A Proofs

A.1 Bayes Optimality of Q-value Estimates in Bernoulli Multi-armed Bandits

Given an instance of a Bernoulli multi-armed bandit MDP, $M_i \sim \mathcal{M}$, and trajectory data $\Upsilon_{1:T}$ up to time T, we would like to show that the probability $P(i|\Upsilon_{1:T})$ can be determined entirely from Q-estimates Q_i^T and action-counts N_i^T, as long as the initial belief is uniform or known.

In the following proof, we represent an instance i of K-armed Bandits as a K-dimensional vector of success probabilities $[p_{i1}, ..., p_{iK}]$, such that pulling arm k is associated with reward distribution $P(r=1|i,k) = p_{ik}$ and $P(r=0|i,k) = (1-p_{ik})$.

Let the number of times arm k is pulled up to time T be N_{ik}^T, and the number of successes associated with pulling arm k up to time T be q_{ik}^T. Given that this is an MDP with just a single state and task horizon of 1, the Q-estimate associated with arm k is just the average reward for that action, which is the ratio of successes to counts associated with that action i.e., $Q_{ik}^T = \frac{q_{ik}^T}{N_{ik}^T}$. To reduce the clutter in the notation, we will drop the superscript T for the rest of the subsection.

Now,

$$P(i|\Upsilon_{1:T}) = \alpha P(i) \cdot P(\Upsilon_{1:T}|i) \tag{7}$$

where α is the normalization constant, $P(i)$ is the prior probability of task i (which is assumed to be known beforehand), and $\Upsilon_{1:T}$ is the sequence of actions and the corresponding rewards up to time T. Assuming, without loss of generality, that the sequence of actions used to disambiguate tasks is a given, $P(\Upsilon_{1:T}|i)$ becomes simply the product of probabilities of reward outcomes up to time T, noting that the events are independent. Therefore,

$$\begin{aligned}
P(\Upsilon_{1:T}|i) &= \prod_{k=1:K} \prod_{t=1:T} ([r_{tk}=1]p_{ik} + [r_{tk}=0](1-p_{ik})) && (8)\\
&= \prod_{k=1:K} p_{ik}^{q_{ik}} \cdot (1-p_{ik})^{N_{ik}-q_{ik}} && (9)\\
&= \prod_{k=1:K} p_{ik}^{Q_{ik}N_{ik}} \cdot (1-p_{ik})^{N_{ik}-Q_{ik}N_{ik}} && (10)
\end{aligned}$$

Putting everything together,

$$P(i|\Upsilon_{1:T}) = \alpha P(i) \cdot \prod_{k=1:K} p_{ik}^{Q_{ik}N_{ik}} \cdot (1-p_{ik})^{N_{ik}-Q_{ik}N_{ik}} \tag{11}$$

This equation proves that N_i^T and Q_i^T are sufficient statistics to determine $P(i|\Upsilon_{1:T})$ in this domain, assuming that the prior over task distribution is known. $\square$

A.2 Non-Bayes Optimality of Q-value Estimates in Gaussian Multi-armed Bandits

Given an instance of a Gaussian multi-armed bandit MDP, $M_i \sim \mathcal{M}$, and trajectory data $\Upsilon_{1:T}$ up to time t, here we derive the closed-form expression of the probability $P(i|\Upsilon_{1:T})$ and show that it contains terms other than Q-estimates Q_i^t and action-counts N_i^t.

In the following proof, we represent an instance i of K-armed Bandits as a $2K$-dimensional vector of means and standard deviations $[\mu_{i1}, ..., \mu_{iK}, \sigma_{i1}, ..., \sigma_{iK}]$, such that pulling arm k is associated with reward distribution $P(r|i,k) = \frac{1}{\sqrt{2\pi\sigma_{ik}}} \exp(\frac{r-\mu_{ik}}{\sigma_{ik}})^2$.

Let the number of times arm k is pulled up to time T be N_{ik}^T. Given that this is an MDP with just a single state and the task horizon is 1, the Q-estimate associated with arm k is just the average reward for that action $\text{Avg}[r_k]$ up to time T. To reduce the clutter in the notation, we will drop the superscript T for the rest of the subsection.

