

A Technical Appendices and Supplementary Material

A.1 Evidence lower bound (ELBO)

At a given time step t , the learning objective of the RNN encoder q_ϕ is to maximize

$$\mathbb{E}_{\rho(\tau_{0:t})}[\log p_\theta(o_{t+1}, r_{t+1} | a_{0:t})] \quad (1)$$

where $\rho(\tau_{0:t})$ is the distribution of trajectories generated by the policy π_ψ . Given that Eq. 1 is intractable, we can instead optimize a tractable evidence lower bound (ELBO), computed as:

$$\mathcal{L} = \mathbb{E}_{\rho(\tau_{0:t})} \left[\mathbb{E}_{\Pi_{t=0}^{T-1} q_\phi(m_t | \tau_{0:t})} \sum_{t=0}^{T-1} \left\{ \log p_\theta(o_{t+1} | m_t, a_t) + \log p_\theta(r_{t+1} | m_t, a_t) \right. \right. \\ \left. \left. - D_{KL}[q_\phi(m_t | \tau_{0:t}) || p_\theta(m_t)] \right\} \right] \quad (2)$$

The terms $\mathbb{E}_q[\log p_\theta(o_{t+1} | m_t, a_t) + \log p_\theta(r_{t+1} | m_t, a_t)]$ are the future predictive loss, and the term $D_{KL}[q_\phi(m_t | \tau_{0:t}) || p_\theta(m_t)]$ is a regularization term using Kullback-Leibler (KL) divergence between the variational posterior $q_\phi(m_t | \tau_{0:t})$ and the prior over the latent variables $p_\theta(m_t)$. To encourage the variational autoencoder (VAE) to approximate Bayesian filtering for belief update, the prior is set to the previous posterior $q_\phi(m_{t-1} | \tau_{0:t-1})$ with the initial prior $q_\phi(m_0 | \tau_0) = \mathcal{N}(0, I)$. To optimize the ELBO (Eq. 2), the expectation is approximated with Monte Carlo sampling as in standard VAE training [49].

A.2 Task details

Two-armed Bernoulli bandit We consider a two-armed Beta-Bernoulli bandit task with an episode length of 40 time steps. At the beginning of each episode, θ_1 and θ_2 , the reward probability biases for the two arms, are independently drawn from a fixed Beta distribution, $p(\theta_a) = \text{Beta}(1, 1)$. θ_1 and θ_2 are then used to define the Bernoulli reward distributions for arm $a \in \{1, 2\}$, respectively. The reward biases θ_a are hidden from the agent. At each time step, the agent chooses an arm $a \sim \pi$ sampled from its policy, and receives a binary reward sampled from $\text{Ber}(\theta_a)$, i.e. $r_t \sim p(r | \theta_a) = \text{Ber}(\theta_a)$. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. For multi-armed bandit tasks, Bayes-optimal policy can be derived using the Gittins index method [52], and the minimally sufficient Bayes-optimal belief state is to keep track of the count of total pulls and the count of rewarded pulls of each arm. In other words, for the two-armed bandit task, the Bayes-optimal belief states are 4-dimensional: $(n_{a_1}, n_{r_{a_1}}, n_{a_2}, n_{r_{a_2}})$, where n_a and n_{r_a} denote the count of total pulls and the count of rewarded pulls for arm a , respectively.

Dynamic two-armed bandit To evaluate the meta-learned representation and behavior when the underlying environmental state can change over time, we consider a dynamic version of the two-armed bandit. At a given time step, each arm is in one of n possible discrete states. For simplicity we choose $n = 2$. We denote the hidden state of arm a at time t as s_t^a , where $s_t^a \in \{\theta_1^a, \theta_2^a\}$, $\forall a \in \{1, 2\}$, with each state associated with a different reward probability, i.e., $r_t^a \sim p(r | s_t^a) = \text{Ber}(s_t^a)$. The state of each arm evolves according to an independent Markovian transition process independent of the action chosen: $p(s_{t+1}^a | s_t^a, a') = p(s_{t+1}^a | s_t^a) = T_{s_t^a, s_{t+1}^a}^a$. The episode length is set to be 300 time steps. To examine different structural and dynamical configurations, we consider three parameter settings:

- Symmetric reward and transition: the two arms share the same set of discrete states where $s_t^a \in \{0.1, 0.9\}$, $\forall a \in \{1, 2\}$, and the same transition dynamics where $T_{0.1, 0.1}^a = T_{0.9, 0.9}^a = 0.9$ and $T_{0.1, 0.9}^a = T_{0.9, 0.1}^a = 0.1$, $\forall a \in \{1, 2\}$. That is, for each arm, state 1 is highly rewarding whereas state 2 is less rewarding, and the state of each arm stays with a probability of 0.9 and switches with a probability of 0.1.
- Asymmetric reward: two arms share the same transition dynamics, but the reward probability states of the two arms are different. Specifically, the reward probability states for arm a_1

are $s_t^1 \in \{0.1, 0.9\}$, and for arm a_2 are $s_t^2 \in \{0.4, 0.6\}$, respectively. Therefore the two states of arm a_1 are more differentiating than those of arm a_2 . The transition dynamics is governed by $T_{0.1,0.1}^1 = T_{0.9,0.9}^1 = T_{0.4,0.4}^2 = T_{0.6,0.6}^2 = 0.9$, and $T_{0.1,0.9}^1 = T_{0.9,0.1}^1 = T_{0.4,0.6}^2 = T_{0.6,0.4}^2 = 0.1$. In other words, both arms follow the same dynamics that stay with a probability of 0.9 and switch with a probability of 0.1.

- **Asymmetric transition:** two arms share the same reward probability states $s_t^a \in \{0.1, 0.9\}, \forall a \in \{1, 2\}$, but the transition dynamics are different. For arm a_1 : $T_{0.1,0.1}^1 = T_{0.9,0.9}^1 = 0.9$ and $T_{0.1,0.9}^1 = T_{0.9,0.1}^1 = 0.1$; for arm a_2 , $T_{0.1,0.1}^2 = T_{0.9,0.9}^2 = T_{0.1,0.9}^2 = T_{0.9,0.1}^2 = 0.5$. That is, whereas the transition dynamics for arm a_1 is the same as previous scenarios, the transition dynamics for arm a_2 is random with equal stay and switch probabilities of 0.5.

