3rd Symposium on Advances in Approximate Bayesian Inference, 2020 1–14

Learning Discrete State Abstractions With Deep Variational Inference

Anonymous Authors
Anonymous Institution

1. Introduction

High-dimensional state spaces are common in deep reinforcement learning (Mnih et al., 2015). Although states may be as large as images, typically the information required to make good decisions is much smaller. This motivates the need for *state abstraction*, the process of encoding states into compressed representations that retain features that inform action and discard uninformative features.

One principled approach to state abstraction is *bisimulation* in Markov decision processes (MDP) (Dean and Givan, 1997). Bisimulations formalize the notion of finding a smaller equivalent *abstract* MDP that preserves transition and reward information; i.e., it retains relevant decision-making information while reducing the state space size. We demonstrate this idea in Figure 1, where a grid world with fifteen states is compressed into an MDP with three states.

In this paper, we introduce an approach to finding approximate MDP bisimulations using the variational information bottleneck (VIB, Tishby et al. (2001); Alemi et al. (2017)). The VIB framework is typically used to learn representations that predict quantities of interest accurately while ignoring certain aspects of the domain.

VIB methods have previously been applied to state abstraction, but the learned abstraction does not in general take the form of an MDP bisimulation (Abel et al., 2019). This is problematic, because the abstract MDP can only represent the policies it was trained on, but cannot be used to plan for new tasks. To resolve this, whereas Abel et al. (2019) use the abstract states to predict *actions* from an expert policy, we use abstract states to predict learned *Q-values* in the VIB objective.

In our setup, a learned encoder maps a state s in the original MDP into a continuous embedding z. We map the continuous state z onto a discrete abstract state $\bar{s}$ by performing inference in a learned probabilistic model. Our VIB method learns an encoder (i.e. a state abstraction $s \mapsto z$) that is predictive of the Q-values that are returned by a deep Q-network, but is regularized using structured prior over the embedding space (z). Concretely, we propose using priors that prefer clusters with Markovian transition structure. A sequence of embedded states $(z_1, z_2, \ldots)$ is treated as observations from either a Gaussian mixture model (GMM) or an action-conditioned hidden Markov model (HMM), where each embedding z_t is emitted from a latent cluster representing abstract state $\bar{s}_t$. In the HMM case, we also learn a cluster transition matrix for each action, serving as the abstract MDP transition model. The key insight is that abstract states $\bar{s}$ group together ground states s (and embeddings z) with similar Q-values and similar transition properties, thereby forming an approximate MDP bisimulation.

© A. Authors.

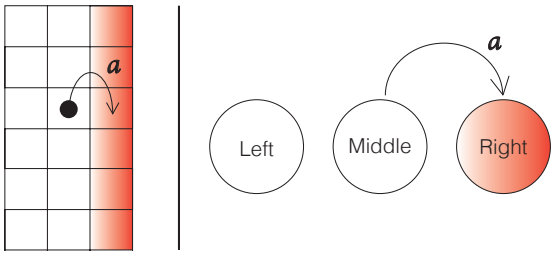

Figure 1: The Column World (left) has 3 columns and 30 rows (we only show 6 rows); the agent travels between adjacent cells. Since the agent gets a reward 1 for being in the right column (red) and 0 otherwise, it is irrelevant in which row it is located. Hence, the environment can be simulated by an MDP with three states (right).

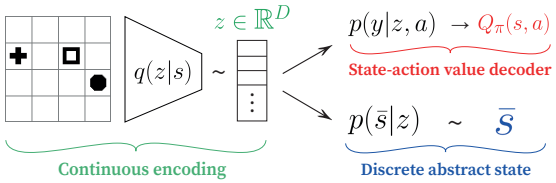

Figure 2: During inference, our model first takes a state, represented as an image here, and encodes it as a continuous vector z (green). Then, we predict state-action values from z for each action a (red) and encode z into a discrete abstract state $\bar{s}$ using our prior (blue).

In addition to the neural encoder, the parameters of our GMM and HMM priors are learned as well. The learned parameters (cluster means, covariances, and discrete transition matrix between clusters) therefore form our abstract MDP state space and transition function. When presented with tasks not seen during training, we can use the learned abstract model to plan to solve these tasks without additional learning efficiently.

2. Learning bisimulations

We propose a variational method for finding bisimulations directly from experience. The end result of our process is an abstract MDP, in which we can efficiently plan policies. Our model consists of three parts:

1. A deep neural encoder $q(z \mid s)$ that projects states (usually represented as images) onto a low-dimensional continuous latent space (Figure 2 left).
2. A generative model $p(z, \bar{s})$ with a tractable posterior $p(\bar{s} \mid z)$, that encodes the prior belief that the experience was generated by a small discrete Markov process (Figure 2 lower right).
3. A linear decoder $p(y \mid z, a)$ that predicts state-action values from the continuous encodings (Figure 2 upper right).

