Switching Linear Dynamics for Variational Bayes Filtering

Philip Becker-Ehmck [1 2] **Jan Peters** [2] **Patrick van der Smagt** [1]

Abstract

System identification of complex and nonlinear systems is a central problem for model predictive control and model-based reinforcement learning. Despite their complexity, such systems can often be approximated well by a set of linear dynamical systems if broken into appropriate subsequences. This mechanism not only helps us find good approximations of dynamics, but also gives us deeper insight into the underlying system. Leveraging Bayesian inference, Variational Autoencoders and Concrete relaxations, we show how to learn a richer and more meaningful state space, e.g. encoding joint constraints and collisions with walls in a maze, from partial and high-dimensional observations. This representation translates into a gain of accuracy of learned dynamics showcased on various simulated tasks.

1. Introduction

Learning dynamics from raw data (also known as system identification) is a key component of model predictive control and model-based reinforcement learning. Problematically, environments of interest often give rise to very complex and highly nonlinear dynamics which are seemingly difficult to approximate. However, switching linear dynamical systems (SLDS) approaches claim that those environments can often be broken down into simpler units made up of areas of equal and linear dynamics (Ackerson & Fu, 1970; Chang & Athans, 1978).

Not only have those approaches demonstrated good predictive performance in various settings, which often is the sole goal of learning a system's dynamics, they also encode useful information into so called *switching variables* which determine the dynamics of the next transition. For

[1]Machine Learning Research Lab, Volkswagen Group, Munich, Germany [2]Department of Computer Science, Technische Universität Darmstadt, Darmstadt, Germany. Correspondence to: Philip Becker-Ehmck <philip.becker-ehmck@volkswagen.de>.

example, when looking at the movement of an arm, one is intuitively aware of certain restrictions of possible movements, e.g., constraints to the movement due to joint constraints or obstacles. The knowledge is present without the need to simulate; it is explicit. Exactly this kind of information will be encoded when successfully inferring switching variables. Our goal in this work will therefore entail the search for richer representations in the form of latent state space models which encode knowledge about the underlying system dynamics. In turn, we expect this to improve the accuracy of our simulation as well. Such a representation alone could then be used in a reinforcement learning approach that possibly only takes advantage of the learned latent features but not necessarily its learned dynamics.

To learn richer representations, we identify one common problem with prevalent recurrent Variational Autoencoder models (Chung et al., 2015; Fraccaro et al., 2016; Karl et al., 2017a; Krishnan et al., 2015): the non-probabilistic treatment of the transition dynamics often modelled by a powerful nonlinear function approximator. From the history of the Autoencoder to the Variational Autoencoder, we know that in order to detect robust features in an unsupervised manner, probabilistic treatment of the latent space is paramount (Dai et al., 2017; Kingma et al., 2014). As our starting point, we will build on previously proposed approaches by Krishnan et al. (2017) and Karl et al. (2017a). The latter already made use of locally linear dynamics, but only in a deterministic fashion. We extend their approaches by a stochastic SLDS model with structured inference and show that such treatment is vital for learning richer representations and simulation accuracy.

2. Background

We consider discretized time-series data consisting of continuous observations $x_t \in \mathcal{X} \subset \mathbb{R}^{n_x}$ and control inputs $u_t \in \mathcal{U} \subset \mathbb{R}^{n_u}$ that we would like to model by corresponding latent states $z_t \in \mathcal{Z} \subset \mathbb{R}^{n_z}$. We will denote sequences of variables by $x_{1:T} = (x_1, x_2, ..., x_T)$.

2.1. Switching Linear Dynamical Systems

A Switching Linear Dynamical Systems (SLDS) model enables us to model nonlinear time series data by splitting it into subsequences governed by linear dynamics. At each
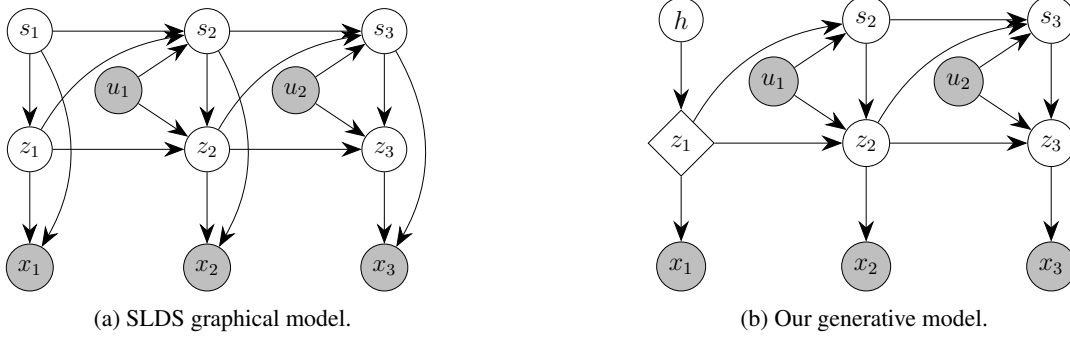
(a) SLDS graphical model.

(b) Our generative model.

