploration strategies for adaptation (Duan et al., 2016; Stadie et al., 2018), or train exclusively using dense reward functions (Gupta et al., 2018b; Rakelly et al., 2019), which are designed to encourage an agent to learn adaptation strategies that can be directly applied to the original sparse reward setting.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.



(a) Hindsight relabeling in goal-conditioned RL (b) Hindsight Task Relabeling (HTR) in meta-RL

Figure 1: In goal-conditioned RL (a), an agent must navigate to a provided goal location $\mathbf{g}$ (filled circle, revealed to the agent). An unsuccessful attempt for goal $\mathbf{g}$ provides no sparse reward signal, but can be relabelled as a successful attempt for goal $\mathbf{g}'$, creating sparse reward that can be used to train the agent. In meta-RL (b), the task $\mathcal{T}$ (i.e., goal, hollow circle) is never revealed to the agent, and instead must be inferred using experience on prior tasks and limited experience ($\tau_{1:t-1}$) on the new task. In (b), there is no shared optimal task $\mathcal{T}'$ to relabel all attempts with. HTR relabels each attempt τ under its own hindsight task $\mathcal{T}'$, and modifies the underlying meta-RL training loop to learn adaptation strategies on the relabelled tasks. Note that we include multiple trajectories τ in (b) vs a single trajectory in (a) to highlight the adaptation stage in meta-RL, which does not exist in goal-conditioned RL and requires significantly different sampling and relabeling procedures.

In this paper, we show that the concept of *hindsight relabeling* (Andrychowicz et al., 2017) from conventional RL can be applied in meta-RL to enable learning to learn in the sparse reward setting. Our key insight is that data collected on the true training tasks can be relabelled as pseudo-expert data for easier hindsight tasks, bootstrapping meta-training with the reward signal needed to train the agent. We introduce a new algorithm for off-policy meta-RL, which we call *Hindsight Task Relabeling* (HTR), and demonstrate its effectiveness by achieving state-of-the-art performance on a collection of challenging sparse reward environments that previously required shaped reward to solve. Not only is our approach able to learn adaptation strategies only using sparse reward, but it also learns strategies with comparable performance to existing approaches that use shaped reward functions.

1 Related Work

In meta-RL, an agent learns an adaptation strategy by repeatedly adapting to training tasks in the inner loop of *meta-training*, with the hope that the learned adaptation procedure will generalize to new, unseen tasks during *meta-testing*. Context-based methods use recent experience (i.e., context, in the form of trajectories or transitions) in a new task to estimate a latent task embedding, and differ mainly in how this embedding is computed: Duan et al. (2016); Wang et al. (2016); Mishra et al. (2018) propose aggregating context into the hidden state of the policy, whereas Rakelly et al. (2019) propose explicitly feeding the task embedding as input to the policy. Gradient-based methods use recent context to update differentiable hyperparameters (Xu et al., 2018b), loss functions (Sung et al., 2017; Houthooft et al., 2018), or to directly update policy parameters (Finn et al., 2017; Stadie et al., 2018; Xu et al., 2018a; Zintgraf et al., 2018; Gupta et al., 2018b; Rothfuss et al., 2019).

Many of the aforementioned approaches struggle to learn effective exploration strategies for tasks with sparse or delayed rewards. Methods that directly update the policy either implicitly (e.g., with a hidden state) or explicitly (e.g., with gradients) are generally unable to learn a policy that meaningfully explores, since the primary source of randomness is action-space, and thus is time-invariant. Gupta et al. (2018b) and Rakelly et al. (2019) propose using a probabilistic task variable that is sampled once per episode, which makes the primary source of variability task-dependent and enables temporally-extended exploration. While this approach enables effective adaptation in the sparse reward setting, both Gupta et al. (2018b) and Rakelly et al. (2019) learn the actual adaptation strategies using shaped (dense) reward functions during meta-training. In the environments they consider, the adaptation strategies learned using dense rewards generalize to sparse rewards despite the difference in reward structure; however, engineering a suitable reward function can be costly (Ng et al., 1999) and error prone (Clark & Amodei, 2016), and thus it is highly desirable to devise a mechanism for learning to learn in the sparse reward setting that does not require an auxiliary dense reward function.

Our proposed method is closely related to prior work in unsupervised meta-RL and goal generation. Unsupervised meta-RL (Jabri et al., 2019; Gupta et al., 2018a) considers the meta-RL setting without access to a training task distribution; training tasks are instead self-generated with the aim of learning skills useful for the test task distribution. Our proposed method also generates a curriculum of training tasks, but with the intent of learning adaptation strategies in the sparse reward setting, as opposed to learning transferable skills in a setting with no rewards at all. Several methods exist for curriculum generation in the context of goal-conditioned policies, including adversarial goal generation (Florensa et al., 2018) and goal relabeling (Kaelbling, 1993; Andrychowicz et al., 2017; Levy et al., 2017; Nair et al., 2018). Recent work (Li et al., 2020; Eysenbach et al., 2020) has proposed the use of inverse RL to generalize goal relabeling to broader families of tasks. Unlike prior work on goal and task relabeling that use goal- or task-conditioned policies, we focus on the meta-RL setting where the task is unknown to the policy and must be inferred through deliberate and coherent exploration.

