Learning discrete state abstractions with deep variational inference

Ondrej Biza [1] Robert Platt [*][1] Jan-Willem van de Meent [*][1] Lawson L.S. Wong [*][1]

Abstract

Abstraction is crucial for effective sequential decision making in domains with large state spaces. In this work, we propose a variational information bottleneck method for learning approximate bisimulations, a type of state abstraction. We use a deep neural net encoder to map states onto continuous embeddings. The continuous latent space is then compressed into a discrete representation using an action-conditioned hidden Markov model, which is trained end-to-end with the neural network. Our method is suited for environments with high-dimensional states and learns from a stream of experience collected by an agent acting in a Markov decision process. Through a learned discrete abstract model, we can efficiently plan for unseen goals in a multi-goal Reinforcement Learning setting. We test our method in simplified robotic manipulation domains with image states. We also compare it against previous model-based approaches to finding bisimulations in discrete grid-world-like environments.

1. Introduction

High-dimensional state spaces are commonly seen in recent reinforcement learning applications (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 the necessary information and discard the rest. One principled approach for state abstraction is via *bisimulation* in Markov decision processes (MDP) (Dean & Givan, 1997). Bisimulations formalize the notion of finding smaller equivalent *abstract* MDPs that preserve transition and reward information, i.e., they retain the 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.

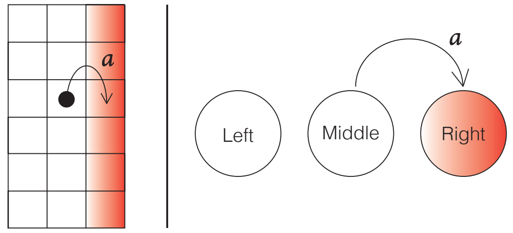

Figure 1. Example of bisimulation abstraction. The Column World (left) has 3 columns and 30 rows (we only show 6 rows); the agent travels between cells like a chess king (Lehnert & Littman, 2018). Since the agent gets a reward 1 for being in the right row (red) and 0 otherwise, it is irrelevant in which row it is located. Hence, the environment can be simulated by an MDP with three states (right), which is a bisimulation of the original problem.

Unfortunately, finding bisimulations with maximally compressed state spaces is NP-hard (Dean et al., 1997). One common approach to circumvent finding an abstract MDP is using bisimulation metrics, which facilitate transfer of *existing* policies to similar states (Ferns et al., 2004). However, this method cannot generalize to *new* tasks, for which there is no existing policy. Instead, we pursue the original bisimulation goal of finding a *discrete abstract MDP*, that can be used to solve unseen tasks efficiently.

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). This framework is typically used to learn representations that predict quantities of interest accurately while ignoring certain aspects of the domain. The VIB approach has even been recently explored in the context of state abstraction, but the state abstraction found in general does not result in 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 on new tasks. 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, which we then infer belongs to a learned discrete abstract state $\bar{s}$. One perspec-

[*]Equal contribution [1]Khoury College of Computer Sciences, Northeastern University, Boston, MA, USA. Correspondence to: Ondrej Biza <biza.o@husky.neu.edu>.

tive on the VIB objective is that it is learning an encoder (state abstraction function: $s \mapsto z$) that predicts observed Q-values well using z alone, but is subject to a prior in 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.

Although structured priors have been used in the context of variational autoencoders, one key difference in our approach is that 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.

In summary, our contributions are:

- Framing bisimulation learning as a VIB objective.
- Introducing two structured priors (GMM, HMM) with learned parameters for VIB-based state abstraction.
- Using the learned parameters of the prior to extract a discrete abstract MDP, which is an approximate bisimulation of the original MDP.
- Using the abstract MDP to plan for new goals.

2. Background

Markov decision process: We model our tasks as episodic Markov Decision Processes (MDPs). An MDP is a tuple $M = \langle S, A, T, R, \gamma \rangle$ (Bellman, 1957), where S and A are state and action sets, respectively. The function $R : S \times A \to \mathbb{R}$ describes the expected reward associated with each state-action pair. The density $T(s, a, s') = p(s'|s, a)$ describes transition probabilities between states. $\gamma \in \mathbb{R}$ is a discount factor. A policy $\pi : S \times A \to [0, 1]$ encodes the behavior of an agent as a probability distribution over A conditioned on S. The state-action value Q_π of a policy π is the expected discounted reward of executing action a from state s and subsequently following policy π:

$$Q_\pi(s, a) := R(s, a) + \gamma \mathbb{E}_{s' \sim T, a' \sim \pi} \left[Q_\pi(s', a') \right] \quad (1)$$

We want to behave optimally both in the ground and abstract MDPs. A policy π^* is optimal when $Q_{\pi^*}(s, a) \geq Q_\pi(s, a),\ \forall s, a \in S \times A$.

State abstraction: We approach state abstraction from the perspective of model minimization. The goal is to find a function that maps from the state space S of the original MDP to a compact state space $\bar{S}$ while preserving the reward and transition dynamics (Dean & Givan, 1997; Givan et al., 2003). Concretely, we want a surjective function $\phi : S \to \bar{S}$ that induces a partition over S. That is, each *abstract state* $\bar{s} \in \bar{S}$ is associated with a block of states in S defined by the preimage of ϕ at $\bar{s}$, $\phi^{-1}(\bar{s}) \subseteq S$. Since ϕ must induce a partition over S, we require $\phi^{-1}(\bar{s}_1) \cap \phi^{-1}(\bar{s}_2) = \emptyset,\ \forall \bar{s}_1, \bar{s}_2 \in \bar{S}$. A *bisimulation* is a surjection $\phi : S \to \bar{S}$ that induces a partition over S and preserves the reward and transition dynamics. It is commonly formalized as:

Definition 1 (MDP Bisimulation)**.** Let $M = \langle S, A, T, R, \gamma \rangle$ and $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R}, \gamma \rangle$ be MDPs. A function $\phi : S \to \bar{S}$ is an MDP bisimulation from M to $\bar{M}$ if the preimage ϕ^{-1} induces a partition of S and for each $s_1, s_2 \in S$, $\bar{s} \in \bar{S}$ and $a \in A$, $\phi(s_1) = \phi(s_2)$ implies both $R(s_1, a) = R(s_2, a)$ and $\sum_{s' \in \phi^{-1}(\bar{s})} p(s_1, a, s') = \sum_{s' \in \phi^{-1}(\bar{s})} p(s_2, a, s')$.