As in the previous subsection, we now compute the likelihood $P(\Upsilon_{1:T}|i)$.

$$P(\Upsilon_{1:T}|i) = \prod_{k=1:K}\prod_{t=1:T}\frac{1}{\sqrt{2\pi\sigma_{ik}}}\exp(\frac{r_{tk}-\mu_{ik}}{\sigma_{ik}})^2 \tag{12}$$

Therefore, the log likelihood is

$$\log P(\Upsilon_{1:T}|i) = \sum_{k=1:K}\sum_{t=1:T}\frac{(r_{tk}-\mu_{ik})^2}{\sigma_{ik}^2} - \log(2\pi\sigma_{ik})/2 \tag{13}$$

$$= \sum_{k=1:K} N_{ik}\frac{\text{Avg}[(r_{tk}-\mu_{ik})^2]}{\sigma_{ik}^2} - N_{ik}\log(2\pi\sigma_{ik})/2 \tag{14}$$

$$= \sum_{k=1:K} N_{ik}\frac{\text{Avg}[r_k^2] - 2\mu_{ik}\text{Avg}[r_k] + \mu_{ik}^2}{\sigma_{ik}^2} - N_{ik}\log(2\pi\sigma_{ik})/2 \tag{15}$$

$$= \sum_{k=1:K} N_{ik}\frac{(\text{Var}[r_k] + \text{Avg}[r_k]^2) - 2\mu_{ik}\text{Avg}[r_k] + \mu_{ik}^2}{\sigma_{ik}^2} - N_{ik}\log(2\pi\sigma_{ik})/2 \tag{16}$$

$$= \sum_{k=1:K} N_{ik}\frac{\text{Var}[r_k] + (Q_{ik})^2 - 2\mu_{ik}Q_{ik} + \mu_{ik}^2}{\sigma_{ik}^2} - N_{ik}\log(2\pi\sigma_{ik})/2 \tag{17}$$

Therefore, computing this expression requires computing the variance in rewards, $\text{Var}[r_k]$, associated with each arm up to time T, apart from the Q-estimates and action-counts. This proves that Q-estimates and action-counts alone are insufficient to completely determine $P(i|\Upsilon_{1:T})$ in Gaussian multi-armed bandits domain. ☐

A.3 Object-level Q-estimates and Meta-level Values

Proof of Equation 4: In standard meta-RL, the only observed variable in the POMDP state $\bar{s}_t = [s_t, i]$ at time t is the state s_t of the current MDP i.e., $\bar{\omega}_t = s_t$, while the task identity i is hidden. However, in RL^3, $\bar{\omega}_t$ includes the vector of Q-estimates $Q_i^t(s_t)$ for the hidden task, which means that the meta-level observation function $\bar{O}(\bar{\omega}|\bar{b}, a)$ factors in the probability that a particular Q-esimate will be observed following an action a given an initial belief $\bar{b}$ state. (Note that we will use $\bar{b}(\bar{s})$ and $\bar{b}(i)$ interchangeably since i is the only hidden variable in $\bar{s}$). In practice, such Q-value estimates provide excellent evidence (see Appendix D) for task identification. This allows for robust belief recovery even if the initial belief is not Bayes-optimal (or altogether not maintained), especially as the Q-estimates converge and stabilize in the limit, leading to two cases:

Case 1: The observed Q-values are unique to MDP M_i. In this case, the belief distribution will collapse rapidly to zero for tasks $j \neq i$, and thus $\max_{a\in A} Q_i(s,a) = \bar{V}^*(\bar{b})$.

Case 2: The observed Q-values are not unique. In this case, belief will not collapse to a single MDP. However, belief will still reduce to zero for tasks not compatible with the observed Q-values. The meta-level value function $\bar{V}^*(\bar{b})$, which will be an expectation over object-level values, will simplify to $\max_{a\in A} Q_i(s,a)$ since Q-values for all remaining tasks are identical, where i may represent any of the (identical Q-valued) tasks with non-zero belief.

This proves equation 4. Note that in the limit, the task can be identified perfectly from the stream of experiences as all state-action pairs are explored, and the meta-level value function becomes equivalent to the optimal object-level value function of the identified (or current) task. However, the above proof demonstrates that RL^3 can infer this equivalency implicitly in the limit without relying on the stream of experiences or identifying the task fully, and furthermore, directly model the meta-value function in terms of the supplied object-level value function. ☐