2 Background

In this section we formalize the meta-RL problem setting and briefly describe two algorithms which our approach, Hindsight Task Relabeling, builds on: Probabilistic Embeddings for Actor-critic RL (PEARL) and Hindsight Experience Replay (HER).

2.1 Meta-Reinforcement Learning (Meta-RL)

Conventional RL assumes environments are modeled by a Markov Decision Processes (MDP) M, defined by the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma, \rho_0, T)$ where $\mathcal{S}$ is the set of possible states, $\mathcal{A}$ is the set of actions, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the transition probability density, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, γ is the discount factor, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the initial state distribution at the beginning of each episode, and T is the time horizon of an episode (Sutton & Barto, 2017).

Let $\mathbf{s}_t$ and $\mathbf{a}_t$ be the state and action taken at time t. At the beginning of each episode, $\mathbf{s}_0 \sim \rho_0(\cdot)$. Under a stochastic policy π mapping a sequence of states to actions, $\mathbf{a}_t \sim \pi(\mathbf{a}_t \mid \mathbf{s}_t, \cdots, \mathbf{s}_0)$ and $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$, generating a trajectory $\tau = \{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t)\}$, where $t = 0, 1, \cdots$. Conventional RL algorithms, which assume the environment is fixed, learn π to maximize the expected reward per episode $J_M(\pi) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{T} \gamma^t \mathbf{r}_t\right]$, where $\mathbf{r}_t = r(\mathbf{s}_t, \mathbf{a}_t)$. They often utilize the concepts of a value function $V_M^{\pi}(\mathbf{s})$, the expected reward conditional on $\mathbf{s}_0 = \mathbf{s}$ and a state-action value function $Q_M^{\pi}(\mathbf{s}, \mathbf{a})$, the expected reward conditional on $\mathbf{s}_0 = \mathbf{s}$ and $\mathbf{a}_0 = \mathbf{a}$.

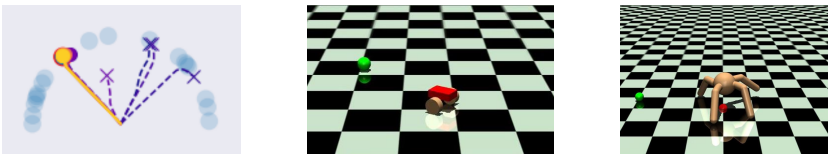
Meta-RL algorithms assume that there is a distribution of tasks $p(\mathcal{T})$, and usually maximize the expected reward over the distribution, $\mathbb{E}^{\pi}_{\mathcal{T} \sim p(\mathcal{T})}\left[J_{\mathcal{T}}(\pi)\right]$. The distribution of tasks corresponds to a distribution of MDPs $M(\mathcal{T})$, where $\mathcal{T}$ can paramaterize an MDP's dynamics or reward. In this work we focus on the latter case, where each $\mathcal{T}$ implies an MDP with a different reward function under the same dynamics. The agent is trained on a set of training tasks $\mathcal{T}_{train} \sim p(\mathcal{T})$ during meta-training, and evaluated on a held-out set of testing tasks $\mathcal{T}_{test} \sim p(\mathcal{T})$ during meta-testing.

An important distinction between meta-RL and multi-task or goal-conditioned RL is that in meta-RL, the task is never explicitly revealed to the agent or policy π through observations $\mathbf{s}$. Instead, π must explore the environment to infer the task to maximize expected reward. This crucial difference is one reason why Hindsight Experience Replay cannot be directly applied to the meta-RL setting.

2.2 Off-Policy Meta-Reinforcement Learning

We demonstrate the effectiveness of our approach by building on top of Probabilistic Embeddings for Actor-critic RL (PEARL) by Rakelly et al. (2019), an off-policy meta-RL algorithm which itself is built on top of soft actor-critic (SAC) by Haarnoja et al. (2018). SAC is an actor-critic algorithm based on the maximum entropy RL objective that optimizes a stochastic policy π with off-policy data. In addition to the policy (actor) π, SAC concurrently learns twin state-action value functions (critics) Q. Transitions collected using the policy π are added to the replay buffer B, and Q and π are optimized using loss functions regularized by the entropy of π.

Rakelly et al. (2019) extend SAC to the meta-RL setting by adding a context encoder that uses recent history in an environment to estimate a task embedding, which is then utilized by the policy.



(a) Point Robot (b) Wheeled Locomotion (c) Ant Locomotion

Figure 2: Sparse reward environments for meta-RL that require temporally-extended exploration. In each environment, the task (i.e., the top-left circle in (a), the green sphere in (b) and (c)) is not revealed to the agent via the observation. The agent must instead infer the task through temporally-extended exploration (illustrated by the dotted lines in (a)), since no reward signal is provided until the task is successfully completed. Prior meta-RL methods such as PEARL (Rakelly et al., 2019) and MAESN (Gupta et al., 2018b) are only able to (meta-)learn meaningful adaptation strategies using dense reward functions. Our approach, Hindsight Task Relabeling (HTR), can (meta-)train with the original sparse reward function and does not require additional dense reward functions.

