Distilling Symbols from Perceptual World Models

Lucas Saldyt, Maxime Zand
School of Computing and Augmented Intelligence
Arizona State University
Tempe, AZ 85202
{lsaldyt, mzand}@asu.edu

1 Introduction

Perceptive world models are highly efficient and principled basis for robot learning, planning, and control. However, black-box models (such as those that use dense real-number vector embeddings to represent world states) are fundamentally uninterpretable, because they were not designed to be. However, these latent vectors presumably capture the state-space sufficiently for prediction, and it may be possible to recover symbols from them. Accordingly, this work explores the extent to which latent symbols form naturally within a predictive world model. Furthermore, it proposes architectural changes and new training objectives meant to potentially elicit symbols.

The purpose of this project is to distill symbols out of the latent vectors learned by the Dreamer family of models. One approach utilizes a sparse autoencoder of the latent vectors, creating a binary vector potentially mapping to internal concepts. Then, we analyze the degree to which this binary vector maps to real concepts from ground-truth data, for instance via mutual information. This is compared to the default dense state vector. Furthermore, mutual information can be converted into a regularization objective itself, as in InfoGAN. Using this analysis as a basis, we propose and study further architectural changes to the model, and methods for recovering human symbols.

2 Related Work

Dreamer learns a recurrent state-space model, which learns both a deterministic and stochastic world model based on perceptual input [1, 2, 3]:

$$h_t = f(h_{t-1}, z_{t-1}, a_{t-1}) \qquad z_t \sim q_\phi(z_t|h_t, x_t) \qquad \hat{x}_t \sim p_\phi(\hat{x}_t|h_t, z_t) \tag{1}$$

Where x is a perceptual input, a is an agents action, h is a deterministic component of the world model, and z is a stochastic component of the world model. In particular, we are interested in both h and z, and the degree to which they encode human-like symbols. By default, the Dreamer algorithm is not intended to be interpretable, but we hypothesize that these vectors contain the same information as symbolic representations of the state-space, indicating that it may be possible to recover human-like symbols from them, or alter Dreamer to become interpretable naturally.

Classical Planning requires categorical variables, which can be produced by concept classifiers [4]. Like in InfoGAN (discussed below), symbols distilled from dreamer may potentially be used as concept classifiers, making these domains amenable to classical planning methods. While Dreamer relies on non-classical planning in latent space, creating intermediate categorical representations would enable more conventional planning, or at least an interpretable form of hybrid planning.

Variational Autoencoders force latents to be represented as gaussians. In practice, this gives some degree of disentanglement to the latent representation [5].

Sparse Autoencoders have been used to find interpretable features in large language models [6]. The proposed method simply introduces a p-norm regularization term to the autoencoding objective.

Preprint. Under review.

InfoGAN modifies the training objectives of a Generative Adversarial Network (GAN) to encourage the formation of symbols [7]. In particular, a lower bound on mutual information is converted into a regularization term for the autoencoding objective. In practice, categorical latents from InfoGAN can be used as effective classifiers, for instance on the MNIST dataset where they have 95% accuracy even though they were never supervised. Hopefully, these concept classifiers could be used as planning predicates, for instance in our proposed alteration to dreamer which mimics InfoGAN.

Gumbel Softmax introduces a method for producing categorical variables differentiably [8]. Similarly, DreamerV2 uses categorical latents on Atari domains [2].

Other related work includes Neural Discrete Representation Learning, Concept Bottleneck Models, and Compositional Neural Feature Fields [9, 10, 11].

3 Methods

3.1 Dreamer Objectives (Reference)

Recall that the Dreamer algorithm is trained with the following three objectives. Relative to the notation in the paper, we have renamed their continuation flag to u_t, opting to use c_t to represent the symbolic latent code. Then, $\mathrm{sg}(\cdot)$ refers to the stop gradient operator, and r_t is a reward prediction.

$$\underset{\text{pred.}}{\mathcal{L}}(\phi) = -\ln p_\phi(x_t|z_t,h_t) - \ln p_\phi(r_t|z_t,h_t) - \ln p_\phi(u_t|z_t,h_t) \tag{2}$$

$$\underset{\text{dyn.}}{\mathcal{L}}(\phi) = \max(1, \mathrm{KL}[\mathrm{sg}(q_\phi(z_t|h_t,x_t))||p_\phi(z_t|h_t)]) \tag{3}$$

$$\underset{\text{repr.}}{\mathcal{L}}(\phi) = \max(1, \mathrm{KL}[q_\phi(z_t|h_t,x_t)||\,\mathrm{sg}(p_\phi(z_t|h_t))]) \tag{4}$$

Which are combined using $\underset{\text{pred.}}{\beta} = 1$, $\underset{\text{dyn.}}{\beta} = 0.5$ and $\underset{\text{repr.}}{\beta} = 0.1$.

$$\mathcal{L}(\phi) = \mathop{\mathbb{E}}_{q_\phi}\left[\sum_{t=1}^{T}\left(\underset{\text{prediction}}{\beta\mathcal{L}} + \underset{\text{dynamics}}{\beta\mathcal{L}} + \underset{\text{representation}}{\beta\mathcal{L}}\right)\right] \tag{5}$$