Proof of Equation 6: We first write the Bellman equation for the optimal meta-level POMDP value function in its belief-MDP representation:

$$\bar{V}^*(\bar{b}) = \max_{a \in A} \Big[\sum_{\bar{s} \in \bar{S}} \bar{b}(\bar{s}) \bar{R}(\bar{s}, a) + \gamma \sum_{\bar{\omega} \in \bar{\Omega}} \bar{O}(\bar{\omega} | \bar{b}, a) \bar{V}^*(\bar{b}') \Big]. \tag{18}$$

However, given that in the POMDP state $\bar{s} = [s, i]$, the only hidden variable is the task i, we can re-write this as

$$\bar{V}^*(\bar{b}) = \max_{a \in A} \Big[\sum_{M_i \in \mathcal{M}} \bar{b}(i) R_i(s, a) + \gamma \sum_{\bar{\omega} \in \bar{\Omega}} \bar{O}(\bar{\omega} | \bar{b}, a) \bar{V}^*(\bar{b}') \Big], \tag{19}$$

where $\bar{b}(i)$ denotes the meta-level belief that the agent is operating in MDP M_i, and $R_i(s, a)$ is the reward experienced by the agent if it executes action a in state s in MDP M_i. Here, $\bar{b}'$ may be calculated via the belief update as in §3.1. □

B Architecture

B.1 RL^2

Our modified implementation of RL^2 uses transformer decoders (Vaswani et al., 2017) instead of RNNs to map trajectories to action probabilities and meta-values, in the actor and the critic, respectively, and uses PPO instead of TRPO for outer RL. The decoder architecture is similar to (Vaswani et al., 2017), with 2 layers of masked multi-headed attention. However, we use learned position embeddings instead of sinusoidal, followed by layer normalization. Our overall setup is similar to (Esslinger et al., 2022).

For each meta-episode of interactions with an MDP M_i, the actor and the critic transformers look at the entire history of experiences up to time t and output the corresponding action probabilities $\pi_1 ... \pi_t$ and meta-values $\bar{V}_1 ... \bar{V}_t$, respectively. An experience input to the transformer at time t consists of the previous action a_{t-1}, the latest reward r_{t-1}, the current state s_t, episode time step t_τ, and the meta-episode time step t, all of which are normalized to be in the range $[0, 1]$. In order to reduce inference complexity, say at time step t, we append t new attention scores (corresponding to experience input t w.r.t. the previous $t-1$ experience inputs) to a previously cached $(t-1) \times (t-1)$ attention matrix, instead of recomputing the entire $t \times t$ attention matrix. This caching mechanism is implemented for each attention head and reduces the inference complexity at time t from $\mathcal{O}(t^2)$ to $\mathcal{O}(t)$.

B.2 RL^3

The input of the transformer in RL^3 includes a vector of Q estimates (in practice, they are supplied as the vector of advantage estimates ($Q - \max_a Q$) along with the value function ($\max_a Q$) separately) and a vector of action counts at each step t for the corresponding state. As mentioned in Section 4.2, this is implemented in our code simply by converting MDPs in the problem set to VAMDPs using a wrapper and running our implementation of RL^2 thereafter. The pseudocode is shown in the algorithm 1. The Markov version of RL^3 uses a dense neural network, with two hidden layers of 64 nodes each, with the ReLU activation function.

For object-level RL, we use model estimation followed by value iteration (with discount factor $\gamma = 1$) to obtain Q-estimates. The transition probabilities and the mean rewards are estimated using maximum likelihood estimation (MLE), with Laplace smoothing (coefficient = 0.1) for transition probabilities estimation. For unseen actions, rewards are assumed to be zero, and transitions equally likely to other states. States are added to the model incrementally when they are visited, so that value iteration does not compute values for unvisited states. Moreover, value iteration is carried out only for iterations equal to the task horizon (which is 1, 10, 250, 350 for Bandits, MDPs, 11x11 GridWorld, 13x13 GridWorld domains, respectively), unless the maximum Bellman error drops below 0.01.