We tie the three models together using the deep variational information bottleneck method (Alemi et al., 2017).

2.1. Bisimulation as an information bottleneck

We apply deep IB methods to learn bisimulations as follows. Let $q(s_t, a, y, s_{t+1})$ be a empirical distribution representing a dataset of transitions from state s_t to state s_{t+1} under an action a, selected by an arbitrary policy π. $y = Q_\pi(s_t, a)$ denotes the state-action value

for the pair (s_t, a) under π. We will use the IB method to find a compact latent encoding of s_t that enables us to predict y while simultaneously matching a prior on the temporal dynamics of the process that generated the data. Let $s = (s_t, s_{t+1})$ denote a sequential pair of states and $z = (z_t, z_{t+1})$ a corresponding sequential pair of latent states. The standard IB formulation is:

$$\begin{aligned} R_{IB} &= \mathbb{E}_{q(s,a,y,z)}\big[I(y;z|a) - \beta I(s;z|a)\big] \\ &\geq \mathbb{E}_{q(s,a,y,z)}\Big[\log p(y|z,a) - \beta \log \frac{q(z|s,a)}{p(z|a)}\Big] \\ &\geq \mathbb{E}_{q(s,a,y,z)}\Big[\log p(y|z_t, z_{t+1}, a) - \frac{q(z_t, z_{t+1}|s_t, s_{t+1}, a)}{p(z_t, z_{t+1}|a)}\Big], \end{aligned} \tag{1}$$

where use $q(s,a,y,z) = q(s_t, a, y, s_{t+1})q(z|s,a)$ as shorthand notation and expand $s = (s_t, s_{t+1})$ and $z = (z_t, z_{t+1})$ in the last identity.

We make two architectural decisions grounded in standard Markov assumptions. First, we assume that the value y is conditionally independent of z_{t+1} given z_t: $q(y|z_t, z_{t+1}, a) = q(y|z_t, a)$. Second, we assume that z_t is conditionally independent of z_{t+1}, s_{t+1}, and a given s_t and likewise that z_{t+1} is conditionally independent of z_t, s_t, and a: $q(z_t, z_{t+1}|s_t, s_{t+1}, a) = q(z_t|s_t)q(z_{t+1}|s_{t+1})$. Together, these two assumptions yield an IB lower bound of the form

$$L_{IB} = \mathbb{E}_{q(s,a,y,z)}\Big[\log p(y|z_t, a) - \beta \log \frac{q(z_t|s_t)q(z_{t+1}|s_{t+1})}{p(z_t, z_{t+1}|a)}\Big]. \tag{2}$$

L_{IB} presents a trade-off between encoding enough information of s_t in order to predict y (the first term of Equation 2) and making the sequence (z_t, z_{t+1}) likely under our prior (the second term). This prior, $p(z_t, z_{t+1}|a)$, is a key element of our approach and is discussed in the next section.

Notice that $\log p(y \mid z_t, a)$ (the first term in Equation 2) predicts a state-action value, not a reward. This provides additional supervision. Without it, the model tends to collapse all states where the agent does not receive a reward (if the reward function is sparse) into a single abstract state.

2.2. Structured Priors

The denominator of the second term in Equation 2 $p(z_t, z_{t+1}|a)$ is the prior. We use this prior to incorporate an inductive bias, which is that we are observing a discrete Markov process. We evaluate two priors for this purpose: a prior based on a Gaussian mixture model and one based on an action-conditioned Hidden Markov model.

2.2.1. GMM Prior

We assume K components, each parameterized by a mean μ_k and a covariance Σ_k

$$p_{\text{GMM}}(z_t) = \sum_{k=1}^{K} p(z_t|c_t = k)p(c_t = k) \tag{3}$$

$$= \sum_{k=1}^{K} \mathcal{N}(z_t|\mu_k, \Sigma_k)\rho_k. \tag{4}$$

Here $\rho_k = p(c_t = k)$ denotes the probability that z_t was generated by component k and $\mathcal{N}(z_t|\mu_k, \Sigma_k) = p(z_t|c_t = k)$ is the Gaussian distribution for the kth component. In this paper, we constrain Σ_k to be diagonal. For the GMM prior, we set $p(z_t, z_{t+1}|a) = p_{\text{GMM}}(z_t)$ and allow μ_k and Σ_k to vary; ρ_k is uniform and fixed. This encodes a desire to find a latent encoding generated by membership in a finite set of discrete states (the mixture components). Each mixture component corresponds to a distinct abstract state. The weighting function ρ_k is the probability that the continuous encoding z_t was generated by the kth abstract state.