3.2 Sparsity Regularized Codes

The architecture of Dreamer already contains an autoencoder, in particular in the inputs. In DreamerV2, discrete intermediate representations are used. However, there are not experiments with enforcing the latent representation to be *sparse* through regularization. In its most basic form, this can be done with an regularization term on the latent representation. We propose leaving Dreamer's z_t and h_t unaltered, and introducing a new learned latent code $\hat{c}_t$, so the new objective becomes:

$$\underset{\text{sparsity}}{\mathcal{L}}(\phi) = \underset{\text{sparsity}}{\beta}\,\|\hat{c}_t\|_k \tag{6}$$

Which can be added to the overall loss $\mathcal{L}(\phi)$ with an appropriate choice of $\underset{\text{sparsity}}{\beta}$.

3.3 Mutual Information Regularized Codes

InfoGAN introduced latent symbols into their model, and attempted to predict them from the latents in the GAN discriminator. In the Dreamer architecture, there are several potential locations that a similar pattern could be used. Unlike GANs, Dreamer does not have a native input of random noise. Furthermore, Dreamer is required to maintain consistency across a large number of timesteps, unlike GAN which can generate individual images independently.

The most logical approach may be to introduce a symbolic latent code c_t for each timestep, which is an input for creating h_t and z_t, and to try to re-predict this latent code using h_t and z_t as inputs:

$$\hat{c}_t \sim d(c_t|z_t,h_t) \qquad h_t = f(h_{t-1},z_{t-1},a_{t-1},c_{t-1}) \qquad z_t \sim q_\phi(z_t|h_t,x_t,c_t) \tag{7}$$

By using the mutual information proxy defined in InfoGAN, this can be approximated using:

$$\underset{\text{info}}{\mathcal{L}}(\phi) = \mathop{\mathbb{E}}_{z_t\sim p_\phi(\cdot),h_t\sim q_\phi(\cdot)}\;\mathop{\mathbb{E}}_{\hat{c}_t\sim d(c_t|z_t,h_t)}[\log d(\hat{c}_t|z_t,h_t)] + H(c_t) \tag{8}$$

Similarly, $\underset{\text{info}}{\mathcal{L}}$ can be added to the main loss $\mathcal{L}(\phi)$ with an appropriate $\underset{\text{info}}{\beta}$ to scale it.

However, it is desirable for there to be temporal consistency in the latent code, e.g. codes c_t and c_{t-1} should not be independent. Instead, to generate correlated codes, we propose starting with an initial random code for each episode, and applying random edits (as bit flips) to the code. For non-categorical distributions, we similarly apply random edits, for instance we perturb the mean and variance of gaussians.

Beyond enforcing temporal consistency through edits, we experiment with generating a random sparse vector using an autoencoder, thresholding this vector to be perfectly sparse, and then info-regularizing this vector with a re-prediction of itself done later on in the network. For instance we predict a latent l_t by autoencoding h_t to $\hat{h}_t$ and then later try to predict $\hat{l}_t$.

4 Experiments / Results

To study symbols within Dreamer, we obtain a ground truth vector of symbols $c_t^\star$. In the example Crafter domain, this encodes information like the types of tiles in the scene (e.g. classes like grass, wood, or rock). In general, this is a vector that corresponds to a perfect observation and potentially to hidden information beyond the scene. For instance, in crafter the agent's view is limited, but a perfect-memory agent would encode information about tiles that are beyond the view distance but were seen already. Accordingly, we have replicated the DreamerV3 algorithm, and studied the correlation between the latent vectors h_t and z_t and the ground truth symbolic states $c_t^\star$. Note that DreamerV3 is never directly trained on the symbols in $c^\star$. Instead, any learning of them comes from x (input images), which are rendered by the game engine (for instance wood is rendered with a particular texture).

Beyond the baseline of studying existing correlation of Dreamer latents (our baseline), we experiment with two separate methods for distilling symbols: sparse autoencoding and mutual information regularization. In these cases, the model now has an approximate symbolic latent code, $\hat{c}_t$, which we add to the comparison with $c_t^\star$

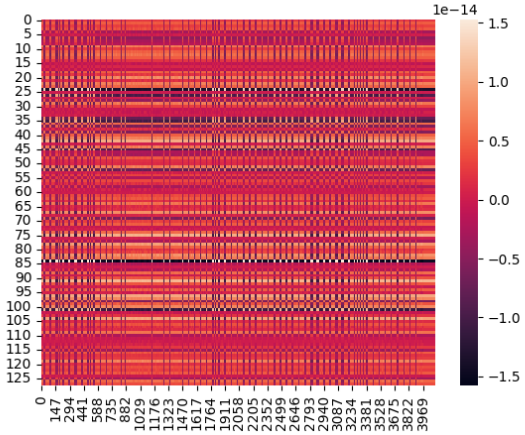

Figure 1: Dense Encodings of Symbols within Crafter (no regularizations)

Figure 1 shows an example of the correlation matrix between a dreamer latent h (y axis) and symbols in the crafter environment (x axis). Overall, the model is encoding materials, but the encoding is spread between many different variables within the latent representation h. In contrast, Figure 2 shows a correlation matrix from the sparse autoencoding method. Like Figure 1, the y axis is the latent vector h, and the x axis is the ground truth symbols. As expected, we find that encouraging sparsity forces the model encode environment materials into a single variable in h.