Given two MDPs M and $\bar{M}$, there may exist many bisimulations. These bisimulations can be placed in a partial order. Given two bisimulations ϕ and ϕ' from M to $\bar{M}$, we will say that ϕ' is a *refinement* of ϕ if the partition induced by ϕ' is a refinement of that induced by ϕ.

Information Bottleneck Methods: We approach the state abstraction problem using information bottleneck (IB) methods (Tishby et al., 2001). These methods assume that we provide a distribution $q(s, y)$ over features s (in our case the state) and a prediction target y (in our case expected reward or return). Given $q(s, y)$, IB methods train an encoder $q(z \mid s)$ that maps s onto a compressed representation z by maximizing the IB objective

$$R_{IB} = I(y; z) - \beta\, I(s; z), \quad (2)$$

where I denotes the mutual information between its arguments. The intuition behind this objective is that we would like to learn a (lossy) compressed representation of s by maximizing the correlation between z and the target y, which ensures that z is predictive of y, whilst minimizing the correlation between z and s, which ensures that any information in s that does not correlate with y is discarded.

In practice, evaluating the IB objective is intractable. Instead of optimizing Equation (2) directly, IB methods introduce two variational distributions $p(y \mid z)$ and $p(y)$ to bound the mutual information terms

$$I(y; z) \geq \mathbb{E}_{q(s,y,z)} \left[\log p(y|z) - \log q(y) \right], \quad (3)$$

$$I(s; z) \leq \mathbb{E}_{q(s,y,z)} \left[\log q(z \mid s) - \log p(z) \right]. \quad (4)$$

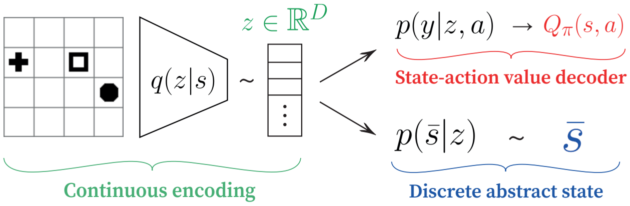
$q(z|s)$
$z \in \mathbb{R}^D$
$p(y|z,a) \rightarrow Q_\pi(s,a)$
State-action value decoder
$p(\bar{s}|z) \sim \bar{S}$
Continuous encoding
Discrete abstract state

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

Combining these terms then bounds the IB objective

$$R_{IB} \geq \mathbb{E}_{q(s,y,z)}\left[\log p(y \mid z) + \beta \log \frac{p(z)}{q(z \mid s)}\right] + H(y),$$

Maximizing the above with respect to $p(y \mid z)$, $p(z)$, and $q(z \mid s)$ is a form of variational expectation maximization; optimizing $p(y \mid z)$ and $p(z)$ tightens the bound, whereas optimizing $q(z \mid s)$ maximizes the IB objective. Note that the entropy term $H(y)$ does not depend on $q(z \mid s)$ and can therefore be ignored safely during optimization.

In practice, modern IB methods use a neural network as the encoder (Alemi et al., 2017), and perform stochastic gradient descent using reparameterized Monte Carlo estimators, resulting in models that are closely related to variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014). We can make the connection between IB methods and variational autoencoders concrete by noting that when $\beta = 1$, the lower bound on R_{IB} becomes

$$\mathcal{L} = \mathbb{E}_{q(s,y)}[\log p(y) - \mathrm{KL}(q(z \mid s) \,||\, p(z \mid y))]. \qquad (5)$$

Maximizing $\mathcal{L}$ is then equivalent to learning an encoder $q(z \mid s)$ that approximates the posterior $p(z \mid y)$ of a generative model $p(y, z)$, and optimizing the generative model to maximize the log marginal-likelihood $\log p(y)$.

3. Related Work

Bisimulation for MDPs was first described by Dean & Givan (1997) together with an algorithm for finding it when a full model of the ground MDP is available. Without the exact model of an environment, comparing transition and reward functions is problematic, as even a fractional deviation between the dynamics of two similar states will result in them being separated. Dean et al. (1997) proposed ϵ-approximate bisimulation: it allows the dynamics of states mapped to a single abstract state to vary up to a constant ϵ. However, the problem of finding the coarsest ϵ-approximate grouping is NP-hard. Ferns et al. (2004; 2006) extended approximate bisimulation to bisimulation metric: a method for comparing states based on their one-step dynamics. The bisimulation metric has been used to transfer policies from simple to complex domains Castro & Precup (2010; 2011), and Abel et al. (2016) proved bounds for the quality of a policy over the approximate abstraction. More recently, Castro (2019) created an objective for a deep neural network based on the bisimulation metric to learn similarities between states in grid worlds and Atari games. The main difference between this line of work and ours is that we aim to learn an abstract MDP with discrete states, in which we can plan efficiently, whereas bisimulation metrics are more commonly used to find similar states for the purpose of transfer of policies.

The Information Bottleneck method defines an objective that maximizes the predictive power of a model while minimizing the number of bits needed to encode an input (Tishby et al., 2001). Abel et al. (2019) drew a connection between Information Bottleneck and Rate-distortion theory for the purpose of learning state abstraction. They propose an Expectation-Maximization-like algorithm for finding state abstractions in discrete-state environments and a loss function for learning compressed policies in Atari games. Apart from state abstraction, the Information Bottleneck method has also been used to regularize policies represented by deep neural networks. Goyal et al. (2019) improved the generalization of goal-conditioned policies by training their agent to detect decision states – states in which the agent requires information about the current goal to act optimally. Here, the Information Bottleneck objective forces the agent to only access the information about the current goal when it is necessary. Teh et al. (2017) used a similar objective to distill a task-agnostic policy in the multitask Reinforcement Learning setting. Strouse et al. (2018) used information bottleneck to control the amount of information transferred between two agents. Tishby & Polani (2011); Rubin et al. (2012) study the interactions of an agent with and environment from an information-theoretic perspective.