Figure 1: (a) s_t denote discrete switch variables, z_t are continuous latent variables, x_t are continuous observed variables, u_t are (optional) continuous control inputs. (b) By introducing a special latent variable h used for initial state inference, we want to make explicit that the first step is treated differently from the rest of the sequence.

time $t = 1, 2, ..., T$, a discrete switch variable $s_t \in 1, ..., M$ chooses of a set of M LDSs a system which is to be used to transform the continuous latent state z_{t-1} to the next time step (Barber, 2012):

$$
\begin{aligned}
z_t &= A(s_t)z_{t-1} + B(s_t)u_{t-1} + \epsilon(s_t) \\
x_t &= H(s_t)z_t + \eta(s_t) \\
\text{with} \quad &\eta(s_t) \sim \mathcal{N}(0, R(s_t)), \; \epsilon(s_t) \sim \mathcal{N}(0, Q(s_t))
\end{aligned}
\tag{1}
$$

Here $A \in \mathbb{R}^{n_z \times n_z}$ is the state matrix, $B \in \mathbb{R}^{n_z \times n_u}$ is the control matrix, ϵ is the transition noise with covariance matrix Q, η is the emission/sensor noise with covariance matrix R and the observation matrix $H \in \mathbb{R}^{n_x \times n_z}$ defines a linear mapping from latent to observation space. These equations imply the following joint distribution:

$$
\begin{aligned}
p(x_{1:T}, z_{1:T}, s_{1:T} \mid u_{1:T}) = \prod_{t=1}^{T} p(x_t \mid z_t, s_t) \\
p(z_t \mid z_{t-1}, u_{t-1}, s_t) \, p(s_t \mid z_{t-1}, u_{t-1}, s_{t-1})
\end{aligned}
\tag{2}
$$

with $p(s_1 \mid z_0, u_0, s_0) = p(s_1)$ and $p(z_1 \mid z_0, u_0, s_1) = p(z_1 \mid s_1)$ being initial state distributions. The discrete switching variables are usually assumed to evolve according to Markovian dynamics, i.e. $\Pr(s_t \mid s_{t-1} = k) = \pi_k$, which optionally may be conditioned on the continuous state z_{t-1}. The corresponding graphical model is shown in figure 1a.

2.2. Stochastic Gradient Variational Bayes

$$
p(x) = \int p(x, z) \, \mathrm{d}z = \int p(x \mid z)p(z) \, \mathrm{d}z
\tag{3}
$$

Given the simple graphical model in equation (3), Kingma & Welling (2014) and Rezende et al. (2014) introduced the *Variational Autoencoder* (VAE) which overcomes the intractability of posterior inference of $p(z \mid x)$ by maximizing the evidence lower bound (ELBO) of the model

log-likelihood:

$$
\begin{aligned}
\log p(x) &\geq \mathcal{L}_{\text{ELBO}}(x; \theta, \phi) \\
&= \mathbb{E}_{q_\phi(z\mid x)}[\ln p_\theta(x \mid z)] - \text{KL}(q_\phi(z \mid x) \,\|\, p(z)).
\end{aligned}
\tag{4}
$$

Their main innovation was to approximate the intractable posterior $p(z \mid x)$ by a *recognition network* $q_\phi(z|x)$ from which they can sample via the *reparameterization trick* to allow for stochastic backpropagation through both the recognition and generative model at once. Assuming that the latent state is normally distributed, a simple transformation allows us to obtain a Monte Carlo gradient estimate of $\mathbb{E}_{q_\phi(z|x)}[\ln p_\theta(x|z)]$ w.r.t. ϕ. Given that $z \sim \mathcal{N}(\mu, \sigma^2)$, we can generate samples by drawing from an auxiliary variable $\epsilon \sim \mathcal{N}(0, 1)$ and applying the deterministic and differentiable transformation $z = \mu + \sigma\epsilon$.

2.3. The Concrete Distribution

One simple and efficient way to obtain samples d from a k-dimensional categorical distribution with class probabilities α is the Gumbel-Max trick:

$$
\begin{aligned}
d &= \text{one\_hot}\left(\arg\max[g_i + \log \alpha_i]\right) \\
&\text{with} \; g_1, \ldots, g_k \sim \text{Gumbel}(0, 1).
\end{aligned}
\tag{5}
$$

However, since the derivative of the argmax is 0 everywhere except at the boundary of state changes, where it is undefined, we cannot learn a parameterization by backpropagation. The Gumbel-Softmax trick approximates the argmax by a softmax which gives us a probability vector (Jang et al., 2017; Maddison et al., 2017). We can then draw samples via

$$
d_k = \frac{\exp((\log \alpha_k + g_k)/\lambda)}{\sum_{i=1}^{n} \exp((\log \alpha_i + g_i)/\lambda)}.
\tag{6}
$$

This softmax computation approaches the discrete argmax as temperature $\lambda \to 0$ while for $\lambda \to \infty$ it approaches a uniform distribution. In the same vein, the bias of the estimator decreases with decreasing λ while its variance increases.