Overall, the goal of this paper is qualitative. So long as overall performance has not deteriorated significantly, finding sparse symbols is a victory. However, in Figure 3, we find slightly degraded

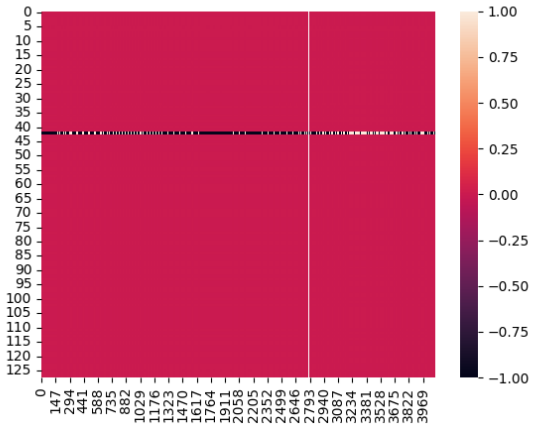

Figure 2: Sparse Encodings of Symbols within Crafter (sparse autoencoder)

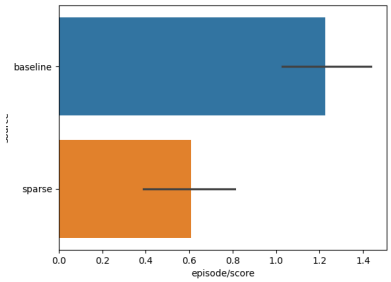

Figure 3: Reward comparison with sparse autoencoder

rewards overall from sparse autoencoding. This result is not particularly surprising, given the sparsity shown in Figure 2. It is possible that the scalar β_{sparsity} is too high, and simply tuning it could produce better results. However, we hypothesize that the proposed mutual information regularization will have overall better performance than sparse autoencoding (and potentially better than the baseline as well, as in InfoGAN).

5 Future Work

Future work will perform comparisons beyond simple correlation, and with more ground truth symbols. The objective of finding these correspondences is to perform planning within the symbolic latent space. Furthermore, we will implement an architecture which regularizes against a proxy objective for mutual information, in the spirit of InfoGAN. To do so, we will structure c_t, for instance as a concatenation of categorical and gaussian random variables. Then, we will introduce dynamics predictions in terms of c_t with a separate function g:

$$f: \quad h_{t-1}, a_{t-1}, z_{t-1} \quad \mapsto \quad h_t \quad \text{Latent dynamics} \tag{9}$$

$$g: \quad c_{t-1}, a_{t-1}, z_{t-1} \quad \mapsto \quad c_t \quad \text{Symbolic dynamics} \tag{10}$$

From this, we can repredict a thresholded (sparse) $\dot{c}_t$ from $\hat{c}_t = p(z_t, h_t)$, which is a lower bound on mutual information and will effectively regularize the network.

In general, this work would like to further explore the role of latent symbol formation in any unsupervised learning system. For example, previous work has done similar mechanistic interpretability for large language models. Beyond studying existing systems, it is far more interesting to propose new architectures and training objectives that encourage interpretability.

References

[1] Danijar Hafner et al. “Learning latent dynamics for planning from pixels”. In: *International conference on machine learning*. PMLR. 2019, pp. 2555–2565.

[2] Danijar Hafner et al. “Mastering atari with discrete world models”. In: *arXiv preprint arXiv:2010.02193* (2020).

[3] Philipp Wu et al. “Daydreamer: World models for physical robot learning”. In: *Conference on Robot Learning*. PMLR. 2023, pp. 2226–2240.

[4] George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. “Constructing symbolic representations for high-level planning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

[5] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[6] Hoagy Cunningham et al. “Sparse autoencoders find highly interpretable features in language models”. In: *arXiv preprint arXiv:2309.08600* (2023).

[7] Xi Chen et al. “Infogan: Interpretable representation learning by information maximizing generative adversarial nets”. In: *Advances in neural information processing systems* 29 (2016).

[8] Eric Jang, Shixiang Gu, and Ben Poole. “Categorical reparameterization with gumbel-softmax”. In: *arXiv preprint arXiv:1611.01144* (2016).

[9] Aaron Van Den Oord, Oriol Vinyals, et al. “Neural discrete representation learning”. In: *Advances in neural information processing systems* 30 (2017).

[10] Pang Wei Koh et al. “Concept bottleneck models”. In: *International conference on machine learning*. PMLR. 2020, pp. 5338–5348.

[11] Michael Niemeyer and Andreas Geiger. “Giraffe: Representing scenes as compositional generative neural feature fields”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11453–11464.