4. Learning bisimulations

We propose a variational model 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 net encoder $q(z_t|s_t)$ that projects states (usually represented as images) onto a low-dimensional continuous latent space (Figure 2 left),
2. a prior $p(z_t, z_{t+1}|a)$ that encodes the prior belief that the experience was generated by a small discrete Markov process (Figure 2 lower right),
3. a linear decoder $p(y|z_t)$ 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). Unlike the standard setting, we encode pairs of ground states (s_t, s_{t+1}) as latent state pairs (z_t, z_{t+1}). This enables us to learn a tabular transition function between discrete states inside a prior $p(z_t, z_{t+1}|a)$. We treat the discrete states of the prior as abstract states, which together with the learned transition function and a given reward function define an abstract MDP.

4.1. Bisimulation as an information bottleneck

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:

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

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: $q(z_t, z_{t+1}|s_t, s_{t+1}, a) = q(z_t|s_t)q(z_{t+1}|s_{t+1})$.

Putting it together, we maximize an IB lower bound

$$\begin{aligned} L_{IB} = \mathbb{E}_{q(s,a,y,z)}\Big[&\log p(y|z_t, a) \\ &- \beta \log \frac{q(z_t|s_t)q(z_{t+1}|s_{t+1})}{p(z_t, z_{t+1}|a)}\Big]. \end{aligned} \quad (6)$$

L_{IB} presents a trade-off between encoding enough information of s_t in order to predict y (the first term of Equation 6) 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 y (what we predict in the first term of Equation 6) is a state-action value, not a reward. This gives our model additional supervision. Without it, our model tends to collapse by predicting a single abstract state for each ground state.

4.2. Structured Priors

The denominator of the second term in Equation 6 $p(z_t, z_{t+1}|a)$ is the prior. We want it to express an expectation that we are observing a discrete Markov process. We explore two approaches: a prior based on a Gaussian mixture model (GMM) and a prior based on an action conditioned Hidden Markov model (HMM).

4.2.1. GMM Prior

A Gaussian mixure model consists of K components, each parameterized by a mean μ_k and a covariance Σ_k:

$$\begin{aligned} p_{\text{GMM}}(z_t) &= \sum_{k=1}^{K} p(z_t|c_t = k)p(c_t = k) &(7) \\ &= \sum_{k=1}^{K} \mathcal{N}(z_t|\mu_k, \Sigma_k)\sigma_k, &(8) \end{aligned}$$

where $\sigma_k = p(c_t = k)$ 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 k^{th} 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; σ_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 σ_k is the probability that the continuous encoding z_t was generated by the k^{th} abstract state. This encodes the prior belief that latent encoding of state should be distributed according to a mixture of Gaussians with unknown mean, covariance, and weights. Note that while this approach gives us an encoder that projects real-valued high dimensional states onto a small discrete set of abstract states, it ignores the temporal aspect of the Markov process.

4.2.2. HMM Prior

In order 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

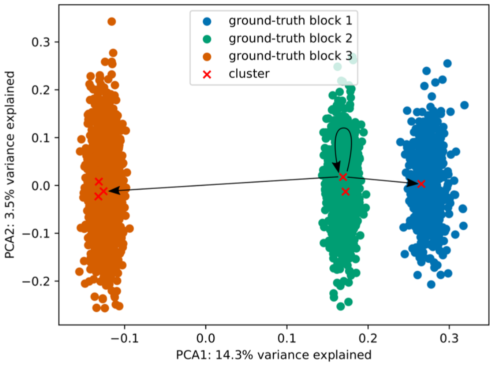

Figure 3. Latent encoding of states in the Column World environment found using our model (PCA projection). The colors represent the three blocks in the coarsest bisimulation (see Figure 1 left), which are unknown during training. The red crosses denote the locations of the component means and the arrows denote the transition function found by our model using the HMM prior.

the probability of transitioning from an abstract state k to an abstract state l under an action a. Using this model, the prior becomes:

$$p_{\text{HMM}}(z_t, z_{t+1}|a) = \sum_{k=1}^{K}\sum_{l=1}^{K} p(z_t|c_t = k) \tag{9}$$

$$p(z_{t+1}|c_{t+1} = l)p(c_{t+1} = l|c_t = k, a)p(c_t = k) \tag{10}$$

$$= \sum_{k=1}^{K}\sum_{l=1}^{K} \mathcal{N}(z_t|\mu_k, \Sigma_k)\mathcal{N}(z_{t+1}|\mu_l, \Sigma_l)T^a_{k,l}\sigma_k \tag{11}$$

As with the GMM model, we allow the parameters of this model (μ_k, Σ_k, and T^a) to vary, except for σ_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 6 is evaluated.

Figure 3 illustrates the latent embedding found using the HMM model for the Column World domain shown in Figure 1. The three clusters correspond to the three states in the coarsest abstract bisimulation MDP. These three clusters are overrepresented by six cluster centroids (the red x's) because we ran our algorithm using six cluster components. The algorithm "shared" these six mixture components among the three bisimulation classes. The result is still a bisimulation – just not the coarsest bisimulation.

4.3. Deep encoder and end-to-end training

The loss L_{IB} (Equation 6) 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 neural network that predicts the mean $\mu^{CNN}(s_t)$ and the diagonal covariance $\Sigma^{CNN}(s_t)$ of a multivariate normal distribution. In our experiments, we used a modified version of the encoder architecture from (Ha & 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 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.