With this task design, the belief state updates in the dynamic two-armed bandit POMDP tasks are analytically tractable by computing the posterior probability of each arm being in state 1 conditioned on the history: $b_t = (p(s_t^1 = \theta_1^1 | h_t), p(s_t^2 = \theta_1^2 | h_t))$. This belief update is tractable using standard Bayesian inference. In turn, the Bayes-optimal policy can be derived using the value iteration algorithm [11] by discretizing the 2-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$.

Stationary Tiger To exemplify sequential decision-making under uncertainty, we consider the classic POMDP Tiger task [3]. In the tiger environment, an agent chooses between two doors—one hiding a tiger (penalty=-100) and the other hiding a treasure (reward=10). The agent may additionally choose to pay a small penalty=-1 for the “listen” action to acquire noisy observations about the tiger’s location. The reliability of the noisy observation is controlled by the parameter of observation accuracy. To succeed, the agent must maintain a belief about the tiger’s location to decide which door to open. This simple yet powerful paradigm tests an agent’s ability to balance information gathering with reward-seeking, making it an ideal benchmark for evaluating RL algorithms in POMDP. We consider two difficulty levels by varying the observation accuracy for the “listen” action. Specifically, we consider a simpler task variant where the observation accuracy is 0.8 and a harder task variant where the observation accuracy is 0.7. For the Tiger tasks, the Bayes-optimal agent tracks the belief of the tiger’s location, for instance, by computing the posterior probability of the tiger being on the left given the history, $b_t = p(\text{tiger at the left} | h_t)$, using standard Bayesian inference. The Bayes-optimal policy can be derived using value iteration [11] by discretizing the 1-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. Typically, as the observation accuracy decreases, the Bayes-optimal solution will require listening for more times as observations are noisier before the belief states crosses the decision threshold for the optimal agent to decide on which door to open.

Dynamic Tiger We extend the classic Tiger task to a dynamic version by allowing the tiger’s location to change over time, following Markov transition dynamics. We denote the tiger location s to be in one of the 2 states $s \in \{L, R\}$, and the tiger location evolves according to an independent Markovian transition process independent of the action chosen: $p(s_{t+1} | s_t, a) = p(s_{t+1} | s_t) = T_{s_t, s_{t+1}}$. Specifically, we choose $T_{L,L} = T_{R,R} = 0.9$ and $T_{L,R} = T_{R,L} = 0.1$. In other words, at each time step, the tiger stays with a probability of 0.9 and switches its location with a probability of 0.1. As the information reliability and relevance are further corrupted by the dynamic nature of the hidden state, the task becomes more challenging because the agent has to balance listening more to increase its confidence against making decisions earlier in case the information gathered so far becomes obsolete. This task design permits tractable belief updates by tracking the posterior probability of the tiger being on the left given the history, $b_t = p(\text{tiger at the left} | h_t)$, using standard Bayesian inference incorporating the Markovian transition matrix. The Bayes-optimal policy can be derived using value iteration [11] by discretizing the 1-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. Similarly to the stationary Tiger tasks, we consider two difficulty levels by varying the observation accuracy to be 0.8 or 0.7.

Oracle bandit task The oracle bandit task is designed to exemplify an environment where a successful policy requires paying an immediate exploration cost and utilize the information acquired to improve long-term return. In an 11-arm bandit environment with an episode length of 6 time steps, one of the first ten arms a_{1-10} is selected uniformly randomly as the target arm a^* , which will give a

payout of 5 upon choosing. The other nine arms out of a_{1-10} are non-target arms, each of which will give a payout of 1 upon choosing. The last arm, a_{11} , is the oracle arm whose payout informs the index of the target arm in the form of $1/10$ of the target arm index a^* , i.e. $r(a_{11}) = 0.1 * a^*$ (e.g. a reward of 0.3 from a_{11} indicates that a_3 is the target arm). The average reward from the oracle arm is 0.55 which is smaller than the payout from the non-target arms. As a result, choosing the oracle arm is less favorable in terms of the immediate reward but provides useful information if an agent knows how to utilize to improve long-term return. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. Note that this setting is similar to a task considered in Wang et al. [7] but differs in that in their setup the oracle information was provided using a structured one-hot encoding format, but in our formulation no other feedback than the reward itself is given, which makes learning a good representation of the history more challenging and critical. With knowledge of the task structure, the Bayes-optimal belief state is to track and update the posterior probability of each of the first ten arms being the true target arm conditioned on the history, i.e. $b_t = (p(a^* = a_1|h_t), p(a^* = a_2|h_t), \dots, p(a^* = a_{10}|h_t))$. The Bayes-optimal policy is to pull the oracle arm at the beginning and continue pulling the target arm as informed by the reward information from the oracle arm until the end of the episode.

Latent goal cart Deriving Bayes-optimal solutions in continuous POMDPs are usually intractable, hindering rigorous representational equivalence analysis. To enable evaluation in continuous observation and action spaces, we design an exemplar continuous control task which still permits tractable Bayes-optimal belief inference and policy. In this task, an agent controls a continuous action, the velocity of a cart, to move along a 1-dimensional track to a hidden goal (+1 or -1) which needs to be inferred from the continuous observation (current position) and reward (negative noisified distance from the hidden goal, with the noise following a Gaussian distribution) it receives. We consider an episode length of 30 time steps. This task design allows for tractable belief updates by tracking the posterior probability of the hidden goal being at +1 given the history, $b_t = p(\text{goal at } +1|h_t)$, using standard Bayesian inference incorporating the knowledge that the noise in reward follows a Gaussian distribution. The Bayes-optimal policy can be derived using value iteration [11] by discretizing the 1-dimensional belief state space. The discounted cumulative return is computed with a discount factor $\gamma = 0.95$. This task is motivated by the Half-Cheetah-Dir task in MuJoCo, and can be seen as a simplified version to allow for tractable belief updates and value iteration to derive the Bayes-optimal solution.