2.2.2. HMM Prior

To capture the temporal aspect of a Markov process, we can model the prior as an action conditioned hidden Markov model (an HMM). Here, the "hidden" state is the unobserved discrete abstract state c_t used to generate "observations" of the latent state z_t. As in the GMM, there are K discrete abstract states, each of which generates latent states according to a multivariate Normal distribution with mean μ_k and (diagonal) covariance matrix Σ_k. Since we are modelling a Markov process, we include a separate transition matrix T^a for each action a where $T^a_{k,l}$ denotes the probability of transitioning from an abstract state k to an abstract state l under an action a. Using this model, the prior becomes:

$$p_{\text{HMM}}(z_t, z_{t+1}|a) = \sum_{k=1}^{K}\sum_{l=1}^{K} p(z_t|c_t = k)p(z_{t+1}|c_{t+1} = l)p(c_{t+1} = l|c_t = k, a)p(c_t = k) \quad (5)$$

$$= \sum_{k=1}^{K}\sum_{l=1}^{K} \mathcal{N}(z_t|\mu_k, \Sigma_k)\mathcal{N}(z_{t+1}|\mu_l, \Sigma_l)T^a_{k,l}\rho_k \quad (6)$$

As with the GMM prior, we allow the parameters of this model (μ_k, Σ_k, and T^a) to vary, except for ρ_k, which is uniform and fixed. The transition model T^a found during optimization is a discrete conditional probability table that defines a discrete abstract MDP. Essentially, this method finds the parameters of a hidden discrete abstract MDP that fits the observed data over which the loss of Equation 2 is evaluated.

2.3. Deep encoder and end-to-end training

The lower bound L_{IB} (Equation 2) is defined in terms of three distributions that we need to parameterize: the encoder $q(z|s)$, the Q-predictor $p(y|z_t, a)$ and the prior $p(z_t, z_{t+1}|a)$. The encoder $q(z|s)$ is a convolutional network that predicts the mean $\mu^{CNN}(s_t)$ and the diagonal covariance $\Sigma^{CNN}(s_t)$ of a multivariate normal distribution. We used a modified version of the encoder architecture from (Ha and Schmidhuber, 2018)[1]: five convolutional layers followed by one fully-connected hidden layers and two fully-connected heads for μ^{CNN} and Σ^{CNN}, respectively. The Q-predictor $p(y|z_t, a)$ is a single fully-connected layer (i.e. a linear transform). We chose this parameterization to impose another constraint on the latent space: the encodings z not only need to form clearly separable clusters to adhere to the prior, but also linearly dependent on their state-action values for each action. When we train on state-action values for multiple tasks, we predict a vector y instead of a scalar. Using the reparameterization trick to sample from $q(z|s)$, we can compute the gradient of the objective

1. Appendix A.1. in version 4 of their arXiv submission.

with respect to the encoder weights, Q-predictor weights and the prior parameters. The prior parameters include the component means and variance, together with the transition function for hidden states in the HMM.

2.4. Planning in the Abstract MDP

A key aspect of our approach is that we can solve new tasks in the original problem domain by solving a compact discrete MDP for new policies. This is one of the critical motivations for using bisimulations: optimal policies in the abstract MDP induce optimal policies in the original MDP (Givan et al., 2003). We define the abstract MDP $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R}, \gamma \rangle$ with the discrete transition table learned by the HMM prior. The abstract reward function can encode any reward function in the ground MDP by projecting ground rewards into the abstract space using the encoder. Now, we can use standard discrete value iteration to find new policies. These policies can be immediately applied in the ground MDP: observations of state in the ground MDP can be projected into the discrete abstract MDP and the new policy can be used to calculate an action.

3. From VIB abstraction to bisimulation

The HMM embedded in our model learns parameters of a compact discrete MDP (Subsection 2.2), but it is not a priori clear that this abstraction is also a bisimulation. We show that under idealized conditions, every optimal solution to the objective L_{IB} is a bisimulation. Under assumptions state in Appendix B, the following theorem holds.

Theorem 1
There exist model parameters θ that reach the global maximum of $L_{IB}(\theta) = -\beta \log K$ with $\beta > 0$. The abstraction mapping ϕ induced by any such model parameters is a bisimulation.

4. Shapes World Experiment

We use a modified version of the Pucks World from Biza and Platt (2019). The world is divided into a 4×4 grid and objects of various shapes and sizes can be placed or stacked in each cell. States are represented as simulated 64×64 depth images. The agent can execute a PICK or PLACE action in each of the cells to pick up and place objects. The goal of abstraction is to recognize that the shapes and sizes of objects do not influence the PICK and PLACE actions. We instantiate eight different tasks described in Figure 4.