Algorithm 1 Value-Augmenting Wrapper for Discrete MDPs

```
procedure ResetMDP(vamdp)
    vamdp.$t \leftarrow 0$; vamdp.$t_\tau \leftarrow 0$
    vamdp.$N[s,a] \leftarrow 0$; vamdp.$Q[s,a] \leftarrow 0 \quad \forall s \in S, a \in A$
    vamdp.rl $\leftarrow$ InitRL()
    $s =$ ResetMDP(vamdp.mdp)
    return OneHot$(s) \cdot Q[s] \cdot N[s]$
procedure StepMDP(vamdp, $a$)
    $s \leftarrow$ mdp.$s$
    $r, s' \leftarrow$ StepMDP(vamdp.mdp, $a$)
    $d \leftarrow$ Terminated(vamdp.mdp)
    vamdp.$t$, vamdp.$N[s,a]$, vamdp.$t_\tau \leftarrow += 1$
    vamdp.$Q \leftarrow$ UpdateRL(vamdp.rl, $s, a, r, s', d$)
    if $d$ or vamdp.$t_\tau \geq$ task_horizon then
        vamdp.$t_\tau \leftarrow 0$
        $s' \leftarrow$ ResetMDP(vamdp.mdp)
    return $r$, OneHot$(s') \cdot Q[s'] \cdot N[s']$        ▷ Concatenate state, $Q$-estimates and action counts
procedure Terminated(vamdp)
    return vamdp.$t \geq H$
```

B.3 RL^3-COARSE

During model estimation in RL^3-coarse, concrete states in the underlying MDP are incrementally clustered into abstract states as they are visited. When a new concrete state is encountered, its abstract state ID is set to that of a previously visited state within a ‘clustering radius’, unless that previous state is already part of a full cluster (determined by a maximum ‘cluster size’ parameter). If multiple visited states satisfy the criteria, the ID of the closet one is chosen. If none of the visited states that satisfy the criteria, then the new state is assigned a new abstract state ID, increasing the number of abstract states in the model. It is worth noting that this method of deriving abstractions does not take advantage of any structure in the underlying domain. However, this simplicity makes it general purpose, efficient, and impartial, while still leading to excellent performance. For our GridWorld domain, we chose a cluster size of 2 and a clustering radius such that only non-diagonal adjacent states are clustered (Manhattan radius of 1).

The mechanism for learning the transition function and the reward function in the abstract MDP is the same as before. For estimating Q-values for a given concrete state, value iteration is carried out on the abstract MDP and the Q-estimates of the corresponding abstract state are returned.

C TRAINING

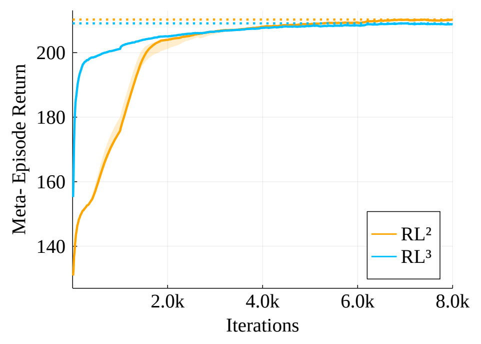
Figs. 5, and 6 show the training curves for MDPs, and GridWorld environments, respectively, across 3 random seeds. The results in the main text correspond to the median model. We ran the experiments on Nvidia GeForce RTX 2080 Ti GPUs for context length ≤ 256 which took approximately 12-24 hours, and on Nvidia A100 GPUs for higher context lengths, which took 1-2 days.

D ADDITIONAL ANALYSIS

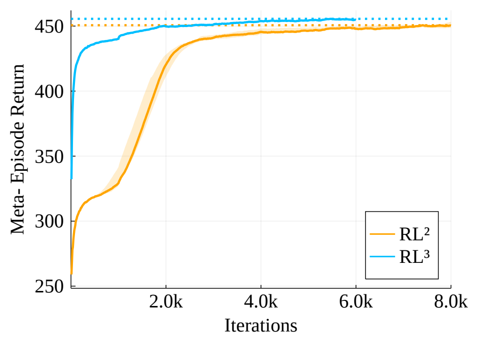
In this section, we show that Q-estimates, though imperfect, produce reasonable signals for task identification. Here, we test this claim thoroughly with 3 analyses.

D.1 REQUIREMENTS FOR A UNIQUE Q^*-FUNCTION

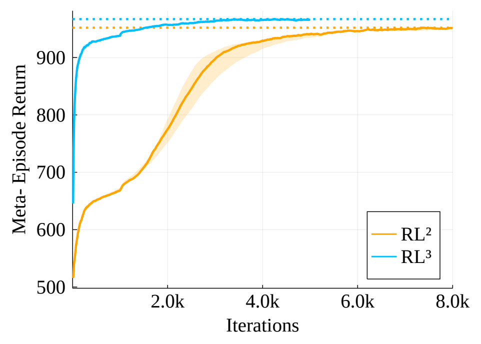
Throughout, we assume fixed state space and action space. Below, we show that if the transition function is fixed, then two Q^*-tables will be identical if and only if both reward functions are also equal. First, we show that identical Q^* functions imply identical reward functions. Given the