A.3 Agent details

RL² The implementation of the baseline black-box memory-based meta-RL models, RL², follows an actor-critic architecture as in previous literature [6, 7]. Here we use RNNs that function as a memory module, consisting of 256 hidden units and hyperbolic tangent activation functions. As shown in Fig. 1B, the input includes the current observation o_t , the previous action in one-hot format a_{t-1} , and the associated reward in scalar format r_t . The recurrent layer is followed by a fully connected bottleneck layer designed to be the counterpart to the latent belief layer b_t in Fig. 1C. After the bottleneck layer is a hidden layer of 32 units and a readout layer that generates a vector of logits for each action a_t for the actor (or a vector that defines the mean and the standard deviation of a continuous Gaussian policy if solving a continuous task such as the Latent goal cart) and a scalar value baseline V_t for the critic. Actions are then sampled from the softmax distribution defined by the action logits. The network is trained end-to-end to maximize the discounted cumulative reward with the Advantage Actor Critic algorithm [50] (Parts of the implementation code are based on [53], under MIT license). The gradient of the objective function is given by:

$$\begin{aligned} \nabla \mathcal{L}_{A2C} &= \nabla \mathcal{L}_\pi + \nabla \mathcal{L}_V + \nabla \mathcal{L}_{entropy} \\ &= \frac{\partial \log \pi(a_t|\tau_{:t}; \psi)}{\partial \psi} \delta_t(\tau_{:t}; \psi_V) + \beta_V \delta_t(\tau_{:t}; \psi_V) \frac{\partial V}{\partial \psi_V} + \beta_e \frac{\partial H(\pi(a_t|\tau_{:t}; \psi))}{\partial \psi} \end{aligned} \quad (3)$$

where

$$\begin{aligned} \delta_t(\tau_{:t}; \psi_V) &= R_t - V(\tau_{:t}; \psi_V) \\ R_t &= \sum_{i=0}^{k-1} [\gamma^i r_{t+i} + \gamma^k V(\tau_{:t+k}; \psi_V)] \end{aligned} \quad (4)$$

defines the n -step temporal difference error advantage function δ_t , the discounted n -step bootstrapped return R_t with discount factor γ , k the number of remaining time steps in the current episode,

Table 2: **Hyperparameters**

	episode length	β_e	β_V	n_updates	bottleneck size
Two-armed bandit	40	0.01 – 0.05	0.01 – 0.05	3e5	8
Dynamic bandit	300	0.01	0.05	1e5	8
Stationary Tiger	20 – 30	0.3	0.1	3e5	4
Dynamic Tiger	30 – 40	0.3	0.1	3e5	4
Oracle bandit	6	0.3 – 0.5	0.01 – 0.05	3e6	16
Latent goal cart	30	0.005 – 0.01	0.01	3e5	8

and V the value function parametrized by ψ_V . The neural network policy is denoted as π and parametrized by ψ , and H_π is the entropy of the policy. Finally, β_V and β_e are hyperparameters for controlling the relative weighting of value estimation loss and entropy regularization. The choice of hyperparameters for each task is summarized in Table. 2. The neural network parameters are trained via backpropagation through time using the Adam Optimizer with a learning rate of 5e-5.

Meta-RL with predictive modules The self-supervised predictive modules are formulated as a VAE. For the encoder q_ϕ we use an RNN with 256 hidden units and hyperbolic tangent activation functions. The output of the encoder RNN, i.e. the bottleneck layer, is treated as estimating the mean and the variance of the latents m_t , as standard in VAE. Therefore, the latent dimension is half of the bottleneck layer size. The decoders R_θ and T_θ are multi-layered perceptrons (MLPs) with one hidden layer of 32 units and ReLU activation functions. As shown in Fig. 1C, the encoder-decoder framework takes as input the current observation o_t , the previous action in one-hot format a_{t-1} , and the associated scalar reward r_t , and is set up to make prediction of the upcoming observations o_{t+1} and rewards r_{t+1} , conditioned on the trajectory as incurred by the policy network π_ψ described below. The entire VAE is trained to maximize the ELBO (Eq. 2) as derived in A.1. A coefficient of 0.01 is used for the relative contribution of the KL-term for training the VAE. We use the Adam Optimizer with a learning rate of 7e-5 to train the VAE using backpropagation through time.

The policy network π_ψ is parametrized as an MLP with one hidden layer of 32 units and hyperbolic tangent activation functions. Similar to the previous paragraph on RL², the policy network is trained with the Advantage Actor Critic algorithm to optimize the same loss function as described in Eq. 3. The choice of hyperparameters for each task is summarized in Table. 2. An Adam Optimizer with a learning rate of 5e-5 is used to optimize the policy network. Note although the policy loss depends on the parameters of the encoder q_ϕ , we do not backpropagate the policy loss gradient through the encoder as the goal of the encoder is to learn a belief b_t over the latent states predictive of the future such that the belief alone should be a sufficient representation for policy learning (similar to Zintgraf et al. [9]). Parts of the implementation code are based on [9] (under MIT license).

Model training The above meta-RL models (RL² and ours, meta-RL with predictive modules) are trained on internal GPU clusters (NVIDIA GeForce RTX 4090), which takes less than 1G GPU memory and between 10-48 hours for training per model, depending on the task. Compared to RL², training time of meta-RL with predictive modules increases by ~ 30 -40% with the overhead coming from two additional decoders and VAE training, whereas inference time is comparable.

Ablation study To pinpoint which algorithmic design choices drive enhanced representation learning in our proposed meta-RL with predictive modules, we performed two targeted ablations:

- (i) no KL: where we remove the VAE’s KL regularization to test whether latent regularization is necessary to enforce compact representations.
- (ii) joint RL: where we allow the policy gradients from RL loss to jointly train the RNN encoder alongside the predictive loss—rather than training the encoder *solely* with the predictive loss—to assess whether policy gradients provide additional representational benefits.

A.4 State machine simulation

Following the procedure of state machine simulation introduced in Mikulik et al. [13], we consider whether a meta-RL system (RL² or our proposed approach, meta-RL with predictive modules) can both *simulate* and *be simulated by*, a Bayes-optimal agent for a given POMDP task.