3. Related Work

Our model can be viewed as a Deep Kalman Filter (Krishnan et al., 2015) with structured inference (Krishnan et al., 2017). In our case, structured inference entails another stochastic variable model with parameter sharing inspired by Karl et al. (2017b) and Karl et al. (2017a) which pointed out the importance of backpropagating the reconstruction error through the generative transition. Marino et al. (2018) adopts this idea by starting an iterative amortized inference scheme with the prior prediction. Iterative updates are then only conditioned on the gradient of the loss, allowing the observation only to adjust, but not completely overrule, the prior prediction. We are different to a number of stochastic sequential models like Bayer & Osendorfer (2014); Chung et al. (2015); Goyal et al. (2017); Shabanian et al. (2017) by directly transitioning the stochastic latent variable over time instead of having an RNN augmented by stochastic inputs. Fraccaro et al. (2016) proposes a transition over both a deterministic and a stochastic latent state sequence, wanting to combine the best of both worlds.

Previous models (Fraccaro et al., 2017; Karl et al., 2017a; Watter et al., 2015) have already combined locally linear models with recurrent Variational Autoencoders, however they provide a weaker structural incentive for learning latent variables determining the transition function. Van Steenkiste et al. (2018) approach a similar multi bouncing ball problem (see section 5.1) by first distributing the representation of different balls into their own entities without supervision and then structurally hardwiring a transition function with interactions based on an attention mechanism.

Recurrent switching linear dynamical systems (Linderman et al., 2017) uses message passing for approximate inference, but has restricted itself to low-dimensional observations and a multi-stage training process. Nassar et al. (2019) builds on this work and suggests a tree structure for enforcing a locality prior on switching variables where subtrees share similar dynamics. Johnson et al. (2016) propose a similar model to ours but combine message passing for discrete switching variables with a neural network encoder for observations learned by stochastic backpropagation.

One feature an SLDS model may learn are interactions which have recently been approached by employing Graph Neural Networks (Battaglia et al., 2016; Kipf et al., 2018). These methods are similar in that they predict edges which encode interactions between components of the state space (nodes). Tackling the problem of propagating state uncertainty over time, various combinations of neural networks for inference and Gaussian processes for transition dynamics have been proposed (Doerr et al., 2018; Eleftheriadis et al., 2017). However, these models have not been demonstrated to work with high-dimensional observation spaces like images.

4. Proposed Approach

We propose learning an SLDS model through a recurrent Variational Autoencoder framework which approximates switching variables by a Concrete distribution (Jang et al., 2017; Maddison et al., 2017). This leads to a model that can be optimized entirely by stochastic backpropagation through time. For inference, we propose a time-factorized approach with a specific computational structure, reusing the generative model. This allows us to learn good transition dynamics. Our generative model is shown in figure 1b an our inference model in figure 2a.

4.1. Generative Model

Our generative model for a single x_t is described by

$$p(x_t) = \int_{s_{\leq t}} \int_{z_{\leq t}} p(x_t \mid z_t)p(z_t \mid z_{t-1}, s_t, u_{t-1}) \tag{7}$$
$$p(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})p(z_{t-1}, s_{t-1})$$

which is close to the original SLDS model (see figure 1a). Latent states z_t are continuous and represent the state of the system while states s_t are switching variables determining the transition. In order to use end-to-end backpropagation, we approximate the discrete switching variables by a continuous relaxation, namely the Concrete distribution. Differently to the original model, we do not condition the likelihood of the current observation $p_\theta(x_t \mid z_t)$ directly on the switching variables. This limits the influence of the switching variables to choosing a proper transition dynamic for the continuous latent space. The likelihood model is parameterized by a neural network with, depending on the data, either a Gaussian or a Bernoulli distribution as output.

As outlined in (7), we need to learn separate transition functions for the continuous states z_t and for the discrete states s_t. For the continuous state transition $p(z_t \mid z_{t-1}, s_t, u_{t-1})$ we follow (1) and maintain a set of M base matrices $\{(A^{(i)}, B^{(i)}, Q^{(i)}) \mid \forall i, \ 0 < i < M\}$ as our linear dynamical systems to choose from:

$$p_\theta(z_t \mid z_{t-1}, s_t, u_{t-1}) = \mathcal{N}(\mu, \sigma^2) \tag{8}$$
$$\text{where } \mu = A_\theta(s_t)z_{t-1} + B_\theta(s_t)u_t, \ \sigma^2 = Q_\theta(s_t).$$

For the transition on discrete latent states $p(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})$, we conventionally require a Markov transition matrix. However, since we approximate our discrete switching variables by a continuous relaxation, we can parameterize this transition by a neural network:

$$p_\theta(s_t \mid s_{t-1}, z_{t-1}, u_{t-1}) = \text{Concrete}(\alpha, \lambda_{\text{prior}}) \tag{9}$$
$$\text{where } \alpha = g_\theta(z_{t-1}, s_{t-1}, u_{t-1}).$$

Finally, the question arises how we determine our transition matrices A, B and Q since our Concrete samples s_t are now

probability vectors and not one-hot vectors anymore. We could execute the forward pass by choosing the linear system corresponding to the highest value in the sample (hard forward pass) and only use the relaxation for our backward pass. This, however, means that we would follow a biased gradient. Alternatively, we can use the relaxed version for our forward pass and aggregate the linear systems based on their corresponding weighting:

$$A_t(s_t) = \sum_{i=1}^{M} s_t^{(i)} A^{(i)}, \; B_t(s_t) = \sum_{i=1}^{M} s_t^{(i)} B^{(i)},$$
$$Q_t(s_t) = \sum_{i=1}^{M} s_t^{(i)} Q^{(i)}. \tag{10}$$

Here, we lose the discrete switching of linear systems, but maintain a valid lower bound. We note that the hard forward pass has led to worse results and focus on the soft forward pass for this paper.