Specifically, PEARL uses a network $q_\phi(\mathbf{z} \mid \mathbf{c})$ that takes as input recently collected data (i.e., context $\mathbf{c}$) to infer a probabilistic latent context variable Z. Samples from the latent context variable condition the actor, $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$, and critic, $Q_\theta(\mathbf{s}, \mathbf{a}, \mathbf{z})$. Instead of a single replay buffer (as in SAC), there are T separate buffers $B_{i=1...T}$, corresponding to the number of tasks sampled for $\mathcal{T}_{train}$. The separate per-task replay buffers enable sampling off-policy (but on-*task*) context during the inner loop of meta-training, which significantly improves data efficiency.

During meta-training, transitions are collected for each B_i by sampling $\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{c})$ and acting according to $\pi_\theta(\mathbf{a} \mid \mathbf{s}, \mathbf{z})$. $\mathbf{z}$ is periodically sampled during data collection such that the context contains transitions collected using the policy conditioned on the prior, as well the same policy conditioned on the posterior. Gradients used to optimize q_ϕ, π_θ and Q_θ are computed in task-specific batches across a random subset of training tasks $\mathcal{T}_i \sim p(\mathcal{T})$, each with an associated replay buffer B_i. During meta-testing, for each test task $\mathcal{T}_i \sim p(\mathcal{T})$, an empty buffer B_i is initialized, $\mathbf{z}$ is sampled from the prior, and the policy conditioned on $\mathbf{z}$ is rolled out. As additional context is added to the replay buffer, the belief over $\mathbf{z}$ narrows, enabling the policy to adapt to the task at hand.

2.3 Hindsight Experience Replay

Hindsight Experience Replay (HER) (Andrychowicz et al., 2017) is a method for off-policy goal-conditioned RL in environments with sparse reward. The key insight behind HER is that unsuccessful attempts (i.e., attempts that received no reward signal) generated via a goal-conditioned policy can be relabelled as successful attempts for a 'hindsight' goal that was actually achieved, e.g., the final state (see Figure 1). HER can be applied to any off-policy RL algorithm that uses goal-conditioned policies (where the actor and/or critic receive the desired goal as part of the state, i.e., $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{g})$ and $Q(\mathbf{s}, \mathbf{a}, \mathbf{g})$). When a transition $\{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{g}) \mid \mathbf{g}\}$ is sampled from the replay buffer during training, with some probability HER rewrites the desired goal $\mathbf{g}$ with an achieved goal $\mathbf{g}'$ and recomputes the reward under the new goal. The modified transition $\{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{g}') \mid \mathbf{g}'\}$ is used to optimize $\pi(\mathbf{a} \mid \mathbf{s}, \mathbf{g}')$ and $Q(\mathbf{s}, \mathbf{a}, \mathbf{g}')$. Hindsight relabeling generates an implicit curriculum: initially, sampled transitions are rewritten with 'easy' goals achievable with random policies, and as training progresses, relabelled goals are increasingly likely to be near the desired goals.

3 Leveraging Hindsight in Meta-Reinforcement Learning

Our algorithm, Hindsight Task Relabeling, applies ideas from hindsight relabeling to the meta-RL setting to enable learning adaptation strategies for tasks with sparse or delayed rewards. Similar to how policies in HER are conditioned on a goal $\mathbf{g}$, policies in context-based meta-RL are conditioned on a latent task embedding $\mathbf{z}$. A crucial difference in meta-RL is that the task is hidden from the agent: the agent must infer relevant features of the task at hand using experience on prior tasks and a limited amount of experience gathered in the environment. Our key insight is that an unsuccessful experience collected in an *unknown* task can be relabelled as a successful experience for a *known* hindsight task, i.e., a transition $\{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t, \mathcal{T})\}$ generated under an unknown $\mathcal{T}$ can be rewritten under a known $\mathcal{T}'$ as $\{\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t, \mathcal{T}')\}$, regardless of whether $\mathcal{T}' \in \mathcal{T}_{train}$.

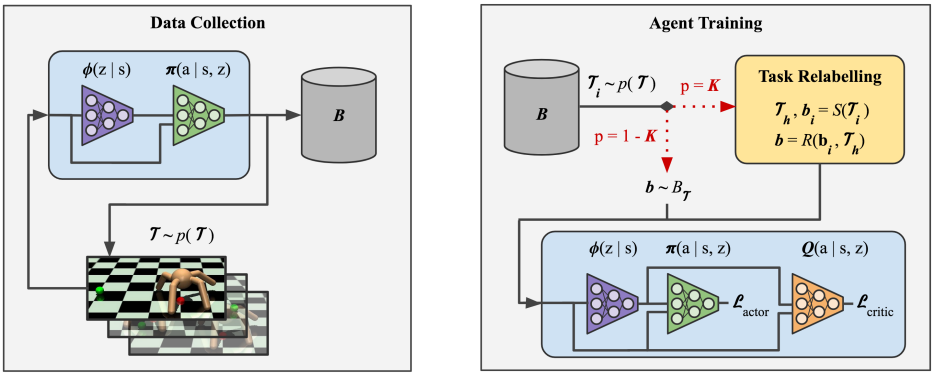


Figure 3: Illustration of Hindsight Task Relabeling (HTR) in a meta-RL training loop. HTR is agnostic to the underlying (off-policy) meta-RL algorithm; the agent architecture and/or training specifics (e.g., the encoder ϕ, actor π and Q-function neural networks shown in blue) can be modified independently of the relabeling scheme. HTR can also be performed in an 'eager' fashion at the data collection stage (as opposed to 'lazy' relabeling in the data sampling stage), see Section 3 for details.