To evaluate how well a state machine M *simulates* another machine N , a function ϕ is first learned to map the states S_N in N into the state space S_M of M . As enumeration over all possible trajectories are not practical if not possible, quality of a simulation is measured along trajectories sampled from some reference distribution. Given trajectories from a reference distribution, quality of the simulation is then measured by (i) the *state-transition dissimilarity* D_s , measured as the mean-squared error (MSE) between the embedded states $\phi(S_N)$ and the target states S_M , and (ii) the *output dissimilarity* D_o , measured as the difference in the expected return generated from the states S_N using the machine N and those generated from the states $\phi(S_N)$ using the machine M . If both the state-transition and output dissimilarities D_s and D_o are low/ negligible, then we establish that M simulates N . If both M simulates N and N simulates M , then we can say M and N are computationally equivalent, and their states S_N and S_M are equivalent.

In practice, the mapping function ϕ is implemented as an MLP with ReLU activations and three hidden layers of 64, 128, and 64 units, respectively. The MLP is trained with the Adam Optimizer with learning rate 0.001 and batch size 64. The training set is consisted of 300 – 1000 trajectories depending on the variability of each task distribution, and the results reported are from another test set of 300 – 1000 trajectories. Compared with Mikulik et al. [13], where the reference distribution is generated by the meta-RL agent, here we modify the procedure by generating the reference distribution using the Bayes-optimal agent, as this provides an even more stringent condition where the simulation is evaluated in the regime of Bayes-optimal solutions.

When evaluating how well a Bayes-optimal agent *simulates* a meta-RL system, we first train an MLP mapping meta-RL states into Bayes-optimal states by minimizing the MSE between the mapped states and the target Bayes-optimal states. After training, quality of the simulation is measured on a test set by evaluating the state-transition dissimilarity (D_s : metaRL→Bayes) and the output dissimilarity (D_o : metaRL→Bayes). If both D_s : metaRL→Bayes and D_o : metaRL→Bayes are low after training, then the Bayes-optimal agent *simulates* the meta-RL agent.

On the other hand, to evaluate how well a Bayes-optimal agent *is simulated by* a meta-RL one, an MLP is trained to map Bayes-optimal states into meta-RL states, and the state-transition dissimilarity is denoted D_s : Bayes→metaRL and the output dissimilarity denoted D_o : Bayes→metaRL. If both D_s : Bayes→metaRL and D_o : Bayes→metaRL are low after training, then the Bayes-optimal agent *is simulated by* the meta-RL agent.

If all the above four dissimilarity measures are low/ negligible, then we can say that the meta-RL agent and the Bayes-optimal agent are computationally equivalent and their representations are equivalent. Our results in this paper demonstrate that after training the proposed meta-RL with predictive coding modules can attain much lower dissimilarities than conventional RL², indicating that meta-RL with predictive modules can more closely approximate the Bayes-optimal belief states.

Note that before training, the state dissimilarity D_s can be low for the recurrent layers in all models, similar to what Mikulik et al. [13] reported and discussed—untrained RNNs may maintain a verbose representation of the history by embedding each trajectory into a unique hidden state, which can be subsequently mapped to minimally sufficient Bayes-optimal statistic with low error using expressive enough functions, like the MLPs used in the above state machine simulation procedure. This observation also highlights that using decoding alone may not provide a thorough assessment of representation equivalence, and we need to consider their structural and computational relevance when comparing representations.

A.5 Additional results

A.5.1 State space visualization

To understand the structure of the learned representations in meta-RL agents and qualitatively compare those to the Bayes-optimal belief states, visualization of example state spaces for the stationary and dynamic bandit tasks, the stationary and dynamic Tiger tasks, the oracle bandit tasks, and the latent goal cart tasks are presented in Figures. [8](#), [9](#), [10](#), and [11](#) respectively.

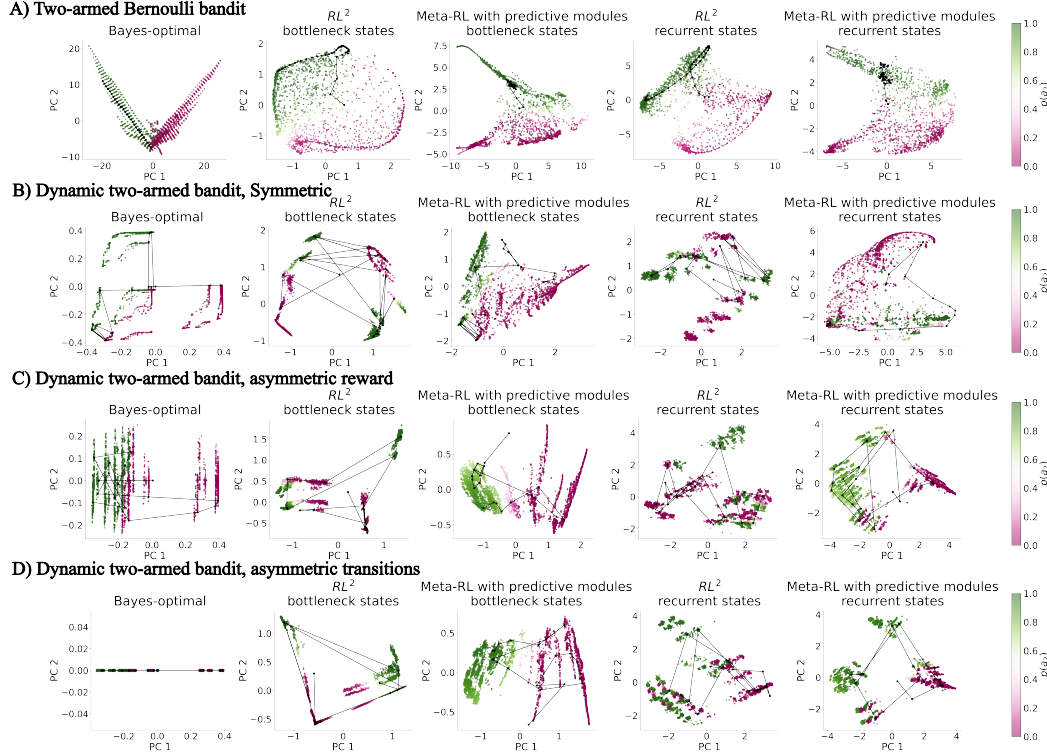


Figure 8: Visualization of state space in the stationary and dynamic two-armed bandit tasks. Example state space for A) Two-armed Bernoulli bandit task, B) Dynamic two-armed bandit task with symmetric reward and transition, C) Dynamic two-armed bandit task with asymmetric reward, and D) Dynamic two-armed bandit task with asymmetric transition. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, i.e. probability of choosing a_2 . Black curves show one example trajectory. For each task, from the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