4.2. Inference

As previously stated, the inference structure is critical for performance. In particular, we require the reconstruction loss gradient to flow through the generative transition model which is not naturally the case for these types of models. Without a properly structured inference scheme, only the KL divergence would guide the generative transition model. To achieve this, we formulate our inference scheme as a local optimization around the prior prediction where information from the observation only adjusts our prior prediction. Generally, we split the inference model into two parts: 1) the generative transition model and 2) encoding the current (and optionally future) observations. Both parts will independently predict a distribution over the next latent state which are then combined in a manner inspired by a Bayesian update. In the following, we will go through the specific construction for both normal and Concrete distributions. The overall inference structure is depicted in figure 2b.

4.2.1. STRUCTURED INFERENCE OF GAUSSIAN LATENT STATE

Starting from the factorization of our true posterior, our approximate posterior takes the form $q_\phi(z_t \mid z_{t-1}, s_t, x_{\geq t}, u_{\geq t-1})$ where we notice that observations up to the last time step can be omitted as they are summarized by the last Markovian latent state z_{t-1}. As mentioned, the inference model splits into two parts: 1) transition model $q_{\text{trans}}(z_t \mid z_{t-1}, s_t, u_{t-1})$ and 2) inverse measurement model $q_{\text{meas}}(z_t \mid x_{\geq t}, u_{\geq t})$ as previously proposed in Karl et al. (2017b). This split allows us to reuse our generative transition model in place of $q_{\text{trans}}(z_t \mid z_{t-1}, s_t, u_{t-1})$. For practical reasons, we only share the computation of the transition mean μ_{trans} but not

the variance σ_{trans}^2 between inference and generative model. Both parts, q_{meas} and q_{trans}, will give us independent predictions about the new state z_t which will be combined in a manner akin to a Bayesian update in a Kalman Filter:

$$q_\phi(z_t \mid z_{t-1}, s_t, x_{\geq t}, u_{\geq t-1}) = \mathcal{N}(\mu_q, \sigma_q^2)$$
$$\propto q_{\text{meas}}(z_t \mid x_{\geq t}, u_{\geq t}) \times q_{\text{trans}}(z_t \mid z_{t-1}, s_t, u_{t-1})$$
$$q_{\text{meas}}(z_t \mid x_{\geq t}, u_{\geq t}) = \mathcal{N}(\mu_{\text{meas}}, \sigma_{\text{meas}}^2)$$
$$\text{where} \;\; [\mu_{\text{meas}}, \sigma_{\text{meas}}^2] = h_\phi(x_{\geq t}, u_{\geq t}) \tag{11}$$
$$q_{\text{trans}}(z_t \mid z_{t-1}, s_t, u_{t-1}) = \mathcal{N}(\mu_{\text{trans}}, \sigma_{\text{trans}}^2)$$
$$\text{where} \;\; \mu_{\text{trans}} = A_\theta(s_t) z_{t-1} + B_\theta(s_t) u_t, \sigma_{\text{trans}}^2 = Q_\phi(s_t)$$

The densities of q_{meas} and q_{trans} are multiplied resulting in another Gaussian density:

$$\mu_q = \frac{\mu_{\text{trans}} \sigma_{\text{meas}}^2 + \mu_{\text{meas}} \sigma_{\text{trans}}^2}{\sigma_{\text{meas}}^2 + \sigma_{\text{trans}}^2},$$
$$\sigma_q^2 = \frac{\sigma_{\text{meas}}^2 \sigma_{\text{trans}}^2}{\sigma_{\text{meas}}^2 + \sigma_{\text{trans}}^2}. \tag{12}$$

To enable online filtering, we can condition the inverse measurement model $q_{\text{meas}}(z_t \mid x_{\geq t}, u_{\geq t})$ solely on the current observation x_t instead of the entire remaining trajectory. We found empirically that despite being theoretically suboptimal, this can yield good results in many cases. Naturally this is the case for actually Markovian settings which are plentiful in the physical world. In this case, this methodology can be used for real-time state filtering in online feedback control.

For the initial state z_1 we do not have a conditional prior from the transition model as in the rest of the sequence. Other methods (Krishnan et al., 2015; Fraccaro et al., 2016) have used a standard normal prior, however this is not a good fit. We therefore decided that instead of predicting z_1 directly, we predict an auxiliary variable h that is then mapped deterministically to a starting state z_1. A standard Gaussian prior is then applied to h:

$$q_\phi(h \mid x_{1:T}, u_{1:T}) = \mathcal{N}(h; \mu_w, \sigma_w^2)$$
$$\text{where} \;\; [\mu_h, \sigma_h^2] = i_\phi(x_{1:T}, u_{1:T}) \tag{13}$$
$$z_1 = t_\phi(h)$$

Alternatively, we could specify a more complex or learned prior for the initial state like the VampPrior (Tomczak & Welling, 2018). Empirically, we were unable to produce good results with these approaches.