We test the ability of the learned abstract models to plan for new goals. We are able to reach a goal only if it is represented as a distinct abstract state in our model–such abstract states can only exist if the training dataset contains examples of the goal. Therefore, we can generalize to unseen goals in the sense that our model does not know about these goals during training, but they are represented in the dataset. During planning for a particular goal, we create a new reward function for the abstract model and assign a reward 1 to all transitions in the dataset that reach that goal. Then, we run Value Iteration in the abstract model and use the found state-action values to create a stochastic softmax policy. See Appendix B.3 for more details.

Source tasks	2S	3S	2R	3R	2&2S	3ST	2D	3D
2S	**99.9** ±0.1	-	98.6 ±0.5	-	-	-	**98** ±0.7	-
2S, 2R	99.2 ±0.9	-	**99.9** ±0.1	-	-	-	81.5 ±3.1	-
3S	90 ±2.6	**98.2** ±0.8	75.7 ±2.7	40.8 ±14.7	-	61.9 ±7.5	67 ±3.6	24.9 ±7.7
3ST	74.4 ±3.5	21.8 ±6.4	**98.8** ±0.2	73.9 ±4.4	-	**98.8** ±0.4	83.6 ±2.7	39.3 ±4.2
3S, 3R	93.8 ±2.2	88.8 ±3.3	91.5 ±1.5	**88.4** ±2.7	-	74.8 ±6.1	**86** ±3.4	**65.2** ±6.6
3S, 3ST	**98.1** ±1.2	97.8 ±1.2	98.1 ±1.7	75.2 ±4.2	-	92.2 ±1.8	84.1 ±4.3	51.3 ±6.9
2&2S	76.9 ±8.2	16.2 ±2.5	65.8 ±3.6	24.9 ±4	**46.2** ±7.1	4.7 ±2.4	46.6 ±5.5	12.2 ±2.8
2&2S, 3S	**92.8** ±2.7	**33.9** ±3.7	67.6 ±4.4	30.8 ±3.2	38 ±1.7	9.6 ±2.6	51.5 ±5.1	**16.8** ±3.4
2&2S, 3R	61.8 ±5.4	18.4 ±3.1	71.6 ±3.1	**70.7** ±4.8	33.9 ±7.6	10.3 ±4.2	50.4 ±3	10.1 ±1.1
2&2S, 3ST	71.5 ±7.6	24.4 ±3.6	**75.6** ±4.4	29.6 ±2.1	36.1 ±4.4	**33.7** ±4.5	**53.4** ±4.9	15 ±2.4

Table 1: Transfer experiments in the Shapes World environment. In the same order as the examples in Figure 4, the tasks are stacking two/three objects (2S/3S), two/three objects in a row (2R/3R), two/three objects diagonal (2D/3D), two and two stacks (2&2S) and stairs from three objects (3ST). We train our model with the HMM prior on one or more source tasks and then use the abstract MDP induced by the HMM prior to plan for every task. We report the *success rate of reaching each goal (%)* with a budget of 20 time steps; we calculate means and standard deviations over 10 runs. If a model is trained on tasks involving two objects, we do not test it on tasks with three objects and so on; hence, the blank spaces in the table.

Our model is trained on one or two tasks and we report its ability to plan for every single task (Table 1). For tasks with a moderate number of abstract states (e.g. 2 objects stacking in a 4×4 grid world has 136 abstract states in the coarsest bisimulation), our method can successfully transfer to new tasks of similar complexity without additional training. For instance, the abstract model learned from two pucks stacking can plan for placing two and three pucks in a row with a 90%+ success rate. The middle section of Table 1 shows tasks with coarsest abstractions at the limit of what our abstract model can represent. We can still transfer to similar tasks with a success rate higher than 75%.

The bottom section of Table 1 demonstrates that our algorithm can find partial solutions even if the number of abstract states in the coarsest bisimulation exceeds the capacity of the HMM prior. See Appendix A for additional experiments.

5. Related Work

Several recent neural-net-based methods use discrete representation for planning. Serban et al. (2018) learn a factored transition given a predefined discrete state abstraction. They focus their empirical evaluation on Natural language processing tasks. Kurutach et al. (2018) proposed a Generative adversarial network for learning a forward model with either continuous or discrete latent states. While they show superior performance on a rope manipulation task with continuous latent states, the discrete state representation learning and planning was only evaluated on a toy 2D navigation task. Finally, Corneil et al. (2018) and van der Pol et al. (2020) both learn an abstract MDP with discrete states based on ground states represented as images.