4.4. 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 solutions in the abstract MDP induce optimal solutions in the original MDP. Using the discrete transition table found using the HMM prior, we define the abstract MDP $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R}, \gamma \rangle$. The abstract reward function can be defined to encode any reward function in the ground MDP by projecting ground rewards into the abstact 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.

5. Connecting VIB abstraction to bisimulation

The HMM embedded in our model learns parameters of an compact discrete MDP (Subsection 4.2). But, is it a bisimulation? We show that under idealized conditions, every optimal solution to the objective L_{IB} is, in fact, a bisimulation. We analyze the idealized case where the following assumptions hold:

1. The agent receives a reward of 1 in goal states and zero elsewhere;
2. The transition function of the ground MDP is deterministic;

[1] Appendix A.1. in the version 4 of the arXiv submission.

3. The HMM prior is parameterized no fewer states than exist in the ground MDP, no two states share their means;
4. The prior over hidden states $p(c_t = k)$ is held fixed;
5. The covariance of our encoder and prior components is 0;
6. The linear decoder $p(y|z_t, a)$ is replaced with a 1-nearest-neighbor regressor: in order to predict y given z_t, it finds the closest encoding z' of state s' to z_t (Euclidean distance) and predicts the state-action value $Q(s', a)$ for an action a.

Theorem 1 (HMM-bisimulation theorem)**.** Given:

1. a ground MDP $M = \langle S, A, T, R, \gamma \rangle$,
2. the state-action value function Q^* of an optimal policy,
3. a set of optimal parameters of our model $\theta^* \in \arg\max L_{IB}$, $\beta \in (0, 1)$, and
4. an abstract $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R}, \gamma \rangle$ induced by the HMM prior

and adhering to the idealized assumptions described above, there exists a bisimulation mapping from M to $\bar{M}$.

See Section A in the Appendix for the proof.

6. Experiments

The aim of our experiments is to investigate the following aspects of our method:

- its ability to find abstractions that are compact and accurately model the ground MDP,
- planning in the abstract MDP for new goals, and
- its performance in environments without a clear notion of an abstract state.

We start with a simple grid world experiment to compare our method against an approximate bisimulation baseline Subsection 6.1. Then we test it in more complex domains with image states in Subsections 6.2 (and Appendix C.1). Finally, we report results for simplified Atari games that break the assumptions of our method in Subsection 6.3.

6.1. 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 & 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.

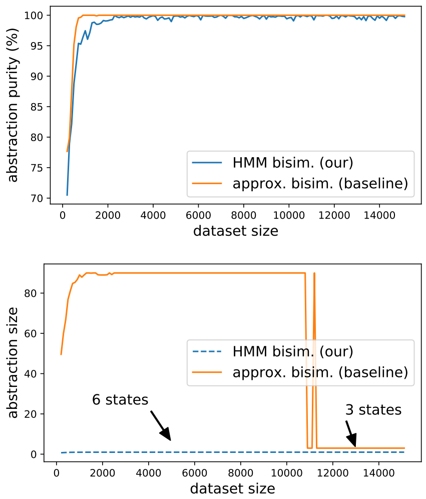

Figure 4. Comparison between model-based approximate bisimulation and our method in Column World (Lehnert & Littman, 2018). We vary the dataset sizes used for learning the bisimulation from 1000 to 20000 samples. Abstraction size refers to the number of abstract states.

Figure 4 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 this measure in Appendix B.1.

Both of the methods can find an abstraction with a 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. This happens only when the dataset has more than 11000 examples. Our method always finds an abstraction with six states (the number of abstract states is a hyper-parameter), but notice that our method finds a compact high purity abstraction much sooner than does the baseline method. Notice that we parameterize our method with more abstract states than the size of the coarsest bisimulation. In practice, this over-parameterization helps our method converge.

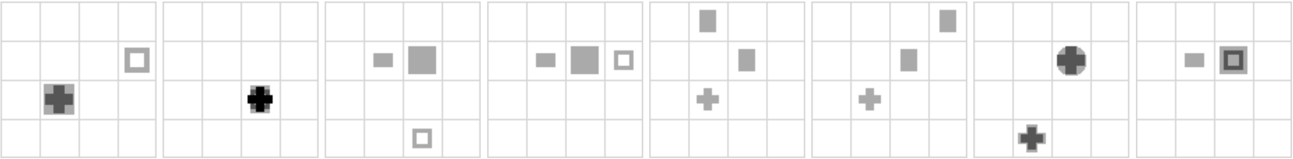

Figure 5. Goal states for the eight tasks in Shapes World. From left to right: two objects stacking, three objects stacking, two objects in a row, three objects in a row, two objects diagonal, three objects diagonal, two and two objects stacking, stairs from three objects.

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

Table 1. Results for learning abstractions for puck stacking (top) and stacking of objects of various shapes (bottom). GMM and HMM refer to the two types of priors our model uses (Subsection 4.2) and we report abstraction purities. Each model was trained 10 times on the same dataset, we report the means and standard deviations.

6.2. Shapes World

We use a modified version of the Pucks World from Biza & 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 here is to recognize that the shapes and sizes of objects do not have any influence on the PICK and PLACE actions. We instantiate eight different tasks in this domain described in Figure 5.

First, we test the ability of our algorithm to find accurate bisimulation partitions. Table 1 shows the results for our method for both the GMM and the HMM prior. Both of the models reach a high abstraction purity (described in Section 6.1) 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.

Next, 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.

Our model is trained on one or two tasks and we report its ability to plan for every single task (Table 2). For tasks with abstractions that are simple to represent (e.g. 2 objects stacking in a 4×4 grid world has 136 abstract states in the coarsest bisimulation, one for each possible configuration of objects ignoring their shape), 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 2 shows tasks whose coarsest abstractions barely fit into our abstract model. We can still transfer to similar tasks with a success rate higher than 75%.