Relabeling transitions under hindsight tasks serves as an additional source of supervision in learning the latent task embedding $\mathbf{z} \Leftarrow f(\mathbf{c})$ by enabling gradient-based optimization of f in the absence of meaningful reward signal (in PEARL, this embedding is parameterized by the context encoder q_ϕ). Similar to how HER generates an implicit curriculum of goals, HTR generates an implicit curriculum of tasks that gradually shifts from easier hindsight tasks towards the true training distribution $\mathcal{T}_{train} \sim p(\mathcal{T})$: initially, the agent is unable to recover reward signal on the true training tasks, but it can bootstrap the learning of an adaptation strategy by learning to adapt on easier hindsight tasks.

We outline our method for meta-training with Hindsight Task Relabeling in Algorithm 1 (task relabeling is only used during meta-training; meta-testing is unchanged). We evaluate our approach using PEARL as the off-policy meta-RL algorithm A, though the approach we describe is general to any off-policy meta-RL algorithm, and could potentially be integrated with on-policy context-based meta-RL algorithms (such as RL^2) via hindsight policy gradients (Rauber et al., 2019).

3.1 Algorithm Design

Two important considerations in adapting hindsight relabeling to meta-RL are (i) how to choose the hindsight task, and (ii) how to sample the transitions to be relabelled. Together these choices form the task relabeling strategy S in Algorithm 1. In HER, transitions are relabelled with goals that are optimal at the trajectory-level (see Figure 1). The key difference between our method and HER is that our meta-RL relabeling scheme needs to construct batches of data for the agent that share the same pseudo-task (or real task), which is fundamentally different from standard HER. HER does not consider the setting where batches of trajectories are collected in different MDPs; in HER, transitions are relabeled independently with locally-optimal pseudo-goals, and batches of training data for the agent contain many different pseudo-goals (and real goals).

In the meta-RL setting, the agent must estimate the task from a recent context of transitions with the same underlying task (unknown to the agent), and therefore relabeling should take into consideration more than a single trajectory; the agent explores an environment which has a fixed task $\mathcal{T} \sim p(\mathcal{T})$, and as it accumulates context it uses the context to identify the nature of the task and update (i.e., adapt) its policy accordingly.

A naive application of hindsight relabeling (e.g., HER) to meta-RL would assign each transition in the batch its own hindsight task (e.g., give each τ in Figure 1b its own $\mathcal{T}'$), however, this is bound to fail if the meta-RL training process is left unchanged, since the context encoder is trained to estimate the task from a collection of transitions generated in that task (not a batch of transitions generated across several distinct tasks). A similarly flawed approach is to generate a single hindsight task from a random transition in the batch, and then to relabel the entire batch of transitions with the same hindsight task (e.g., choose a $\mathcal{T}'$ Figure 1b using a random τ to relabel all τs with). The issue with this approach is that the hindsight task is only guaranteed to be optimal for a single trajectory, and

Algorithm 1: Hindsight Task Relabelling for Off-Policy Meta-Reinforcement Learning

Require: Off-policy meta-RL algorithm A, training tasks $\mathcal{T}_{1:T} \sim p(\mathcal{T})$, context encoder ϕ, actor π, critic Q, relabelling function R, task relabelling strategy S, and task relabelling probability K

Initialize a replay buffer B_i for each training task $\mathcal{T}_i$
while *meta-training* **do**
  for *each* $\mathcal{T}_i$ **do**
    Collect data on $\mathcal{T}_i$ according to A (e.g., by rolling out π conditioned on $z \sim \phi$)
    Add data to task replay buffer B_i
  for *each* $\mathcal{T}_i$ **do**
    $p_h \sim \mathcal{U}(0, 1)$
    if $p_h \geq K$ **then**
      Select a minibatch of transitions b_i and hindsight task $\mathcal{T}_h$ using strategy S
      Relabel transitions in b_i, i.e., $R(b_i, \mathcal{T}_h)$
    else
      Sample minibatch of transitions $b_i \sim B_i$
    Compute gradients and update ϕ, π, and Q using b_i according to A

may in fact be highly sub-optimal for other trajectories in the batch. In practice, we observed that using this method only resulted in a small fraction of transitions in each batch receiving reward under the new task, and despite the increase in reward signal, this method was unable to train the context encoder (likely due to sub-optimal relabeling). We present two relabeling strategies, *Single Episode Relabeling* and *Episode Clustering*, that are inspired by the relabeling strategy used in Andrychowicz et al. (2017) but are specifically designed to work in the meta-RL setting.