4.2.2. INFERENCE OF SWITCHING VARIABLES

Following Maddison et al. (2017) and Jang et al. (2017), we can reparameterize a discrete latent variable with the Gumbel-softmax trick. Again, we split our inference network $q_\phi(s_t \mid s_{t-1}, z_{t-1}, x_{\geq t}, u_{\geq t-1})$ in an identical fashion into two components: 1) transition model
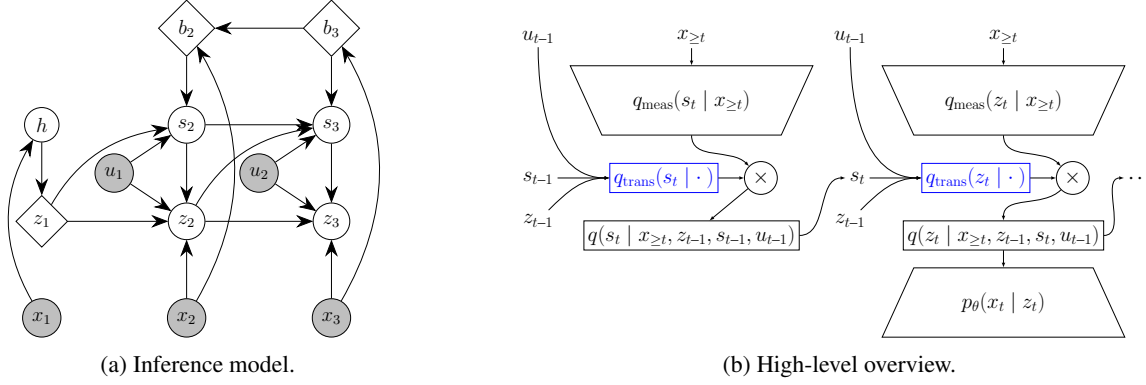
(a) Inference model.

(b) High-level overview.

Figure 2: (a) Depicts the inference model. b_t is the hidden state of the backward RNN of $q_{\text{meas}}(s_t \mid x_{\geq t}, u_{\geq t})$. Initial inference of h may be conditioned on the entire sequence of observations, or just a subsequence. We've omitted the arrows for sake of clarity for the rest of the graph. (b) Shows schematically how we combine the transition with the inverse measurement model in the inference network. Transitions (in blue) are shared with the generative model.

$q_{\text{trans}}(s_t \mid s_{t-1}, z_{t-1}, u_{t-1})$ and 2) inverse measurement model $q_{\text{meas}}(s_t \mid x_{\geq t}, u_{\geq t})$. The transition model is again shared with the generative model and is implemented via a neural network as we potentially require quick changes to chosen dynamics. The inverse measurement model is parametrized by a backward LSTM (or an MLP in the online filtering setting). For Concrete variables, we let each network predict the logits of a Concrete distribution and our inverse measurement model $q_\phi(s_t \mid x_{\geq t}, u_{\geq t})$ produces an additional vector γ, which determines the value of a gate deciding how the two predictions are to be weighted:

$$q_\phi(s_t \mid s_{t-1}, z_{t-1}, x_{\geq t}, u_{\geq t-1}) = \text{Concrete}(\alpha, \lambda_{\text{posterior}})$$
$$\text{with} \quad \alpha = \gamma \alpha_{\text{trans}} + (1 - \gamma) \alpha_{\text{meas}}$$
$$q_{\text{meas}}(s_t \mid x_{\geq t}, u_{\geq t}) = \text{Concrete}(\alpha_{\text{meas}}, \lambda_{\text{posterior}}) \quad (14)$$
$$\text{where} \quad [\alpha_{\text{meas}}, \gamma] = k_\phi(x_{\geq t}, u_{\geq t})$$
$$q_{\text{trans}}(s_t \mid s_{t-1}, z_{t-1}, u_{t-1}) = \text{Concrete}(\alpha_{\text{trans}}, \lambda_{\text{prior}})$$
$$\text{where} \quad \alpha_{\text{trans}} = g_\theta(z_{t-1}, s_{t-1}, u_{t-1})$$

The temperatures $\lambda_{\text{posterior}}$ and λ_{prior} are set as a hyperparameter and can be set differently for the prior and approximate posterior. The gating mechanism gives the model the option to balance between prior and approximate posterior. If the prior is good enough to explain the next observation, γ will be pushed to 1 which will ignore the measurement and will minimize the KL between prior and posterior by only propagating the prior. If the prior is not sufficient, information from the inverse measurement model can flow by decreasing γ and incurring a KL penalty.

4.2.3. REINTERPRETATION AS A HIERARCHICAL MODEL

Lastly, we could step away from the theory of SLDS and instead view our model simply as an hierarchical graphical model where some variables should explain the current observation while others should determine the transition over said variables. When viewed in this manner, it suggests to model the switching variables by any distribution, e.g. also as normally distributed with a specific decoder structure predicting some kind of transition parameters. If this worked better, it would highlight still existing optimization problems of discrete random variables. Additionally, it could support our initial claim that stochastic treatment of the transition dynamics in general is important, irrespective of the specific implementation. As such, it will act as an ablation study for our model. Although any number of parameterizations are now viable, the one that we explore here is to let our decoder to predict mixing coefficients α for our transition matrices. We suggest just using a single linear layer:

$$\alpha = \text{softmax}(Ws_t + b) \in \mathbb{R}^M. \quad (15)$$

In this scenario, our inference scheme for normally distributed switching variables is then identical to the one described in the previous section.
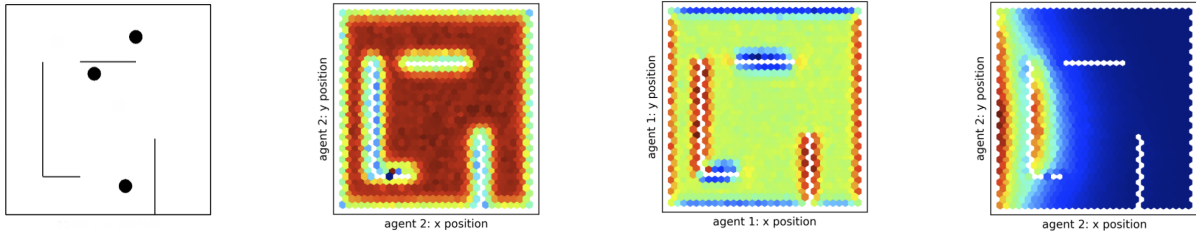
4.3. Training

Our objective function is the commonly used evidence lower bound for our hierarchical model:

$$\mathcal{L}_{\theta,\phi}(x_{1:T} \mid u_{1:T}) \geq$$
$$\mathbb{E}_{q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T})}[\log p_\theta(x_{1:T} \mid z_{1:T}, s_{2:T}, u_{1:T})] \quad (16)$$
$$- \text{KL}(q_\phi(z_{1:T}, s_{2:T} \mid x_{1:T}, u_{1:T}) \| p(z_{1:T}, s_{2:T} \mid u_{1:T})).$$

As we chose to factorize over time, the loss for a single observation x_t becomes:

$$\mathcal{L}_{\theta,\phi}(x_t \mid u_{1:T}) = \mathbb{E}_{s_t}[\mathbb{E}_{z_t}[\log p_\theta(x_t \mid z_t)]] \quad (17)$$
$$- \mathbb{E}_{s_{t-1}, z_{t-1}}[\text{KL}(q_\phi(s_t \mid \cdot) \| p_\theta(s_t \mid s_{t-1}, z_{t-1}, u_{t-1}))]$$
$$- \mathbb{E}_{z_{t-1}}[\mathbb{E}_{s_t}[\text{KL}(q_\phi(z_t \mid \cdot) \| p_\theta(z_t \mid z_{t-1}, s_t, u_{t-1}))]].$$

(a) Multi agent maze environment.

(b) Variable encoding free space for agent 2.

(c) Variable encoding walls for agent 1.

(d) System activation for deterministic transition.

Figure 3: Figures (b) and (c) depict an agent's position coloured by the average value of a single latent variable s marginalized over all controls and velocities. Figure (d) shows a typical activation map for a single transition system for deterministic treatment of transition dynamics. It does not generalize to the entire maze and stays fairly active near the wall.

The full derivation can be found in appendix A. We generally approximate the expectations with one sample by using the reparametrization trick, the exception being the KL between two Concrete random variables in which case we take 10 samples for the approximation. For the KL on the switching variables, we further introduce a scaling factor $\beta < 1$ (as first suggested in Higgins et al. (2016), although they suggested increasing the weighting of the KL term) to scale down its importance. This decision can be justified by other work which has demonstrated theoretically and empirically the inadequacy of maximum likelihood training on the ELBO for learning latent representations (Alemi et al., 2018). More details on the training procedure can be found in appendix C.

5. Experiments

In this section, we evaluate our approach on a diverse set of physics simulations based on partially observable system states or high-dimensional images as observations. We show that our model outperforms previous models and that our switching variables learn meaningful representations.

Models we compare to are Deep Variational Bayes Filter (DVBF) (Karl et al., 2017a), DVBF Fusion (Karl et al., 2017b) (called fusion as they do the same Gaussian multiplication in the inference network) which is closest to our model but does not have a stochastic treatment of the transition, the Kalman VAE (KVAE) (Fraccaro et al., 2017) and a vanilla LSTM (Hochreiter & Schmidhuber, 1997).

5.1. Multiple Bouncing Balls in a Maze

Our first experiment is a custom 3-agent maze environment simulated with Box2D. Each agent is fully described by its (x, y)-coordinates and its current velocity and may accelerate in either direction. We learn in a partially observable setting and limit the observations to the agents' positions, therefore $x \in \mathbb{R}^6$ while the true state space is in $\mathbb{R}^{12}$ and $u \in \mathbb{R}^6$.
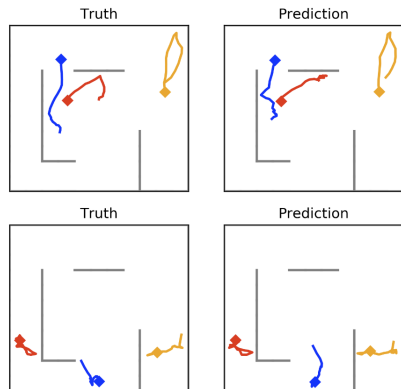

Figure 4: Comparison of actual and predicted 20 step trajectories. The diamond marker denotes the starting position of a trajectory. These results have been produced with Concrete switching variables.