The bottom section of Table 2 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. This is in stark contrast to methods related to Partition Iteration (e.g. the baseline in Figure 1 and Biza & Platt (2019)) that either find the coarsest bisimulation or start creating new abstract states until the abstraction has the same size as the original problem or some threshold is reached. Even approximate bisimulation, which should tolerate inaccurate models, has the unfortunate property that once it makes an erroneous split of a state block, errors in the following step of Partition Improvement become much more likely, often resulting in a useless abstraction.

We present additional transfer experiments with a house building task similar to Shapes World in Appendix B.4.

Source tasks	2S	3S	2R	3R	2&2S	ST	2D	3D
2S	$99.9 \pm 0.1\%$	$1.2 \pm 0.5\%$	$98.6 \pm 0.5\%$	$94 \pm 2.5\%$	0%	$97.9 \pm 1.2\%$	$98 \pm 0.7\%$	$93.1 \pm 2.3\%$
2S, 2R	$99.2 \pm 0.9\%$	$3.7 \pm 0.5\%$	$99.9 \pm 0.1\%$	$47.3 \pm 4.4\%$	0%	$79.8 \pm 8.9\%$	$81.5 \pm 3.1\%$	$37.9 \pm 6.5\%$
3S	$90 \pm 2.6\%$	$98.2 \pm 0.8\%$	$75.7 \pm 2.7\%$	$40.8 \pm 14.7\%$	0%	$61.9 \pm 7.5\%$	$67 \pm 3.6\%$	$24.9 \pm 7.7\%$
ST	$74.4 \pm 3.5\%$	$21.8 \pm 6.4\%$	$98.8 \pm 0.2\%$	$73.9 \pm 4.4\%$	0%	$98.8 \pm 0.4\%$	$83.6 \pm 2.7\%$	$39.3 \pm 4.2\%$
3S, 3R	$93.8 \pm 2.2\%$	$88.8 \pm 3.3\%$	$91.5 \pm 1.5\%$	$88.4 \pm 2.7\%$	0%	$74.8 \pm 6.1\%$	$86 \pm 3.4\%$	$65.2 \pm 6.6\%$
3S, ST	$98.1 \pm 1.2\%$	$97.8 \pm 1.2\%$	$98.1 \pm 1.7\%$	$75.2 \pm 4.2\%$	0%	$92.2 \pm 1.8\%$	$84.1 \pm 4.3\%$	$51.3 \pm 6.9\%$
2&2S	$76.9 \pm 8.2\%$	$16.2 \pm 2.5\%$	$65.8 \pm 3.6\%$	$24.9 \pm 4\%$	$46.2 \pm 7.1\%$	$4.7 \pm 2.4\%$	$46.6 \pm 5.5\%$	$12.2 \pm 2.8\%$
2&2S, 3S	$92.8 \pm 2.7\%$	$33.9 \pm 3.7\%$	$67.6 \pm 4.4\%$	$30.8 \pm 3.2\%$	$38 \pm 1.7\%$	$9.6 \pm 2.6\%$	$51.5 \pm 5.1\%$	$16.8 \pm 3.4\%$
2&2S, 3R	$61.8 \pm 5.4\%$	$18.4 \pm 3.1\%$	$71.6 \pm 3.1\%$	$70.7 \pm 4.8\%$	$33.9 \pm 7.6\%$	$10.3 \pm 4.2\%$	$50.4 \pm 3\%$	$10.1 \pm 1.1\%$
2&2S, ST	$71.5 \pm 7.6\%$	$24.4 \pm 3.6\%$	$75.6 \pm 4.4\%$	$29.6 \pm 2.1\%$	$36.1 \pm 4.4\%$	$33.7 \pm 4.5\%$	$53.4 \pm 4.9\%$	$15 \pm 2.4\%$

Table 2. Transfer experiments in the Shapes World environment. In the same order as the examples in Figure 5, 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 trained each model 10 times over the same dataset; we report means and standard deviations.

Game	DQN	Value Iteration	Mean Q	Random
Breakout	14	0.83 ± 0.39	19.08 ± 11	0.66
Space Invaders	55	5.81 ± 3.12	20.52 ± 2.95	3.06
Freeway	54	34.95 ± 8.46	36.21 ± 8.08	0.2
Asterix	20	0.49 ± 0.08	0.53 ± 0.1	0.5

Table 3. DQN and abstract policies tested on MinAtar games. We train a DQN once for each game and report the mean return over the last 100 episodes. Then, we train our model for each game 10 times and report the mean returns and standard deviations for planning in the MDP (Value Iteration) and averaging predicted state-action values for each abstract state (Mean Q).

6.3. MinAtar

The challenge of the Shapes World (Subsection 6.2) is that the coarsest bisimulation can have thousands of abstract states. On the other hand, each task can be solved in less than ten time steps. The simplified Atari games of MinAtar pose an interesting challenge because each episode could potentially last tens or hundreds of time steps (Young & Tian, 2019). MinAtar has five Atari games–Breakout, Space Invaders, Freeway, Asterix and Seaquest[2]. The state of the games is fully observable and the dynamics are simplified. We use the same process of training a deep Q-network to create a dataset of transitions and then training our model on it. Appendix Subsection B.5 contains further details.

We test the quality of the learned abstraction in two ways. First, we employ standard Value Iteration in the learned abstract model to plan for the optimal policy. Planning in this domain might be challenging, as we do not use any temporal abstractions. We also test our abstraction from the perspective of compression: we average over the values of each state-action pair (predicted by the deep Q-network) belonging to each abstract state. This gives us a single value for each abstract-state action pair–we call this approach Mean Q. Intuitively, we compress the policy represented by the deep Q-network into a discrete representation.

Mean Q outperforms DQN in Breakout–an unexpected result–and reaches around 35% of the performance of DQN in Space Invaders and around 60% in Freeway (Table 3). Both of the Breakout policies suffer from a high variance of returns in-between episodes; we hypothesize that the compression makes the policy more robust. Figure 7 in Appendix C.2 further analyses Mean Q on Breakout. Value Iteration only works in Freeway and we fail to learn a useful abstraction in Asterix.