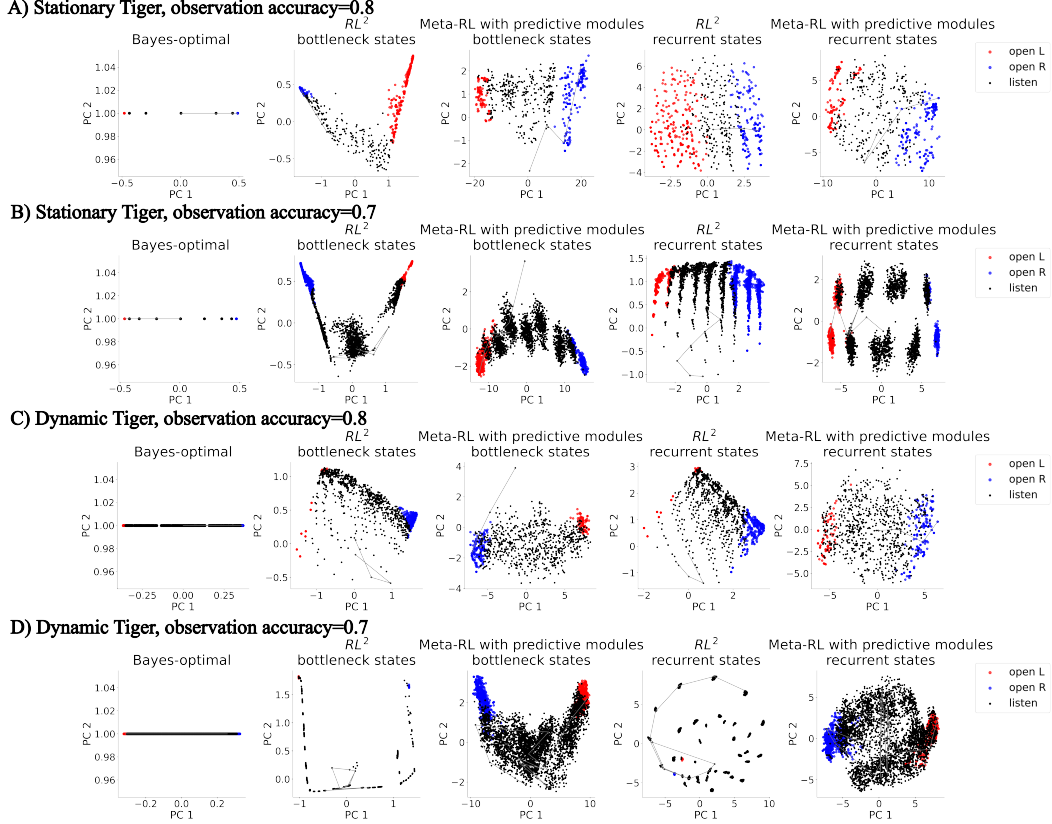


Figure 9: Visualization of state space in the stationary and dynamic Tiger tasks. Example state space for A) Stationary Tiger task with observation accuracy of 0.8, B) Stationary Tiger task with observation accuracy of 0.7, C) Dynamic Tiger task with observation accuracy of 0.8, and D) Dynamic Tiger task with observation accuracy of 0.7. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, with black denoting choosing listen, red choosing to open the left door, and blue choosing to open the right door. Black curves show one example trajectory. For each task, from the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

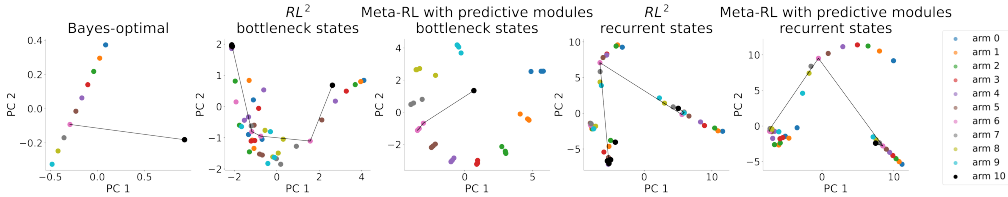


Figure 10: Visualization of state space in the oracle bandit task. Example state space for the oracle bandit task. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, with black denoting choosing the oracle arm a_{11} and other colors denoting choosing one of the first ten arms a_1 – a_{10} . Black curves show one example trajectory. From the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