References

David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L. Littman, and Lawson L. S. Wong. State abstraction as compression in apprenticeship learning. In *AAAI*, 2019.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Ondrej Biza and Robert Platt. Online abstraction with mdp homomorphisms for deep learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 1125–1133, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

Dane S. Corneil, Wulfram Gerstner, and Johanni Brea. Efficient modelbased deep reinforcement learning with variational state tabulation. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 1057–1066, 2018.

Thomas L. Dean and Robert Givan. Model minimization in markov decision processes. In *AAAI/IAAI*, 1997.

Thomas L. Dean, Robert Givan, and Sonia M. Leach. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *UAI*, 1997.

Robert Givan, Thomas Dean, and Matthew Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163 – 223, 2003. ISSN 0004-3702. Planning with Uncertainty and Incomplete Information.

David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J. Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 8747–8758, 2018.

Lucas Lehnert and Michael L. Littman. Transfer with model features in reinforcement learning. *CoRR*, abs/1807.01736, 2018.

Lihong Li, Thomas J. Walsh, and Michael L. Littman. Towards a unified theory of state abstraction for mdps. In *In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, Feb 2015.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Iulian Vlad Serban, Chinnadhurai Sankar, Michael Pieper, Joelle Pineau, and Yoshua Bengio. The bottleneck simulator: A model-based deep reinforcement learning approach. *CoRR*, abs/1807.04723, 2018.

Naftali Tishby, Fernando Pereira, and William Bialek. The information bottleneck method. *Proceedings of the 37th Allerton Conference on Communication, Control and Computation*, 49, 07 2001.

Elise van der Pol, Thomas Kipf, Frans A. Oliehoek, and Max Welling. Plannable approximations to MDP homomorphisms: Equivariance under actions. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pages 1431–1439, 2020.

Appendix A. Additional Experiments

A.1. Column World

The purpose of this experiment is to compare our method to a model-based approximate bisimulation baseline in a simple discrete environment. Column World is a grid world with 30 rows and 3 columns (Lehnert and Littman, 2018). The agent can move left, right, top and down, and it receives a reward 1 for any action executed in the right column; otherwise, it gets 0 reward. Hence, the agent only needs to know if it is in the left, middle or right column, as illustrated in Figure 1.

First, we train a deep Q-network on this task and use it to generate a dataset of transitions. As a baseline, we train a neural network model to predict r_t and s_{t+1} given (s_t, a_t). We then find a coarse approximate bisimulation for this model using a greedy algorithm from (Dean et al., 1997) with the approximation constant ϵ set to 0.5. We compare it with our method trained with an HMM prior on $Q_\pi(s_t, a_t)$ predicted by the deep Q-network. We represent each state as a discrete symbol and use fully-connected neural networks for all of our models. See Appendix B.2 for details.

Figure 3 shows the purity and the size of the abstractions found by our method and the baseline as a function of dataset size. We need a ground-truth abstraction to calculate the abstraction purity–in this case, it is the three-state abstraction shown in Figure 1 right. We assign each ground state to an abstract state (Figure 2) and find the most common ground-truth label for each abstract state. The abstraction purity is the weighted average of the fraction of members of an abstract states that share its label. We include a snippet of code that computes purity in Appendix B.1.

Both methods can find an abstraction with high purity. However, approximate bisimulation does not reduce the state space (there are 90 ground states) until the model of the environment is nearly perfect, which requires more than 11000 training examples. Our method always finds an abstraction with six states (the number of abstract states is a hyper-parameter), but our method finds a compact high-purity abstraction much faster than the baseline. Notice that we parameterize our method with more abstract states than the size of the coarsest bisimulation. In practice, this over-parameterization aids convergence.

A.2. Shapes World GMM and HMM comparison

We test the ability of our algorithm to find accurate bisimulation partitions. Table 2 shows the results for our method for both the GMM and the HMM prior. Both models reach a high abstraction purity (described in Section A.1) in all cases except for the three objects stacking task in a 4×4 grid world. The smallest MDP for which a bisimulation exists contains 936 abstract states; our algorithm has 1000 possible abstract states available. Our experiment shows that the HMM prior can leverage the temporal information, which is missing from the GMM, to allocate abstract states better.

Setting	GMM	HMM
2 pucks, 2×2 grid	100 ±0	97 ±1
2 pucks, 3×3 grid	100 ±0	98 ±1
2 pucks, 4×4 grid	99 ±1	97 ±0
3 pucks, 2×2 grid	99 ±1	97 ±1
3 pucks, 3×3 grid	99 ±1	96 ±1
3 pucks, 4×4 grid	56 ±15	89 ±1
2 objects, 2×2 grid	100 ±0	97 ±1
2 objects, 3×3 grid	100 ±0	97 ±0
2 objects, 4×4 grid	100 ±0	97 ±0
3 objects, 2×2 grid	100 ±0	97 ±1
3 objects, 3×3 grid	98 ±1	96 ±1
3 objects, 4×4 grid	67 ±3	91 ±1

Table 2: Results for learning abstractions for stacking objects. The top section involves manipulating objects of only one type (pucks), whereas the bottom section involves four object types (puck, box, square and plus. GMM and HMM refer to the two types of priors our model uses (Subsection 2.2) and we report *abstraction purities (%)*. We report means and standard deviations over 10 runs.