7. Conclusion

In this work, we present a new method for finding state abstractions from collected image states. We derive our objective function from the information bottleneck framework and learn abstract MDP through an HMM prior conditioned on actions. Our experiments demonstrate that our model is able to learn high-quality bisimulation partitions that contain up to 1000 abstract states. We also show that our abstractions enable transfer to goals not known during training. Our method fails gracefully in environments that have complexity greatly exceeding the capacity of our abstraction. Finally, we report experimental results on tasks with long time horizons, showing that we can use learned abstractions to compress DQN policies.

In future work, we plan to address the two main weaknesses of bisimulation: it does not leverage symmetries of the state-action space to minimize the size of the found abstraction and it does not scale with the temporal horizon of the task. The former problem can be addressed with MDP homomorphisms (Ravindran, 2004). The time horizon problems of bisimulation could be solved with hierarchical Reinforcement Learning.

[2] We skip Seaquest because we had trouble running it.

References

Abel, D., Hershkowitz, D. E., and Littman, M. L. Near optimal behavior via approximate state abstraction. In *Proceedings of the International Conference on Machine Learning*, pp. 2915–2923, 2016.

Abel, D., Arumugam, D., Asadi, K., Jinnai, Y., Littman, M. L., and Wong, L. L. S. State abstraction as compression in apprenticeship learning. In *AAAI*, 2019.

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

Bellman, R. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. ISSN 00959057, 19435274.

Biza, O. and Platt, R. Online abstraction with mdp homomorphisms for deep learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS 19, pp. 11251133, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

Castro, P. S. Scalable methods for computing state similarity in deterministic markov decision processes. *ArXiv*, abs/1911.09291, 2019.

Castro, P. S. and Precup, D. Using bisimulation for policy transfer in mdps. In *AAAI*, 2010.

Castro, P. S. and Precup, D. Automatic construction of temporally extended actions for mdps using bisimulation metrics. In *EWRL*, 2011.

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

Dean, T. L., Givan, R., and Leach, S. M. Model reduction techniques for computing approximately optimal solutions for markov decision processes. In *UAI*, 1997.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *AAAI*, 2004.

Ferns, N., Castro, P. S., Precup, D., and Panangaden, P. Methods for computing state similarity in markov decision processes. *ArXiv*, abs/1206.6836, 2006.

Givan, R., Dean, T., and Greig, M. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1):163 – 223, 2003. ISSN 0004-3702. doi: https://doi.org/10.1016/S0004-3702(02)00376-4. Planning with Uncertainty and Incomplete Information.

Goyal, A., Islam, R., Strouse, D., Ahmed, Z., Larochelle, H., Botvinick, M., Levine, S., and Bengio, Y. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations*, 2019.

Ha, D. and Schmidhuber, J. World models. *CoRR*, abs/1803.10122, 2018.

Ioffe, S. and Szegedy, C. 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*, pp. 448–456, 2015.

Kingma, D. P. and Ba, J. 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.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, Feb 2015.

Ravindran, B. *An Algebraic Approach to Abstraction in Reinforcement Learning*. PhD thesis, 2004. AAI3118325.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Xing, E. P. and Jebara, T. (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, June 2014. PMLR.

Rubin, J., Shamir, O., and Tishby, N. *Trading Value and Information in MDPs*, pp. 57–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-24647-0. doi: 10.1007/978-3-642-24647-0_3.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Strouse, D., Kleiman-Weiner, M., Tenenbaum, J., Botvinick, M., and Schwab, D. J. Learning to share and hide intentions using information regularization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 10249–10259. Curran Associates, Inc., 2018.

Teh, Y. W., Bapst, V., Czarnecki, W., Quan, J., Kirkpatrick, J., Hadsell, R., Heess, N. M. O., and Pascanu, R. Distral: Robust multitask reinforcement learning. In *NIPS*, 2017.

Tishby, N. and Polani, D. *Information Theory of Decisions and Actions*, pp. 601–636. Springer New York, New York, NY, 2011. ISBN 978-1-4419-1452-1. doi: 10.1007/978-1-4419-1452-1_19.

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

Young, K. and Tian, T. Minatar: An atari-inspired testbed for more efficient reinforcement learning experiments. *CoRR*, abs/1903.03176, 2019. URL http://arxiv.org/abs/1903.03176.

Supplementary Material: Learning discrete state abstractions with deep variational inference

A. Proof of Theorem 1

Theorem 1 (HMM-bisimulation theorem)**.** Given:

1. a ground MDP $M = \langle S, A, T, R, \gamma \rangle$,
2. the state-action value function Q^* of an optimal policy,
3. a set of optimal parameters of our model $\theta^* \in \arg\max L_{IB}$, $\beta \in (0, 1)$, and
4. an abstract $\bar{M} = \langle \bar{S}, A, \bar{T}, \bar{R}, \gamma \rangle$ induced by the HMM prior

and adhering to the idealized assumptions described in Section 5, there exists a bisimulation mapping from M to $\bar{M}$.

Proof by contradiction. Our strategy is to show that for every set of parameters $\theta \in \Theta_{no-bisim}$ of our model with an HMM prior that does not induce a bisimulation, we can find $\theta' \in \Theta$ with a higher L_{IB} that does induce a bisimulation. We break down our analysis into two cases: first, we show that for any model that does not preserve the reward function, we can find one that does with a higher L_{IB}. Then, we show that a reward-preserving model that violates transition dynamics of the ground environment is also suboptimal. We start by defining a SPLIT operation we use in every part of our analysis.