Our first objective is to evaluate the learned latent space. We start by training a linear regression model on the latent space z to see if we have recovered a linear encoding of the unobserved velocities. Here, we achieve an R2 score of 0.92 averaged over all agents and velocity directions.

We now shift our focus to the switching variables that we anticipated to encode interactions with walls. We provide a qualitative confirmation of that in figure 3 where we see switching variables encoding space where there is no interaction in the next time step and variables which encode walls, distinguishing between vertical and horizontal ones. In figure 3d one can see that if the choice of locally linear transition is treated deterministically, we do not learn global features of the same kind. To confirm our visual inspection, we train a simple decision tree based on latent space s in order to predict an interaction with a wall. Here, we achieve an F1 score of 0.46. It is difficult to say what a good value should look like as collisions with low velocity are virtually indistinguishable from no collision, but certainly a signif-

Table 1: Mean squared error (MSE) on predicting future observations. Static refers to constantly predicting the first observation of the sequence.

	REACHER			3-BALL MAZE		
PREDICTION STEPS	1	5	10	1	5	10
STATIC	5.80E-02	5.36E-01	1.25E+00	1.40E-02	5.74E-01	2.65E+00
LSTM	3.07E-02	3.67E-01	1.02E+00	7.20E-03	1.58E-01	2.60E-01
DVBF	1.10E-02	3.06E-01	6.05E-01	6.20E-03	1.36E-01	1.82E-01
DVBF FUSION	4.90E-03	2.97E-02	8.25E-02	4.33E-03	2.03E-02	4.88E-02
OURS (CONCRETE)	1.06E-02	5.73E-02	1.56E-01	2.28E-03	1.22E-02	3.40E-02
OURS (NORMAL)	**3.39E-03**	**1.85E-02**	**4.97E-02**	**1.30E-03**	**5.52E-03**	**1.38E-02**

icant portion of collisions has been successfully captured. We were unable to capture collisions between two agents which can be explained by their rare occurrence and much more complicated ensuing dynamics.

Going over to quantitative evaluation, we compare our prediction quality to several other methods in table 1 where we outperform all of our chosen baselines. Also, curiously, modelling switching variables by a Gaussian distribution outperforms the Concrete distribution in all of our experiments. Aside from known practical issues with training a discrete variable via backpropagation, we explore one reason why that may be in section 5.5, which is the greater susceptibility to the chosen scale of temporal discretization.

5.2. Reacher

We then evaluate our model on the RoboschoolReacher (OpenAI, 2017) environment. Again, we learn only on partial observations, removing velocities and leaving us with just the positions or angles of the joints as observations. The reacher's dynamics are globally the same unless the upper and lower joints collide which is again a feature we expect our switching variables to detect. Similar to before, we inspect the learned latent space in figure 5 visually where show linear encoding of shoulder joint's velocity and encoding of collision dynamics. The chosen dynamics are unaffected by the shoulder angle, but are sensitive to the relative angle of the elbow to the upper link. To confirm our qualitative analysis, we again learn a linear classifier based on latent space s and reach an F1 score of 0.53. The predictive quality of our model is compared to other methods in table 1.

5.3. Ball in a Box on Image Data

Here, we evaluate our method on high-dimensional image observations using the single bouncing ball environment used by Fraccaro et al. (2017). They simulated 5000 sequences of 20 time steps each of a ball moving in a two-dimensional box, where each video frame is a 32×32 binary image. There are no forces applied to the ball, except for
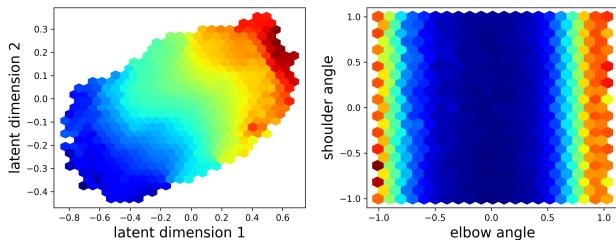
(a) Encoding shoulder velocity.   (b) Encoding joint collisions.

Figure 5: Encoding features of the reacher environment. Figure (a) shows two latent dimensions colored by the ground truth shoulder velocity. The model captures the shoulder's velocity purely out of provided joint angle data. Figure (b) highlights the activation of a Concrete switching variable. Note that the elbow's angle is provided relative to the upper link, meaning that a (normalized) value of -1 or 1 leads to a collision with the upper link.

the fully elastic collisions with the walls. Initial position and velocity are randomly sampled.

In figure 7a we compare our model to both the smoothed and generative version of the KVAE. The smoothed version receives the final state of the trajectory after the n predicted steps which is fed into the smoothing capability of the KVAE. One can see that our model learns a better transition model, even outperforming the smoothed KVAE for longer sequences. For short sequences, KVAE performs better which highlights the value of it disentangling the latent space into separate object and dynamics representation. A sample trajectory is plotted in figure 6.