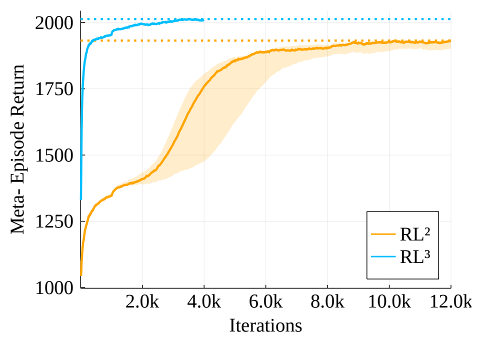
(a) $H = 128$



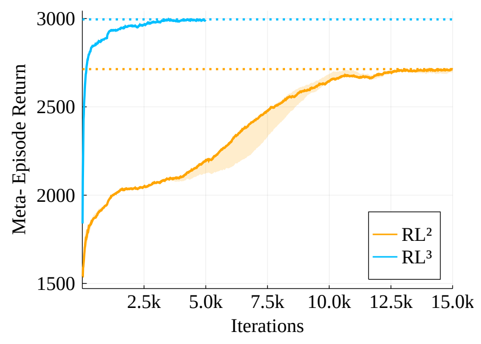
(b) $H = 256$



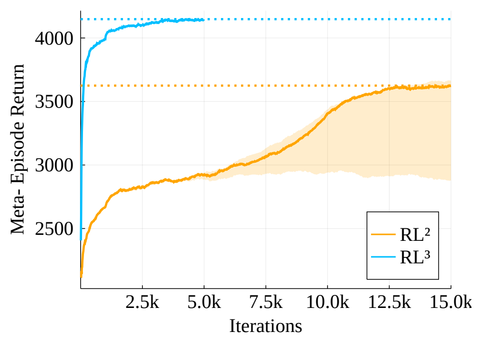
(c) $H = 512$



(d) $H = 1024$

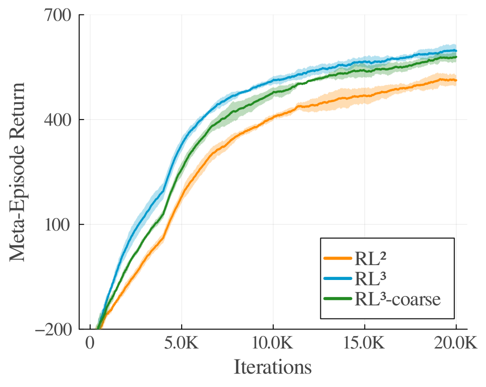


(e) $H = 1500$



(f) $H = 2048$

Figure 5: Average meta-episode return vs PPO iterations for MDPs domain for different interaction budgets.



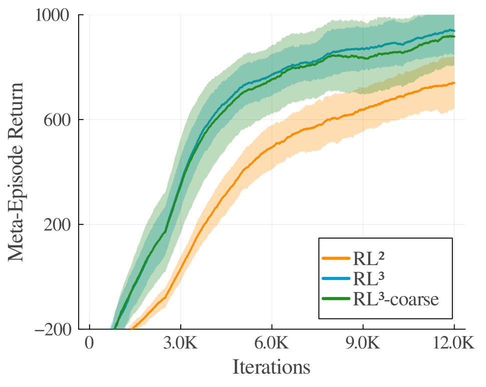


Figure 6: Average meta-episode return vs PPO iterations for GridWorld 11x11 (*left*) and 13x13 (*right*).

Bellman equations,

$$Q_1^*(s,a) = R_1(s,a) + \gamma \sum_{s'} T(s,a,s') \text{max}_{a'} Q_1^*(s',a') \tag{20}$$

$$Q_2^*(s,a) = R_2(s,a) + \gamma \sum_{s'} T(s,a,s') \text{max}_{a'} Q_2^*(s',a') \tag{21}$$

Substituting $Q_2^* = Q_1^*$ in Equation equation 21, we get

$$Q_1^*(s,a) = R_2(s,a) + \gamma \sum_{s'} T(s,a,s') \text{max}_{a'} Q_1^*(s',a') \tag{22}$$

Subtracting Equation equation 20 from Equation equation 22, we get $R_1(s,a) = R_2(s,a)$. Thus, $(Q_1^* = Q_2^*) \wedge (T_1 = T_2) \implies (R_1 = R_2)$.