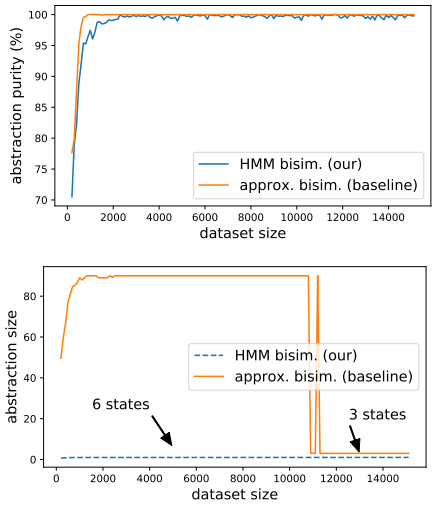

Figure 3: Comparison between model-based approximate bisimulation and our method in Column World (Lehnert and Littman, 2018). We vary the dataset sizes used for learning the bisimulation from 1000 to 20000 samples. Abstraction size refers to the number of abstract states.

Appendix B. Theoretical Analysis

We analyze the idealized case where the following assumptions hold:

1. The transitions in the ground MDP are deterministic;
2. The HMM prior has enough components to represent a bisimulation, no two components share a mean;
3. The prior over hidden states $p(c_t = k)$ is fixed;
4. The encoder is deterministic and the prior observation model is an identity function over component means;
5. The decoder $p(y|z_t, a)$ makes a prediction for each component using a table of state-action values.

Assumptions on the MDP

Let $M = \langle S, A, T, R, \gamma \rangle$ be a ground MDP with finite S and A, an arbitrary R and a deterministic T. We assume there exists a bisimulation mapping $\phi_{bisim} : S \to C$ with K abstract states ($|C| = K$). ϕ_{bisim} does not have to be the coarsest bisimulation (i.e. the one

using the lowest possible number of abstract states).

Assumptions on the model
Let $f : S \to \mathbb{R}^D$ be a deterministic encoder. For simplicity, we use a Hidden Markov Model with a Kronecker delta observation model parameterized by K component means $C = \{c_1, c_2, ..., c_K\}$ with probability density function

$$p(z|c = k) = \mathbb{I}\left[\mu_k = z\right]. \tag{7}$$

The encoder and the Hidden Markov Model induce an abstraction mapping $\phi : S \to C$

$$\phi(s) = \arg\max_{c \in C} p(c|f(s)). \tag{8}$$

We assume $\forall_{c,c' \in C} \mu_c \neq \mu_{c'}$ (i.e. no two components share their means) and $\forall_{s \in S} \exists_{c \in C} f(s) = \mu_c$ (i.e. each encoding equals to some cluster mean). The component prior $p(c)$ assign a uniform probability to each cluster to match our experimental setting.

Finally, we assume the decoder $p(y|z, a)$ is tied to the cluster assignment ϕ. That is, each components has a parameter $q_c \in \mathbb{R}^{|A|}$, which stores the state-action value of this component for each action. The decoder probability density is a normal distribution with a mean of $q_{c,a}$ given an action a and a fixed standard deviation σ

$$p(y|z, a) = \mathcal{N}(y|q_{\phi(z),a}, \sigma^2). \tag{9}$$

We can decompose the objective into a Q-value loss L_Q and a prior loss L_P

$$L_Q(\theta) = \frac{1}{\sigma|S||A|} \sum_{s \in S, a \in A} \Big(Q^*(s, a) - q_{\phi(f(s)),a}\Big)^2, \tag{10}$$

$$L_P(\theta) = -\frac{1}{|S||A|} \sum_{s_t, s_{t+1} \in S, a \in A} T(s_t, a, s_{t+1}) \log \Big(\sum_{c_t, c_{t+1} \in C} p(f(s_t)|c_t) p(f(s_{t+1})|c_{t+1}) p(c_t) p(c_{t+1}|c_t, a) \Big), \tag{11}$$

$$L_{IB}(\theta) = -L_Q(\theta) - \beta L_P(\theta). \tag{12}$$

Definition 2 *An abstraction mapping ϕ is homogeneous if for each $s, s' \in S, a \in A, \hat{c} \in C$ $\phi(s) = \phi(s')$ implies $\sum_{\hat{s} \in \phi^{-1}(\hat{c})} T(s, a, \hat{s}) = \sum_{\hat{s} \in \phi^{-1}(\hat{c})} T(s', a, \hat{s})$.*