We define $\phi(s)$ as the abstraction function that maps each ground state to its mostly likely hidden state in the HMM prior, $\phi(s) = \arg\max p(\bar{s}|z)q(z|s)$. If $\phi(s_1) \neq \phi(s_2)$ for all states s_1 and s_2, ϕ is trivially a bisimulation. In other cases, we can always find a hidden state $\bar{s}$, such that $\phi(s) \neq \bar{s}$ for each state s because there are at least as many hidden states as ground states. We define SPLIT$(s_1, s_2, \bar{s})$ as an operation that takes states s_1 and s_2, such that $\phi(s_1) = \phi(s_2) = \bar{s}$. It finds an empty hidden state $\bar{s}'$ and assigns $z_2 = \mu_{\bar{s}'}$ (therefore, $\phi(s_2) = \bar{s}'$) and $p(\cdot|\bar{s}', a) = p(\cdot|\bar{s}, a)$, where $\mu_{\bar{s}}$ is the mean of the observation model corresponding to hidden state $\bar{s}$. For incoming transition probabilities, we set $p(\bar{s}|\bar{s}_0, a) = \frac{1}{\bar{s}_0} \sum_{s_0 \in \bar{s}_0} \left(\sum_{s \in \bar{s}} p(s|s_1, a) \right)$ for all abstract states $\bar{s}_0$ and actions a; $p(\bar{s}'|\bar{s}_0, a)$ is set analogously. We also assigns $z_1 = \mu_{\bar{s}}$ if that is not already the case and assume $\mu_{\bar{s}} \neq \mu_{\bar{s}'}$.

Our objective function has two components:

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

Further analysing the second term, the encoder distributions $q(z_t|s_t)$ and $q(z_{t+1}|s_{t+1})$ turn into Dirac delta functions because their covariance is set to zero. The term $p(z_t, z_{t+1}|a)$ can be decomposed as:

$$\sum_{\bar{s}_t, \bar{s}_{t+1}} p(z_t|\bar{s}_t)p(z_{t+1}|\bar{s}_{t+1})p(\bar{s}_{t+1}|\bar{s}_t, a)p(\bar{s}_t)$$

Since the covariance of the hidden states is fixed to zero, $p(z_t|\bar{s}_t)$ is maximized if $z_t = \mu_{\bar{s}_t}$, where $\mu_{\bar{s}_t}$ is the mean of the observation model for hidden state $\bar{s}_t$. The term $p(\bar{s}_t)$ is held uniform fixed.

Assume that we have θ^* such that $\phi(s_1) = \phi(s_2) = \bar{s}$ implies $r(s_1, a) \neq r(s_2, a)$ for some pair of states $s_1, s_2 \in S$ and some action $a \in A$. Since M is episodic and has sparse rewards, $Q^*(s_1, a) \neq Q^*(s_2, a)$. We analyze two cases:

1. $z_1 = z_2 = \mu_{\bar{s}}$: in this case, $p(y|z_1, a)$ cannot distinguish between z_1 and z_2 given a new encoding z_3 such that $z_1 = z_2$ are the nearest neighbors of z_3. We can increase the first component of the objective by executing SPLIT$(s_1, s_2, \bar{s})$, the second term of the objective stays fixed or increases if the model can better simulate the ground transition dynamics.

2. $z_1 \neq \mu_{\bar{s}}$ or $z_2 \neq \mu_{\bar{s}}$: we can increase the value of the second component of the objective function by executing SPLIT$(s_1, s_2, \bar{s})$. The first term stays fixed because we can still distinguish between z_1 and z_2, and the second term increase because $p(z_1|\bar{s}_1)$ or $p(z_2|\bar{s}_2)$ increases and the rest stays the same.

In both cases, θ^* is not an optimal solution. Next, we consider the case where $z = \mu_{\bar{s}}$ for each encoding z and some hidden state $\bar{s}$, and $\phi(s_1) = \phi(s_2)$ implies $r(s_1, a) = r(s_2, a)$ and $\sum_{s' \in \phi^{-1}(\bar{s})} p(s_1, a, s') \neq \sum_{s' \in \phi^{-1}(\bar{s})} p(s_2, a, s')$. This implies that there exists a state s' such that $p(s_1, a, s') \neq p(s_2, a, s')$. We can increase the term $p(\bar{s}_{t+1}|\bar{s}_t, a)$ with SPLIT$(s_1, s_2, \bar{s})$ while keeping the other terms fixed. Hence, θ^* is again suboptimal.

We have shown that for every θ that does not induce a bisimulation, we can find a θ' with a higher L_{IB} that does induce a bisimulation.

□

B. Experimental Details

We ran all of our experiments on a machine with Intel Core i7-9700K CPU @ 3.60GHz, 64GB of RAM and two Nvidia GeForce RTX 2080 Ti graphics cards.

B.1. Abstraction purity

We include a snippet of the Python code that computes abstraction purity. We use the package numpy version 1.16.1. The inputs to the function below are probability distribution over hidden states for each sample in the validation dataset and an array of labels, one for each sample.

```
import numpy as np

def evaluate_purity(self, cluster_probs, labels):
  """
  :param cluster_probs:       NxK matrix where N is the number of samples and K the number
                                                 of components.
  :param labels:              A label for each sample.
  """

  # compute the probability of each component-label pair
  label_masses = []

  for label in np.unique(labels):
      # find all samples with a particular label and sum over them
      label_mass = np.sum(cluster_probs[labels == label], axis=0)
      label_masses.append(label_mass)

  label_masses = np.stack(label_masses)

  sizes = np.sum(cluster_probs, axis=0)
  sizes[sizes == 0.0] = 1.0

  # assign a label with the highest probability mass to each cluster
  # calculate the fraction of that mass to the mass of all other labels
  purities = np.max(label_masses, axis=0) / sizes

  # average of cluster purities weighted by cluster sizes
  mean_purity = np.sum(purities * sizes) / np.sum(sizes)

  return purities, sizes, mean_purity
```