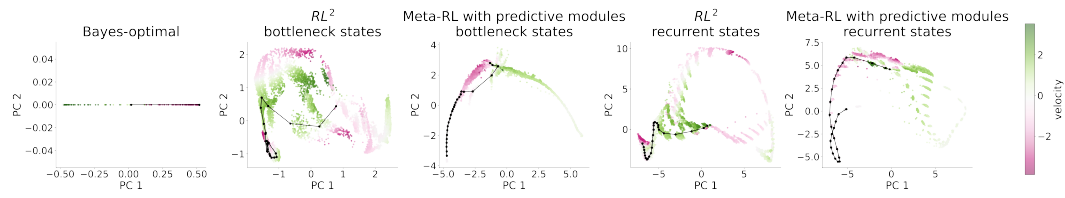
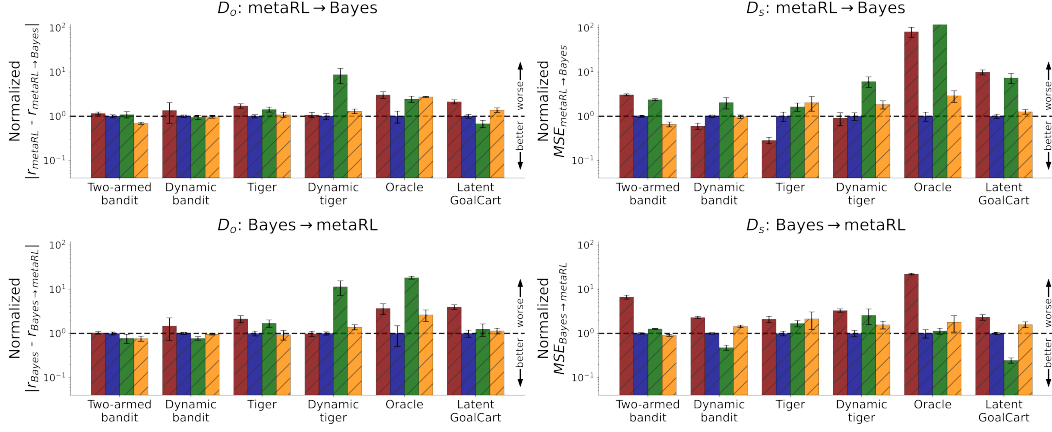


Figure 11: **Visualization of state space in the latent goal cart task.** Example state space for the oracle bandit task. For each panel, the first two principal components are plotted. States are colored by the corresponding policy, i.e. the velocity of the cart. Black curves show one example trajectory. From the left to the right are the state space of the Bayes-optimal agent (i.e. the belief states), the bottleneck layer in the RL^2 model, the bottleneck layer in the meta-RL with predictive modules, the recurrent layer in the RL^2 model, and the recurrent layer in the meta-RL with predictive modules, respectively.

A.5.2 Full state machine simulation results

Full results of state machine simulation analysis on the trained models of RL² and the proposed meta-RL with predictive modules (ours), together with models considered in the ablation studies—no KL and joint RL, are summarized in Fig. 12 and Tables 3, 4, 5, and 6.

A) Bottleneck layer



B) Recurrent layer

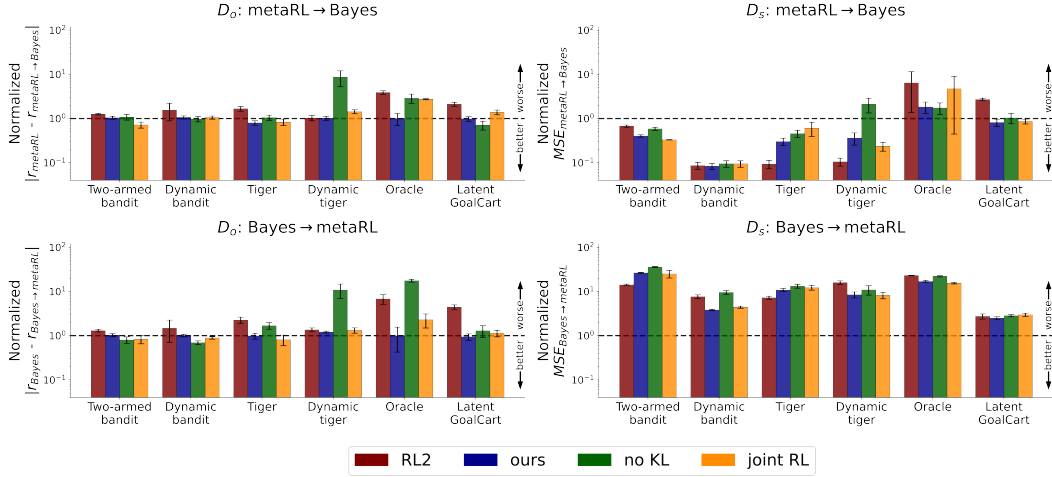


Figure 12: Results of state machine simulation. State and output dissimilarities (D_o and D_s) for both mapping directions (meta-RL \rightarrow Bayes and Bayes \rightarrow meta-RL) across all tasks considered, using A) the bottleneck layer and B) the recurrent layer of the fully-trained models of RL² (red), meta-RL with predictive modules (ours, blue), no KL (where KL regularization is ablated for VAE training, green), and joint RL (where the policy gradient of RL loss is used to jointly train the RNN encoder with the predictive loss, orange). To pool across different variants of the same task type, all dissimilarity measures are normalized by the mean of the corresponding dissimilarities of the bottleneck layer in the meta-RL with predictive modules models (denoted as *ours* in the legend.) In other words, when normalized with the mean bottleneck dissimilarity of our model, values significantly greater than 1 indicate that the representation is worse than the bottleneck layer in our model (i.e. deviating more from the Bayes-optimal belief states). In contrast, values significantly smaller than 1 indicate the representation is better than the bottleneck layer in our model (i.e. more closely approximating the Bayes-optimal belief states). Error bars denote s.e.m. across at least five random seeds per model type.

Table 3: **State machine simulation results for the stationary and dynamic bandit tasks:** State and output dissimilarities of the fully-trained models for both mapping directions. M for meta-RL, B for Bayes-optimal solutions. Values indicate mean \pm s.t.d. Bold indicates the best dissimilarities among the trained models and better than untrained models ($p < 0.05$).