Now, if two MDPs have the same reward and transition function, they are the same MDP and will have the same optimal value function. So, $(R_1 = R_2) \wedge (T_1 = T_2) \implies (Q_1^* = Q_2^*)$.

Since encountering similar Q^*-tables is thus dependent on both transitions and rewards 'balancing' each other, the question is then for practitioners: How likely are we to get many MDPs that all appear to have very similar Q^*-tables?

D.2 Empirical Test using Max Norm

Given an MDP with 3 states and 2 actions, we want to find the probability that $||Q_1^* - Q_2^*||_\infty < \delta$, where Q_1^* and Q_2^* are 6-entry (3 states × 2 actions) Q^*-tables. The transition and reward functions are drawn from distributions parameterized by α and β, respectively. Transition probabilities are drawn from a Dirichlet distribution, Dir(α), and rewards are sampled from a normal distribution, $\mathcal{N}(1, \beta)$. In total, we ran 3 combinations of α and β, each with 50,000 MDPs, a task horizon of 10, and $\delta = 0.1$. To get the final probability, we test all $((50,000 - 1)^2)/2$ non-duplicate pairs and count the number of max norms less than δ.

Results: For $\alpha = 1.0$, $\beta = 1.0$, we found the probability of a given pair of MDPs having duplicate Q^*-table to be $\epsilon = 2.6 \times 10^{-9}$. For $\alpha = 0.1$, $\beta = 1.0$, which is a more deterministic setting, we found $\epsilon = 4.6 \times 10^{-9}$. Further, with $\alpha = 0.1$, $\beta = 0.5$, where rewards are more closely distributed, we found $\epsilon = 1.1 \times 10^{-7}$. Overall, we can see that even for a set of very small MDPs, the probability of numerically mistaking one Q^*-table for another is vanishingly small.

D.3 Predicting Task Families

The near uniqueness of Q^*-functions is encouraging, but max norm is not a very sophisticated metric. Here, we test whether a very simple multi-class classifier (1 hidden layer of 64 nodes), can accurately identify individual tasks based on their *Q-estimates*. Moreover, we track how the classification accuracy improves as a function of the number of steps taken within the MDP as the estimates improve. In this experiment, the same random policy is executed in each MDP for 50 time steps. As before, our MDPs have 3 states and 2 actions.

We instantiate 10,000 MDPs whose transition and reward functions are drawn from the same distribution as before: transitions from a Dirichlet distribution with $\alpha = 0.1$ and rewards sampled from a normal distribution $N(1, 0.5)$. Thus, this is a classification problem with 10,000 classes. *A priori*, this exercise seems relatively difficult given the number of tasks and the parameters chosen for the distributions. Fig. 7 shows a compelling result given the simplicity of the model and the relative difficulty of the classification problem. Clearly, Q-estimates, even those built from only 20 experiences, provide a high signal-to-noise ratio w.r.t. task identification. And this is for a *random* policy. In principle, the meta-RL agent could follow a much more deliberate policy that actively disambiguates trajectories such that the Q-estimates evolve in a way that leads to faster or more reliable discrimination.

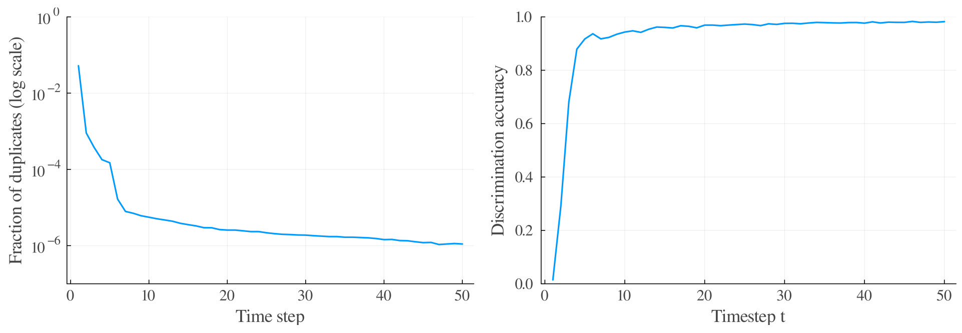


Figure 7: The task-identification power of Q-estimates. *Left*: Fraction of δ-duplicates, with $\delta = 0.1$, as a function of time steps in a set of 5,000 random MDPs. *Right*: Accuracy of a simple multi-class classifier in predicting task ID given Q-table estimates, as function of time step. Both figures are generated using the same policy.