B.2. 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

Source tasks	2SB	3SB	3T	3B	3L	3R
2SB	99.3 ± 0.6	$0.5 \pm 0.2\%$	$38 \pm 2.5\%$	$38.5 \pm 2.8\%$	$41.5 \pm 6\%$	$40.7 \pm 2.6\%$
3SB	$10 \pm 2.5\%$	$61.2 \pm 26.4\%$	$3.7 \pm 0.3\%$	$3.9 \pm 0.9\%$	$3.6 \pm 0.7\%$	$2.7 \pm 2.6\%$
3T	$55.2 \pm 3.2\%$	$0.6 \pm 0.1\%$	$82.4 \pm 5.1\%$	$70.2 \pm 5.3\%$	$32 \pm 2.9\%$	$26 \pm 4.2\%$
3L	$57.2 \pm 6.5\%$	$2.6 \pm 3.6\%$	$23.1 \pm 5.2\%$	$28.5 \pm 6.4\%$	$84.6 \pm 7.1\%$	$71.7 \pm 8.4\%$
3SB, 3T	$36.5 \pm 2.9\%$	$74.6 \pm 6.6\%$	$68.8 \pm 1\%$	$39.4 \pm 5.4\%$	$29.7 \pm 4.3\%$	$31.3 \pm 3.1\%$
3T, 3L	$64.1 \pm 1.9\%$	$6.4 \pm 7.6\%$	$78.8 \pm 4.5\%$	$75 \pm 5.2\%$	$79.7 \pm 3.6\%$	$74.3 \pm 4.3\%$
3SB, 3T, 3L	$58.1 \pm 0.7\%$	$60.8 \pm 5.2\%$	$62.7 \pm 2.5\%$	$56.5 \pm 3.1\%$	$68.2 \pm 2.2\%$	$49.3 \pm 6.2\%$

Table 4. Transfer experiments in the Building World domain. The setup is the same as in Table 2. The tasks are to build a building from a block and a roof (2SB), building from two stacked blocks and a roof (3SB), 2SB with an block added from top (3T), bottom (3B), left (3L) and right (3R). See Figure 6 for examples of goal states.

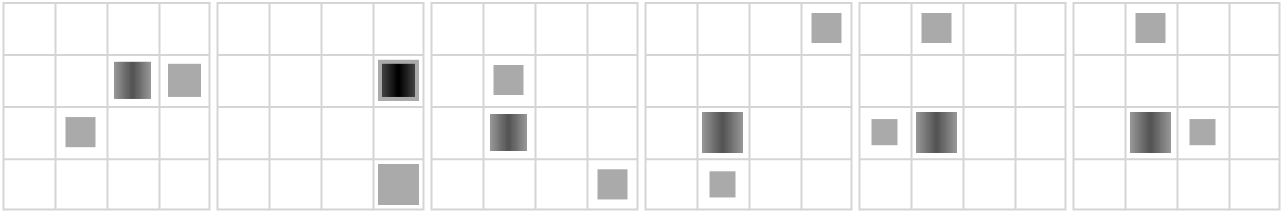

Figure 6. Goal states in the Building World environment. From left to right: stacking a roof on one block, stacking two blocks on top of each other and a roof on top, a roof stacked on top of a block and another block added either from the top, bottom, left or right. We show simulated depth images with inverted luminosity (the darker a pixel the higher it is); the agent does not see the grid.

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 & Ba, 2015).

B.3. 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 & 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 & 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.

B.4. Stacking Buildings

The hyper-parameters are the same as for Shapes World (Subsection B.3).

B.5. MinAtar

We use a deep Q-network architecture and a training script provided by Young & Tian (2019). We collect 100000 transitions with ϵ set to 0.1 after training it for 3M time steps. The authors train up to 5M time steps, but we disable sticky actions and difficulty ramping, making the games easier.

The details of our model are similar to Subsection B.3, except we use a smaller convolutional network with 32 and 64 filters in two layers, filter size set to three and stride set to one. We do not use batch normalization and the hidden layer after convolutions has only 128 neurons. β is set to $5 * 10^{-5}$ and the rest of the parameters stay the same. For our abstract agent, we do not threshold goal states and set τ to $5 * 10^{-4}$.

C. Additional Experiments

C.1. Stacking buildings

The set up for this experiment is the same as Shapes World (Subsection 6.2). We instantiate five different tasks (Figure 6) and report our transfer results in Table 4. The tasks are more difficult than the ones in Shapes World. All tasks except for the first have too many abstract states in the coarsest bisimulation to be represented by the 1000 hidden states available in the HMM prior.

One data point of interest is our models ability to generalize between different orientations of buildings (Figure 6, images 3 (3T–top), 4 (3B–bottom), 5 (3L–left), and 6 (3R–right)). The abstraction trained on the 3T task can generalize to 3B, but not to 3L and 3R. Conversely, 3L can generalize to 3R, but not 3R and 3L. Training on 3T and 3L leads to an abstraction that can solve both 3B and 3R (Table 4 line 6), albeit not as well as the abstractions from 3T (line 3) and 3L (line 4) separately.

C.2. MinAtar

In Figure 7, we investigate the impact of the number of abstract states on the performance of the Mean Q abstract agent (Subsection 6.3) in Breakout. Even though there is a high variance between the qualities of abstraction learned in different runs, the violin plot shows an approximately linear dependence between the number of abstract states and the mean return.

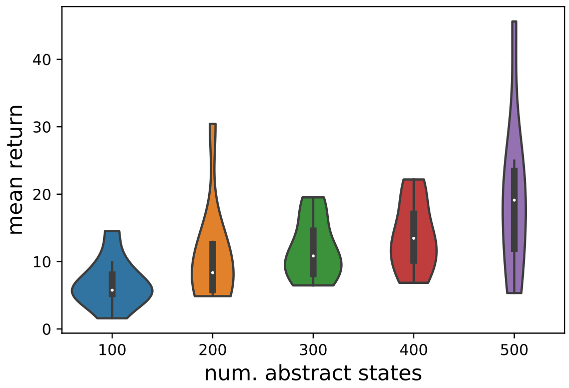

Figure 7. Mean return of the Mean Q agent as a function of the number of abstract states in the HMM prior in Breakout. Each setting was run 10 times and we report the results as a violin plot. We cut the plots at the minimum and maximum values of data points.