Two-armed Bernoulli bandit (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
Bottleneck				Recurrent				
$D_o^{M \rightarrow B}$	1.52 \pm 0.36	1.31 \pm 0.37	1.44 \pm 0.4	0.91\pm0.05	1.65 \pm 0.24	1.37 \pm 0.38	1.42 \pm 0.36	0.94\pm0.15
$D_s^{M \rightarrow B}$	5.41 \pm 0.96	1.77 \pm 0.33	4.21 \pm 0.38	1.16 \pm 0.15	1.19 \pm 0.23	0.71 \pm 0.16	1.04 \pm 0.14	0.58 \pm 0.01
$D_o^{B \rightarrow M}$	1.19\pm0.25	1.15\pm0.31	0.89\pm0.36	0.88\pm0.12	1.51\pm0.35	1.19\pm0.34	0.93\pm0.27	0.95\pm0.2
$D_s^{B \rightarrow M}$	5.38 \pm 1.81	0.81\pm0.11	1.02 \pm 0.04	0.73\pm0.04	11.43 \pm 1.83	21.37 \pm 2.12	29.05 \pm 1.43	20.33 \pm 3.96
Dynamic two-armed bandit: symmetric reward and transition (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
Bottleneck				Recurrent				
$D_o^{M \rightarrow B}$	2.09\pm3.95	1.17\pm0.18	0.86\pm0.48	1.06\pm0.1	2.18\pm3.92	1.25\pm0.21	0.86\pm0.43	1.08\pm0.12
$D_s^{M \rightarrow B}$	0.19 \pm 0.02	0.21 \pm 0.05	0.49 \pm 0.36	0.22 \pm 0.1	0.03 \pm 0.01	0.03 \pm 0.01	0.03 \pm 0.02	0.04 \pm 0.01
$D_o^{B \rightarrow M}$	2.06\pm3.81	0.98\pm0.36	0.79\pm0.33	0.89\pm0.19	2.18\pm3.75	1.05\pm0.44	0.72\pm0.3	0.82\pm0.3
$D_s^{B \rightarrow M}$	2.38 \pm 0.6	1.09 \pm 0.22	0.45\pm0.13	1.43 \pm 0.23	10.26 \pm 2.01	4.86 \pm 0.44	10.25 \pm 2.37	6.11 \pm 0.69
Dynamic two-armed bandit: asymmetric reward (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
Bottleneck				Recurrent				
$D_o^{M \rightarrow B}$	0.8\pm0.28	0.86\pm0.27	0.76\pm0.19	0.82\pm0.24	1.01\pm0.14	1.22\pm0.34	0.59\pm0.13	1.04\pm0.34
$D_s^{M \rightarrow B}$	0.34 \pm 0.06	0.93 \pm 0.11	0.4 \pm 0.17	1.04 \pm 0.16	0.04 \pm 0.01	0.05 \pm 0.0	0.03 \pm 0.01	0.04 \pm 0.01
$D_o^{B \rightarrow M}$	1.15\pm0.27	1.39\pm0.03	0.99\pm0.37	1.37\pm0.16	1.03\pm0.31	1.37\pm0.03	0.97\pm0.34	1.28\pm0.35
$D_s^{B \rightarrow M}$	3.49 \pm 0.51	1.45\pm0.1	0.99\pm0.44	1.98 \pm 0.33	10.47 \pm 3.6	4.91 \pm 0.75	18.2 \pm 5.4	5.56 \pm 0.6
Dynamic two-armed bandit: asymmetric transition (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
Bottleneck				Recurrent				
$D_o^{M \rightarrow B}$	0.85\pm0.23	0.93\pm0.33	1.11\pm0.47	0.98\pm0.39	1.22\pm0.33	0.78\pm0.29	1.39\pm0.56	0.95\pm0.4
$D_s^{M \rightarrow B}$	0.04 \pm 0.02	0.33 \pm 0.11	1.1 \pm 1.06	0.24 \pm 0.09	0.01 \pm 0.0	0.01 \pm 0.01	0.03 \pm 0.01	0.02 \pm 0.01
$D_o^{B \rightarrow M}$	1.52\pm0.1	1.7\pm0.17	1.36\pm0.46	1.67\pm0.12	1.21\pm0.39	1.64\pm0.19	1.12\pm0.5	1.55\pm0.16
$D_s^{B \rightarrow M}$	5.14 \pm 0.75	2.15 \pm 0.45	0.7\pm0.39	3.56 \pm 1.08	9.0 \pm 1.06	7.41 \pm 0.91	14.35 \pm 5.85	8.16 \pm 1.69

Table 4: State machine simulation results for the stationary and dynamic Tiger tasks.

Stationary Tiger: observation accuracy 0.8 (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M \rightarrow B}$	3.33±0.91	2.57±1.01	2.93±1.09	2.97±0.87	3.05±0.64	2.07±1.03	2.13±0.37	2.6±1.22
$D_s^{M \rightarrow B}$	0.08±0.05	0.31±0.33	0.46±0.3	0.67±0.87	0.03±0.03	0.06±0.04	0.11±0.04	0.23±0.27
$D_o^{B \rightarrow M}$	3.03±1.54	2.26±0.88	2.98±1.7	1.88±1.24	4.15±1.72	1.85±1.22	2.98±1.7	1.34±0.92
$D_s^{B \rightarrow M}$	8.04±4.38	2.92±1.84	4.1±1.3	9.07±10.54	24.95±7.07	28.31±8.99	35.61±8.74	35.95±12.55
Stationary Tiger: observation accuracy 0.7 (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M \rightarrow B}$	3.25±1.13	1.57±0.49	2.68±1.0	1.58±0.74	3.27±1.14	1.24±0.62	2.0±0.9	1.02±0.36
$D_s^{M \rightarrow B}$	0.09±0.05	0.3±0.27	0.53±0.39	0.58±0.44	0.03±0.01	0.13±0.09	0.17±0.1	0.15±0.08
$D_o^{B \rightarrow M}$	3.06±1.54	1.09±0.73	2.25±1.34	1.13±0.85	2.86±1.67	1.33±0.76	2.25±1.1	1.13±0.85
$D_s^{B \rightarrow M}$	4.51±0.96	2.97±0.78	5.81±3.41	3.46±1.53	18.18±1.76	36.56±11.68	42.07±15.13	36.21±14.37
Dynamic Tiger: observation accuracy 0.8 (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M \rightarrow B}$	2.19±0.96	2.39±0.41	3.5±1.35	3.48±1.18	2.25±0.95	2.16±0.7	3.1±0.22	3.93±1.03
$D_s^{M \rightarrow B}$	0.04±0.01	0.1±0.05	0.31±0.17	0.29±0.06	0.01±0.0	0.04±0.04	0.12±0.09	0.02±0.02
$D_o^{B \rightarrow M}$	2.01±0.93	1.99±0.45	3.48±0.6	3.28±1.44	3.28±0.59	2.34±0.42	3.21±0.76	3.28±1.26
$D_s^{B \rightarrow M}$	6.27±1.15	1.78±0.98	2.9±1.76	2.03±0.88	29.5±2.07	14.39±4.93	24.2±10.58	11.82±5.04
Dynamic Tiger: observation accuracy 0.7 (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M \rightarrow B}$	2.51±0.66	2.05±1.28	27.92±21.09	2.33±1.01	2.3±0.67	2.37±0.53	27.75±21.84	2.51±0.95
$D_s^{M \rightarrow B}$	0.39±0.23	0.28±0.17	2.23±1.53	0.2±0.18	0.03±0.02	0.08±0.06	0.74±0.77	0.07±0.06
$D_o^{B \rightarrow M}$	1.73±0.32	1.8±0.47	32.11±22.34	2.05±0.57	1.95±0.31	2.27±0.17	30.65±20.94	1.8±0.65
$D_s^{B \rightarrow M}$	7.1±1.89	2.3±0.72	7.37±8.41	4.62±2.85	35.44±12.21	20.65±10.63	21.05±19.62	22.64±12.53