Single Episode Relabeling (SER) strategy. In SER, a single episode (i.e., trajectory) is randomly selected to generate the hindsight task and sample transitions from (i.e., sample N transitions from 1 trajectory, each relabelled under 1 new task). Despite the fact that this reduces the pool of context from the entire task buffer to a single episode, this setting closely resembles meta-testing (where context is drawn purely online), and we found this resampling strategy to work well in practice since it results in dense reward signal. HTR using SER is outlined in Algorithm 1.

Episode Clustering (EC) strategy. An alternative strategy to single episode resampling is to cluster trajectories that satisfied similar hindsight tasks into the same task buffers B_h, essentially creating additional task buffers for hindsight tasks that can be used for training in-place of the real task buffers B_i (which are also sampled with some probability K). For example, in goal-reaching tasks, trajectories can easily be mapped to hindsight buffers with trajectories from separate trials by discretizing the state space and creating a buffer for each partition. In more complex tasks, alternative clustering approaches (e.g., learning-based) can be used to group together trajectories that can be relabeled with high reward conditioned on the same tasks. Episode clustering creates similarly dense reward to the resampling strategy, but with less duplicate transitions per-batch. In practice, we found the SER strategy to be far less complex and similarly effective to the EC approach. See Section 4 for an empirical comparison between SER and EC, and the supplement for a more detailed comparison.

Comparison to HER. Both SER and EC are similar to the relabeling strategy used in HER in some respects, but differ in others: HER samples N transitions from the full replay buffer, and relabels each transition independently (N different goals). HTR with SER samples 1 trajectory, and samples N transitions from that trajectory all relabelled with 1 new task. HTR with EC samples N transitions from 1 hindsight replay buffer, where each transition in that buffer has been relabeled with 1 new task. In HER, the hindsight goal for a given transition is chosen by selecting a random transition that occurs after the sampled transition in the same trajectory. In HTR, hindsight tasks are assigned in the same way as HER, but a single hindsight task is applied to an entire batch of transitions.

Limitations. The key limitation of our method is that it assumes trajectories can be relabeled under new task (i.e., a mapping $\mathbf{s} \rightarrow \mathcal{T}$ or $\tau \rightarrow \mathcal{T}$ exists), which is a reasonable assumption for goal-reaching environments (as studied in Rakelly et al. (2019); Gupta et al. (2018b); Andrychowicz et al. (2017)), but may be significantly more challenging in other environments. Imagine a complex robotic manipulation environment (e.g., retrieving an object from a drawer) where reward is sparse and only received upon successful completion of the entire task; in this scenario, relabeling the zero-reward

transitions generated by an initial random policy into useful signal for training an optimal policy is less straightforward than the goal-reaching setting, and may require explicit sub-task specification using domain expertise.

Similar to how HER is not necessarily only for goal-reaching *goal-conditioned* RL, HTR is not necessarily only for goal-reaching *meta*-RL, and if a good relabeling function exists then either approach can be successfully applied in their respective setting (HER for goal-conditioned RL, HTR for meta-RL). However, as in the original HER work, we focus our experiments on goal-reaching tasks, where the reward function is a simple function of the state and can be easily repurposed for relabeling. It may be possible to relax this assumption and extend HTR to more families of tasks by employing more complex task relabeling schemes, e.g., by extending work on inverse RL for hindsight relabeling (Li et al., 2020; Eysenbach et al., 2020) to the meta-RL setting, or by carefully engineering reward functions specific to each environment, however we leave this to future work.

Additionally, our method adds a new hyperparameter K (to the underlying off-policy meta-RL algorithm) which may need to be tuned for optimal results. However we note that the specification of a relabeling probability K is also required in standard HER (see Andrychowicz et al. (2017)).

4 Experiments

In evaluating our proposed method, we aim to answer the following questions: (i) Can Hindsight Task Relabeling enable (meta-)learning of adaptation strategies in challenging sparse reward environments, where existing meta-RL methods fail without shaped reward? (ii) How do the adaptation strategies learned using Hindsight Task Relabeling compare to those learned using shaped reward functions? (iii) How do key implementation choices (such as relabeling probability K) affect its performance?

4.1 Environments

We evaluate our method on a suite of sparse reward environments based those proposed by Gupta et al. (2018b) and Rakelly et al. (2019) (see Figure 2). In prior work, each environment exposes two reward functions: a dense reward function used during meta-training, and a sparse reward function used during meta-testing. The key difference in our experimental setup is that we consider the setting where sparse reward is used both during meta-testing *and* meta-training. In each environment, a set of 100 tasks is sampled for meta-training, and a set of 100 tasks is sampled from the same task distribution for meta-testing. The environments were each modified to increase their difficulty, such that random exploration is far less likely to encounter sparse reward. Refer to the supplement for full hyperparameter, environment, and experimental details.