E Domain Descriptions

E.1 Bernoulli Multi-Armed Bandits

We use the same setup described by Duan et al. (2016). At the beginning of each meta-episode, the success probability corresponding to each arm is sampled from a uniform distribution $\mathcal{U}(0, 1)$. To test OOD generalization, we sample success probabilities from $\mathcal{N}(0.5, 0.5)$

E.2 Random MDPs

We use the same setup described by Duan et al. (2016). The MDPs have 10 states and 5 actions. For each meta-episode, the mean rewards $R(s, a)$ and transition probabilities $T(s, a, s')$ are initialized from a normal distribution ($\mathcal{N}(1, 1)$) and a flat Dirichlet distribution ($\alpha = 1$), respectively. Moreover, when an action a is performed in state s, a reward is sampled from $\mathcal{N}(R(s, a), 1)$. To test OOD generalization, the transition probabilities are initialized with Dirichlet $\alpha = 0.25$.

Each episode begins at state $s = 1$ and ends after `task_horizon` = 10 time steps.

E.3 GridWorlds

A set of navigation tasks in a 2D grid environment. We experiment with 11x11 (121 states) and 13x13 (169 states) grids. The agent always starts in the center of the grid and needs to navigate through obstacles to a single goal location. The goal location is always at a minimum of `min_goal_manhat` Manhattan distance from the starting tile. The grid also contains slippery wet tiles, fatally dangerous tiles and warning tiles surrounding the latter. There are `num_obstacle_sets` set of obstacles, and each obstacle set spans `obstacle_set_len` tiles, in either horizontal or vertical configuration. There are `num_water_sets` set of wet regions and each wet region always spans `water_set_length`, in either a horizontal or vertical configuration. Entering wet tiles yields an immediate reward of -2. There are `num_dangers` danger tiles and entering them ends the episode and leads to a reward of -100. Warning tiles always occur as a set of 4 tiles non-diagonally surrounding the corresponding danger tiles. Entering warning tiles causes -10 reward. Normal tiles yield a reward of -1 to incentivize the agent to reach the goal quickly. On all tiles, there is a chance of slipping sideways with a probability of 0.2, except for wet tiles, where the probability of slipping sideways is 1.

The parameters for our canonical 11x11 and 13x13 GridWorlds are: `num_obstacle_sets` = 11, `obstacle_set_len` = 3, `num_water_sets` = 5, `water_set_length` = 2, `num_dangers` = 2, and `min_goal_manhat` = 8. The parameters for the OOD variations are largely the same and the differences are as follows. For Deterministic variation, the slip probability on non-wet tiles is 0. For Dense variation, `obstacle_set_len` is increased to 4. For Watery variation, `num_water_sets` is increased to 8. For Dangerous variation, `num_dangers` is increased to

Table 3: RL^2 /RL^3 Hyperparameters

Hyperparameter	**Value**
Learning Rate (Actor and Critic)	0.0003 (Bandits, MDPs)
	0.0002 (GridWorlds)
Adam $\beta1, \beta2, \epsilon$	$0.9, 0.999, 10^{-7}$
Weight Decay (Critic Only)	10^{-2}
Batch size	32768
Rollout Length	Interaction Budget (H)
Number of Parallel Envs	Batch Size $\div$ H
Minibatch Size	4096
Entropy Regularization Coeff	0.1 with decay (MDPs)
	0.04 (GridWorlds)
	0.01 (Bandits)
PPO Iterations	See training curves
Epochs Per Iteration	8
Max KL Per Iteration	0.01
PPO Clip ϵ	0.2
GAE λ	0.3
Discount Factor γ	0.99
Decoder Layers	2
Attention Heads	4
Activation Function	gelu
Decoder Size (d_model)	64

4. For CORNER variation, `min_goal_manhat` is set to 12, so that the goal is placed on one of the corners of the grid.

There is no fixed task horizon for this domain. An episode ends when the agent reaches the goal or encounters a danger tile. In principle, an episode can last through the entire meta-episode if a terminal state is not reached.

When a new grid is initialized at the beginning of each meta-episode, we ensure that the optimal, non-discounted return within a fixed horizon of 100 steps is between 50 and 100. This is to ensure that the grid both has a solution and the solution is not trivial.