Lemma 3 *Let ϕ be an abstraction mapping that is a Q^*-irrelevance abstraction and is also homogeneous. Then ϕ is a bisimulation.*

Proof Fix an abstract state c and an action a. For each $s, s' \in \phi^{-1}(c)$ we have that $Q^*(s, a) = Q^*(s', a)$ (by Q^*-irrelevance) and $\forall \hat{c} \in C$ $\sum_{\hat{s} \in \phi^{-1}(\hat{c})} T(s, a, \hat{s}) = \sum_{\hat{s} \in \phi^{-1}(\hat{c})} T(s', a, \hat{s})$ (by ϕ

being homogeneous). Let us expand the state-action values of (s, a) using the Bellman equation

$$Q^*(s,a) = R(s,a) + \gamma \sum_{\hat{s} \in S} T(s,a,\hat{s}) V^*(s) \tag{13}$$

$$= R(s,a) + \gamma \sum_{\hat{c} \in C} T(s,a,\hat{c}) V^*(\hat{c}) \tag{14}$$

$$= R(s,a) + \gamma x. \tag{15}$$

We can perform a similar expansion for (s', a)

$$Q^*(s',a) = R(s',a) + \gamma \sum_{\hat{c} \in C} T(s',a,\hat{c}) V^*(\hat{c}) \tag{16}$$

$$= R(s',a) + \gamma \sum_{\hat{c} \in C} T(s,a,\hat{c}) V^*(\hat{c}) \tag{17}$$

$$= R(s',a) + \gamma x \tag{18}$$

We write $T(s, a, \hat{c})$ as a shorthand for $\sum_{\hat{s} \in \phi^{-1}(\hat{c})} T(s, a, \hat{s})$. $T(s', a, \hat{c}) = T(s, a, \hat{c})$ holds for all abstract states if $\phi(s) = \phi(s')$ because ϕ is homogeneous. All states mapped to a particular abstract state have the same value (for an optimal policy) due to Q^*-irrelevance; hence, we can write $V^*(\hat{c})$.

Since $Q^*(s, a) = Q^*(s', a)$ and $x = x$ it follows that $R(s, a) = R(s', a)$ (i.e. ϕ is reward-respecting). A reward-respecting homogeneous state abstraction is a bisimulation by definition.

■

Lemma 4 *There exist model parameters θ such that $L_Q(\theta) = 0$.*

Proof *Strategy: we can achieve zero L_Q by assigning states into components such that states in each component have equal state-action values for all actions. We need to show that there are enough components to perform this assignment.*

Parameters θ induce an abstraction mapping ϕ. Assume that $L_Q(\theta) = 0$. Then for each $c \in C$, $s, s' \in \phi^{-1}(c), a \in A$ we have $(Q^*(s, a) - Q^*(s', a))^2 = 0$, which implies $|Q^*(s, a) - Q^*(s', a)| = 0$. Hence, ϕ is a Q^*-irrelevance abstraction (Li et al., 2006). By Li et al. (2006), for each bisimulation abstraction there exists a Q^*-irrelevance abstraction that is equal in size or coarser (i.e. it uses fewer abstract states). By our assumption that we have enough components to represent a bisimulation abstraction, there are also enough components to represent a Q^*-irrelevance abstraction. ■

Lemma 5 *There exist model parameters θ such that $L_P(\theta) = \log K$. $L_P(\theta) = \log K$ is the global minimum.*

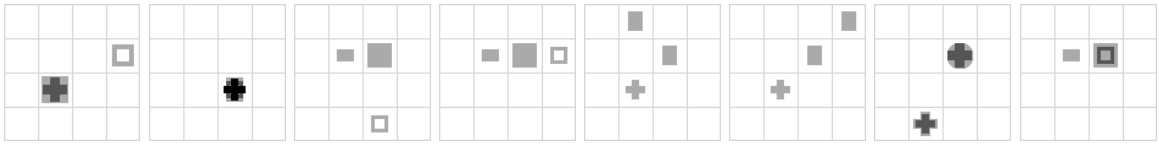

Figure 4: Goal states for tasks in Shapes World. There are four types of objects–pucks, boxes, squares and pluses–placed in a grid world. From left to right we have examples of goal states for two objects stacking, three objects stacking, two objects in a row, three objects in a row, two objects diagonal, three objects diagonal, two and two objects stacking, stairs from three objects.