2D Navigation. A point robot must navigate a 2D plane to different goals (i.e., tasks) located along the perimeter of a half-semicircle with radius 2. The state includes the robot's coordinates but does not include the goal, therefore the agent must explore the environment to discover the goal. In the dense reward variant of the environment, the reward is the negative $L2$ distance to the goal. In the sparse reward variant, reward is given only when the agent is within a short distance to the goal. An example optimal exploration strategy in the sparse reward variant is to efficiently traverse the perimeter of the semicircle until the goal is found.

Wheeled Locomotion. To test if our approach generalizes to more complex state and action spaces, we use several continuous control locomotion environments. In the wheeled locomotion environment, the agent must navigate to the goal distribution by controlling two independent wheels to turn and accelerate. Similar to the point robot, the dense reward function is the negative distance to the goal, while the sparse reward function only provides reward signal near the goal.

Ant (Quadruped) Locomotion. This environment requires controlling a quadruped ant with a high-dimensional state and action space. The task distribution and reward functions are the same as the point robot and wheeled locomotion environment, however, exploration is significantly more challenging since it requires coordinating movement with four legs to navigate to specific locations.

4.2 Results

Meta-training using only sparse reward. To answer question (i), we compare our hindsight task relabeling approach (using PEARL as the base meta-RL algorithm) to standard PEARL, as well

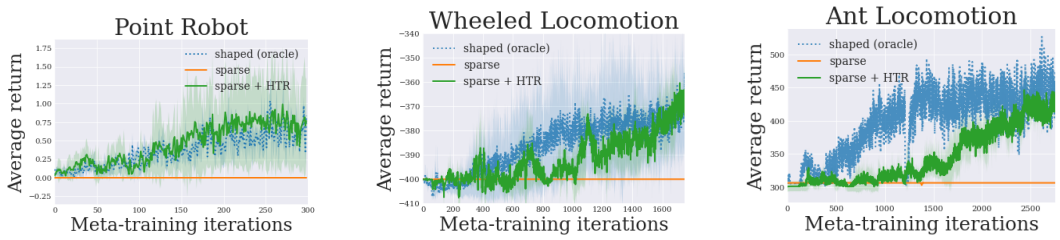


Figure 4: Evaluating adaptation to train tasks progressively *during* meta-training. Y-axis measures average sparse return during adaptation throughout meta-training (shaded std dev), though the oracle is still trained using dense reward. Conventional meta-RL methods are unable to learn using sparse reward. Hindsight Task Relabeling (HTR) is comparable to dense reward meta-training performance.

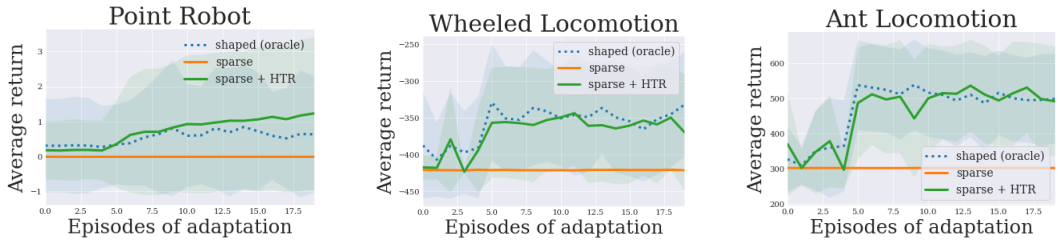


Figure 5: Evaluating adaptation to test tasks *after* meta-training. Y-axis measures average (sparse) return during adaptation using context collected online, using sparse reward only. Adaptation strategies learned with Hindsight Task Relabeling (HTR) generalize to held-out tasks as well as or better than the oracle which is learned using shaped reward functions. Without HTR or access to a shaped reward during meta-training, the agent is unable to learn an optimal adaptation strategy.

as an oracle PEARL which uses the dense reward function during training. Our oracle baseline is equivalent to the standard setup in the 'sparse' reward experiments in Gupta et al. (2018b) and Rakelly et al. (2019). Note that the term 'oracle' is a misnomer, since training on a separate dense reward function designed to aid the learning of an RL agent (to be evaluated on the sparse reward function) is inherently optimizing a different objective. Optimizing an RL agent for an auxiliary objective has been shown to lead to inadvertently bad performance on the true objective (Clark & Amodei, 2016), and therefore it is often better to directly optimize for the true objective (e.g., the original sparse reward function) when possible.

As discussed in prior work, meta-learning on sparse reward environments proves extremely challenging: existing meta-RL methods are unable to make progress within a reasonable timespan just using sparse rewards (see Figures 4 and 6). Although PEARL is unable to learn using sparse reward in the environments considered, it can learn an adaptation strategy using dense reward functions that meaningfully transfers to the sparse reward settings. However, as mentioned in Section 1, in many settings designing a separate dense reward function to aid training in infeasible or impractical, and therefore it is of significant interest to devise meta-RL algorithms that can learn to learn with sparse reward. Hindsight Task Relabeling is extremely effective at providing the reward signal needed to meta-train with sparse reward, as seen in Figures 4 and 6. In all environments, HTR is not only comparable to using shaped reward during meta-training, but it also can learn adaptation strategies that perform as well as the oracle during meta-testing (as seen in Figure 5).