Table 5: State machine simulation results for the oracle bandit tasks.

Oracle bandit (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M \rightarrow B}$	17.21±9.73	5.64±4.5	13.79±4.44	15.48±1.02	21.96±6.65	5.64±4.5	16.26±7.76	15.48±1.02
$D_s^{M \rightarrow B}$	6.45±5.14	0.08±0.05	24.4±6.94	0.23±0.14	0.51±1.27	0.14±0.11	0.14±0.08	0.37±0.68
$D_o^{B \rightarrow M}$	12.01±10.45	3.26±4.26	59.34±8.43	8.57±5.0	22.18±17.23	3.24±4.9	57.34±8.39	7.49±5.26
$D_s^{B \rightarrow M}$	51.29±6.73	2.33±1.34	2.64±0.82	4.18±3.38	53.72±2.63	39.33±5.68	51.5±3.42	36.02±3.26

Table 6: State machine simulation results for the latent goal cart tasks.

Latent goal cart (unit: 1e-2)								
	RL ²	ours	no KL	joint RL	RL ²	ours	no KL	joint RL
	Bottleneck				Recurrent			
$D_o^{M \rightarrow B}$	33.68±11.84	15.83±3.80	10.73±4.94	21.86±8.88	33.37±11.44	15.56±4.61	11.20±6.19	21.93±9.05
$D_s^{M \rightarrow B}$	2.35±1.02	0.24±0.07	1.75±1.10	0.30±0.12	0.64±0.14	0.19±0.09	0.25±0.16	0.20±0.08
$D_o^{B \rightarrow M}$	31.67±13.36	8.01±3.62	9.96±7.73	9.14±4.47	35.42±15.36	7.56±3.07	10.39±7.69	9.08±5.10
$D_s^{B \rightarrow M}$	30.51±14.64	13.16±1.84	3.22±1.09	20.90±10.58	36.02±16.02	32.78±6.28	37.38±5.29	39.16±10.85

A.5.3 Sensitivity analysis

To evaluate whether representational equivalence between Bayes-optimal solutions and meta-RL with predictive modules is sensitive to the choice of bottleneck dimensions, using the oracle bandit task as an example, we performed systematic evaluation of state machine simulation while varying the bottleneck dimension ranging from 1 to 32. As shown in Fig. 13, when the dimension of the bottleneck is greater than task complexity (i.e. the belief state dimension for the task, e.g. > 2 for the oracle bandit task), the bottleneck of the meta-RL with predictive modules achieve similarly low dissimilarities, indicating that regularization helps to maintain interpretable belief representation even when the bottleneck is more than expressive enough. In contrast, when the dimension of the bottleneck is smaller than task complexity, both performance (suboptimal return) and representational equivalence (high dissimilarity) significantly degrade, indicating that the bottleneck dimension should be selected to be at least surpassing the task complexity in order to learn the task effectively.

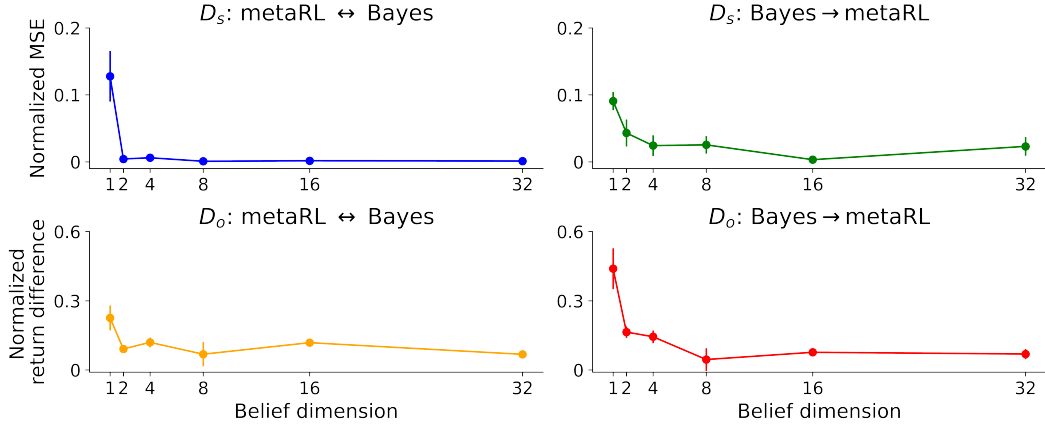


Figure 13: **Sensitivity analysis.** State machine simulation results of the bottleneck layers in meta-RL with predictive modules in the Oracle bandit task, as the belief dimension varies from 1 to 32. When the belief dimension is smaller than task complexity (e.g. ≤ 2), the models present with high dissimilarities in all four metrics. In contrast, as long as the belief dimension is greater than task complexity, all four dissimilarities remain consistently low even when the belief dimension is large, indicating that regularization is useful for maintaining compact, interpretable belief representation.