Proof Under our assumptions, we can reduce the prior loss function to

$$L_P(\theta) = -\frac{1}{|S||A|} \sum_{s_t \in S, a \in A, s_{t+1} \in S} T(s_t, a, s_{t+1}) \log \frac{p(\phi(f(s_{t+1}))|\phi(f(s_t)), a)}{K}. \tag{19}$$

The minimum of this loss function is achieved when the term $p(\phi(f(s_{t+1}))|\phi(f(s_t)), a) = 1$ for all states and actions. Since we assume the transition dynamics of the ground MDP are deterministic, this transition model is only possible if the abstraction mapping ϕ is homogeneous. Our model can represent such abstraction mapping because, by our assumption, it can represent a bisimulation, which is homogeneous.

■

Theorem 6 *There exist model parameters θ that reach the global maximum of $L_{IB}(\theta) = -\beta \log K$ with $\beta > 0$. The abstraction mapping ϕ induced by any such model parameters is a bisimulation.*

Proof Since we assume we have enough components to represent a bisimulation, we can represent a Q^*-irrelevance abstraction (by Lemma 2) that is also homogeneous (by Lemma 3). Any such abstraction is a bisimulation by Lemma 1. ■

Appendix C. Experimental Details

C.1. Columns World

The deep Q-network that is used to collect the dataset has two hidden layers of 256 neurons followed by ReLU activation functions. We train it for 40000 time steps with an ϵ-greedy policy; ϵ linearly decays from 1 to 0.1 over 20000 time steps. We use a learning rate of 0.0001, 32 mini-batch size, the target network is updated every 100 time steps and we use prioritized replay with the default settings (Schaul et al., 2016). The optimizer used for training is mini-batch gradient descent with momentum set to 0.9. The dataset for training the abstract and direct models is collected after training with ϵ set to 0.5. We compute the abstraction purity over every possible ground state.

Each state is represented as a 90-dimensional one-hot encoded vector. As a baseline, we train a model with two fully-connected layers of 128 neurons followed by ReLU activations and two heads, one for predicting the reward and the other for the next state. We use a mean squared error loss for the reward prediction and cross-entropy loss for the next state (we treat each dimension of the predicted 90-dimensional vector as a probability of being in that particular state). Finally, we run an approximate partition iteration algorithm following Dean et al. (1997).

Our model with an HMM prior uses the same architecture as the above model, except it makes only one prediction: the state-action value associated with a given state-action pair. We set the number of hidden states to 6, the observation model of the HMM is 32-dimensional, encoder and model learning rates are 0.01, β is 0.0001, the means of the HMM observations are initialized with 0 mean and 0.01 standard deviation and the diagonal covariances are initialized with -1 mean and 0.1 standard deviation before being exponentiated. We train the models using the Adam optimizer (Kingma and Ba, 2015).

C.2. Shapes World

For dataset collection, the input image is resized to 64×64 before being fed into a deep Q-network. We use four convolutional layers with 32, 64, 128, and 256 filters; the filter size is four and the stride is set to two (each convolutional downsamples the input by a factor of two); we use "same" padding. The convolutions are followed by a single fully-connected layer with 512 neurons and a head for predicting the state-action values. The learning rate is set to 0.00005, the batch size is 32, the buffer size is 100000 and we train for 100000 steps. Actions are selected with an ϵ-greedy policy–ϵ is linearly decayed from 1.0 to 0.1 over 50000 time steps. We collect a dataset of 100000 transitions after training the model with ϵ set to 0.1. 80% of the dataset is used for training and 20% for computing the abstraction purity.

Our model uses the same neural network, except we insert batch normalization between each layer and its activation function (we use ReLU) (Ioffe and Szegedy, 2015). The model predicts a 32-dimensional vector of means and a diagonal covariance, from which we sample the continuous encoding z. The GMM or HMM uses 1000 components (hidden states), the initial means of the components are drawn from a Gaussian distribution with 0 mean and 0.1 variance. The variances are drawn from a Gaussian with -1.0 mean and 0.1 before being exponentiated. We train the model for 50000 steps, then we collect batch normalization statistics over the whole dataset, and we resume training only the prior with a fixed encoder and unfrozen component weights $p(c_t)$ (previously held uniform fixed) for another 50000 steps. β is set to 10^{-6}, encoder learning rate to 10^{-3} and prior learning rate to 10^{-2}. We train the model with Adam optimizer (Kingma and Ba, 2015).

To get a reward function over the abstract MDP induced by the HMM, we find abstract states with 99% of ground states that are mapped to them being goal states for a given goal. We plan state-action values for each abstract-state action pair using Value Iteration and run an agent with a softmax policy with τ set to 10^{-2} for 100 episodes.

We ran a hyper-parameter search for the learning rates and β on the task of stacking three pucks and then used the same parameters for all other experiments. We tried the following learning rates: $\{0.005, 0.001, 0.0005, 0.0001\}$ and β: $\{0.005, 0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$.