Exploration during meta-testing. In Figure 5, we see that the adaptation strategies learned with Hindsight Task Relabeling are similarly effective while only using the original sparse reward function. Because HTR does not train on dense reward, there is no opportunity for a mismatch between reward functions at meta-training and meta-testing: HTR optimizes for the true reward function on a superset of tasks $\mathcal{T}_{hindsight} \cup \mathcal{T}_{train} \sim p(\mathcal{T})$, whereas using a shaped reward optimizes for a proxy reward function on the true set of training tasks $\mathcal{T}_{train} \sim p(\mathcal{T})$.

Varying relabeling probability K. We found a relatively low relabeling probability (e.g., $K = 0.1$ and 0.3) was often most effective for HTR (see Figure 7), e.g., $K = 0.1$ means that relabelled data from a hindsight task is (approximately) as likely to be used for training as data from any of the 100

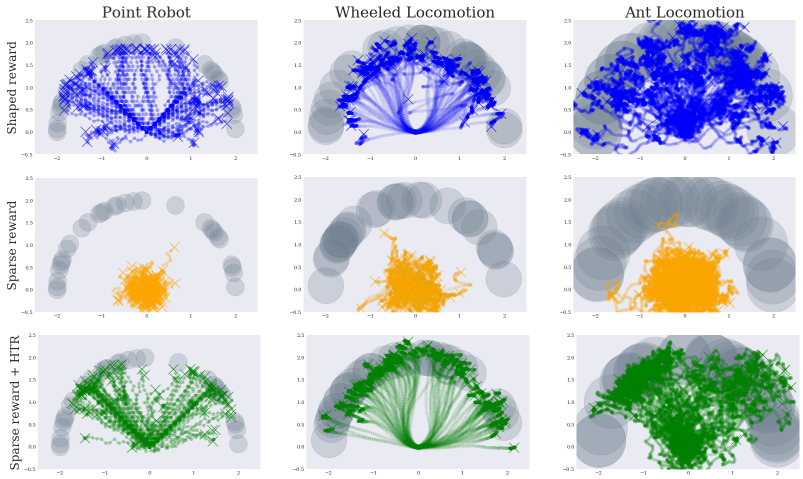


Figure 6: Visualizing exploration behavior learned during meta-training using 100 pre-adaptation trajectories (i.e., sampled from the latent task prior). In the sparse reward setting, without HTR (middle row) the agent is unable to learn a meaningful exploration strategy and appears to explore randomly near the origin. With HTR (bottom row), the agent learns to explore near the true task distribution (grey circles), similar to an agent trained with a shaped dense reward function (top row).

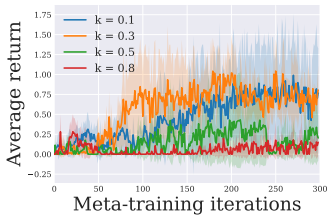


Figure 7: Average return when varying K on Point Robot.

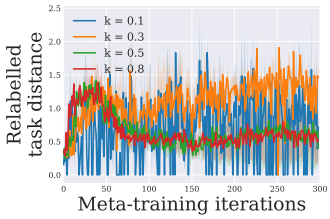


Figure 8: Average task distance when varying K on Point Robot.

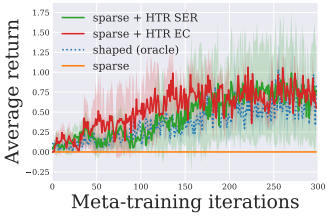


Figure 9: Comparing HTR with SER vs EC on Point Robot.

tasks in $\mathcal{T}_{train}$. $K = 0.1$ corresponds to using far less relabelled data for training than in conventional HER: in contrast, Andrychowicz et al. (2017) relabel 80% of sampled transitions. We hypothesize that a low K works best for HTR since it provides enough reward signal to bootstrap learning, without biasing the task distribution too far from the ground truth distribution; in Fig. 8, we can see that both $K = 0.5$ and 0.8 converge on a mean relabelled task distance of 0.5, whereas $K = 0.1$ and 0.3 are closer to the ground truth task distance of 1.0.

Empirical comparison of Single Episode Relabeling (SER) and Episode Clustering (EC). As described in Section 3.1, we found that EC generally performed similarly to SER, despite being significantly more complex to implement, and requiring additional tuned hyperparameters (see Figure 9 for SER vs EC on Point Robot). Both SER and EC are implemented in the code in the supplement.

5 Conclusion

In this paper, we propose a novel approach for meta-reinforcement learning in sparse reward environments that can be incorporated into any off-policy meta-RL algorithm. The sparse reward meta-RL setting is extremely challenging, and existing meta-RL algorithms that learn adaptation strategies for sparse reward environments require dense reward functions during meta-training. Our approach, Hindsight Task Relabeling (HTR), enables learning adaptation strategies in challenging sparse reward environments without engineering additional reward functions. HTR relabels data collected on ground truth meta-training tasks as data for achievable hindsight tasks, which allows for meta-training on the original sparse reward objective. Not only does our approach enable learning adaptation strategies in challenging sparse reward environments without reward engineering, but it also generates adaptation strategies comparable to prior approaches that use shaped reward functions.

References

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. In *Neural Information Processing Systems (NeurIPS)*, pp. 5048–5058, 2017.

Bengio, Y., Bengio, S., and Cloutier, J. *Learning a synaptic learning rule*. Citeseer, 1990.

Clark, J. and Amodei, D. Faulty reward functions in the wild. *https://blog.openai.com/faulty-reward-functions*, 2016.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2019.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv:1611.02779*, 2016.

Eysenbach, B., Geng, X., Levine, S., and Salakhutdinov, R. Rewriting history with inverse rl: Hindsight inference for policy improvement. *arXiv preprint arXiv:2002.11089*, 2020.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Florensa, C., Held, D., Geng, X., and Abbeel, P. Automatic goal generation for reinforcement learning agents. In *International Conference on Machine Learning (ICML)*, 2018.

Gupta, A., Eysenbach, B., Finn, C., and Levine, S. Unsupervised meta-learning for reinforcement learning. *arXiv preprint arXiv:1806.04640*, 2018a.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. In *Neural Information Processing Systems (NeurIPS)*, pp. 5302–5311, 2018b.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Houthooft, R., Chen, Y., Isola, P., Stadie, B., Wolski, F., Ho, O. J., and Abbeel, P. Evolved policy gradients. In *Neural Information Processing Systems (NeurIPS)*, pp. 5400–5409, 2018.

Jabri, A., Hsu, K., Gupta, A., Eysenbach, B., Levine, S., and Finn, C. Unsupervised curricula for visual meta-reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, pp. 10519–10530, 2019.

Kaelbling, L. P. Learning to achieve goals. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pp. 1094–1099. Citeseer, 1993.

Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research (JMLR)*, 17(1):1334–1373, 2016.

Levy, A., Platt, R., and Saenko, K. Hierarchical actor-critic. *arXiv preprint arXiv:1712.00948*, 2017.

Li, A. C., Pinto, L., and Abbeel, P. Generalized hindsight for reinforcement learning. *arXiv preprint arXiv:2002.11708*, 2020.

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

Mishra, N., Rohaninejad, M., Chen, X. P., and Abbeel, P. A simple neural attentive meta-learner. In *International Conference on Learning Representations (ICLR)*, 2018.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540), 2015.

Nair, A. V., Pong, V., Dalal, M., Bahl, S., Lin, S., and Levine, S. Visual reinforcement learning with imagined goals. In *Neural Information Processing Systems (NeurIPS)*, pp. 9191–9200, 2018.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *International Conference on Machine Learning (ICML)*, volume 99, pp. 278–287, 1999.

Packer, C., Gao, K., Kos, J., Krähenbühl, P., Koltun, V., and Song, D. Assessing generalization in deep reinforcement learning. *arXiv:1810.12282*, 2018.

Rakelly, K., Zhou, A., Quillen, D., Finn, C., and Levine, S. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning (ICML)*, 2019.

Rauber, P., Ummadisingu, A., Mutz, F., and Schmidhuber, J. Hindsight policy gradients. In *International Conference on Learning Representations (ICLR)*, 2019.

Rothfuss, J., Lee, D., Clavera, I., Asfour, T., and Abbeel, P. Promp: Proximal meta-policy search. In *International Conference on Learning Representations (ICLR)*, 2019.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

Stadie, B. C., Yang, G., Houthooft, R., Chen, X., Duan, Y., Wu, Y., Abbeel, P., and Sutskever, I. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

Sung, F., Zhang, L., Xiang, T., Hospedales, T., and Yang, Y. Learning to learn: Meta-critic networks for sample efficient learning. *arXiv:1706.09529*, 2017.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2017.

Thrun, S. and Pratt, L. Learning to learn: Introduction and overview. In *Learning to learn*, pp. 3–17. Springer, 1998.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *arXiv:1611.05763*, 2016.

Xu, T., Liu, Q., Zhao, L., and Peng, J. Learning to explore via meta-policy gradient. In *International Conference on Machine Learning (ICML)*, pp. 5463–5472, 2018a.

Xu, Z., van Hasselt, H. P., and Silver, D. Meta-gradient reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, pp. 2396–2407, 2018b.

Zhang, A., Ballas, N., and Pineau, J. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv:1806.07937*, 2018.

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. Fast context adaptation via meta-learning. *arXiv preprint arXiv:1810.03642*, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] **See Section 3.1.**
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] **See Appendix section *Broader Impacts* for discussion of potential societal impacts.**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] **See the included ZIP file with full code and README instructions.**
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] **Full training details are included in the Appendix.**
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] **Figures plotting averages include mean and standard deviation (shaded areas).**
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] **Details on compute resources (type of compute, compute provider, and time required to run experiments) included in the Appendix.**
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] **See README in code repository.**
   (b) Did you mention the license of the assets? [Yes] **See README in code repository (all licensed under the MIT License).**
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] **See included ZIP file for (new) code that accompanies the paper. Code will be made public to accompany the paper upon publication.**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]