5.4. FitzHugh-Nagumo

To compare to recurrent SLDS (rSLDS, Linderman et al. (2017)) and tree-structured SLDS (TrSLDS, Nassar et al. (2019)), we adopt their FitzHugh-Nagumo (FHN, FitzHugh (1961)) experimental setup. FHN is a 2-dimensional relaxation oscillator commonly used throughout neuroscience to describes a prototype of an excitable system (e.g., a neuron).
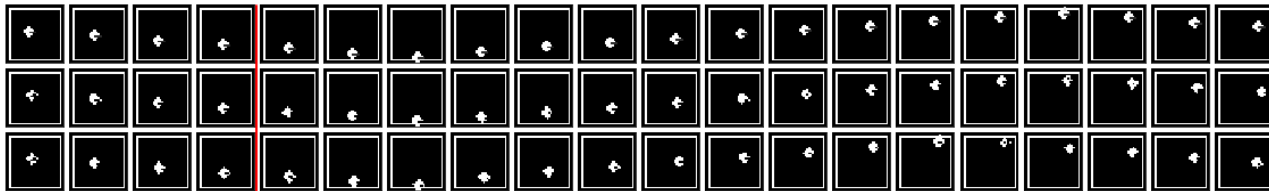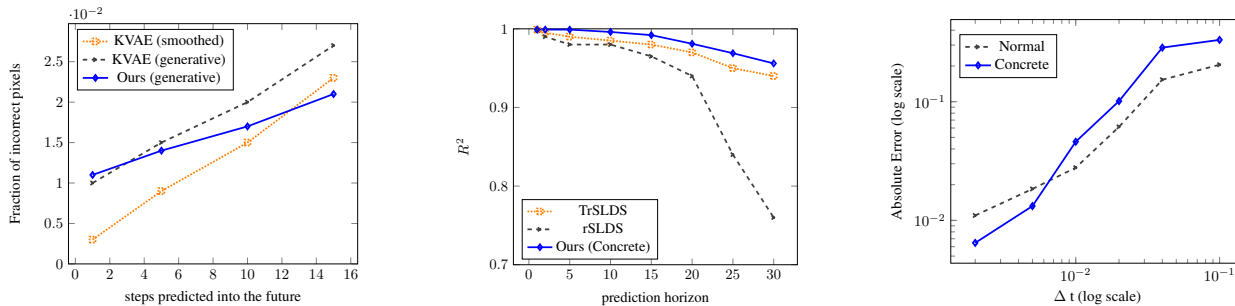
Figure 6: First row: data, second row: filtered reconstructions, third row: predictions. The first 4 steps are used to find a stable starting state, predictions start with step 5 (after the red line).

(a) Fraction of incorrectly predicted pixels. (b) FitzHugh-Nagumo multi-step prediction. (c) Susceptibility to discretization scale.

Figure 7: (a) Our dynamics model is outperforming even the smoothed KVAE for longer trajectories. (b) Our models performs as well as TrSLDS. (c) Modelling switching variables as Concrete random variables scales less favourably with increasing time discretization intervals.

It is fully described by the following system of differential equations:

$$\dot{v} = v - \frac{v^3}{3} - w + I_{\text{ext}}, \quad \tau\dot{w} = v + a - bw \quad (18)$$

Following their setup, we set the parameters to $a = 0.7$, $b = 0.8$, $\tau = 12.5$ and external stimulus $I_{\text{ext}} \sim \mathcal{N}(0.7, 0.04)$. We create 100 trajectories of length 430 where the last 30 time steps are withheld during training and used for evaluation. Starting states are drawn uniformly from $[-3; 3]^2$. In figure 7b, we compare the models based on normalized multi-step predictive performance where our model matches the performance of TrSLDS.

5.5. Susceptibility to the Scale of Temporal Discretization

In this section, we would like to explore how the choice of Δt when discretizing a system influences our results. This is a crucial factor often ignored or presumed to be chosen appropriately, although there has been some recent work addressing this issue specifically (Jayaraman et al., 2019; Neitz et al., 2018).

In particular, we suspect our model with discrete (Concrete) switching latent variables to be more susceptible to scaling than when modelled by a normal distribution. This is because in the latter case the switching variables can scale the

various matrices more freely, while in the former scaling up one system necessitates scaling down another. For empirical comparison, we go back to our custom maze environment (this time with only one agent as this is not pertinent to our question at hand) and learn the dynamics on various discretization scales. Then we compare the absolute error's growth for both approaches in figure 7c which supports our hypothesis. While the discrete approximation even outperforms for small Δt, there is a point where it rapidly becomes worse and gets overtaken by the normally distributed approximation. This suggests that Δt was simply chosen to be too large in both the reacher and the ball in a box with image observations experiment.

6. Conclusion

We have shown that our construction of using switching variables encourages learning a richer and more interpretable latent space. In turn, the richer representation led to an improvement of simulation accuracy in various tasks. In the future, we would like to look at other ways to approximate the discrete switching variables and exploit this approach for model-based control on real hardware systems. Furthermore, addressing the open problem of disentangling latent spaces is essential to fitting simple dynamics and would lead to significant improvements of this approach.

References

Ackerson, G. and Fu, K. On state estimation in switching environments. *IEEE Transactions on Automatic Control*, 15(1):10–17, 1970.

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A., and Murphy, K. Fixing a broken elbo. In *International Conference on Machine Learning*, pp. 159–168, 2018.

Barber, D. *Bayesian Reasoning and Machine Learning*, pp. 547–567. Cambridge University Press, 2012.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pp. 4502–4510, 2016.

Bayer, J. and Osendorfer, C. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Chang, C.-B. and Athans, M. State estimation for discrete systems with switching parameters. *IEEE Transactions on Aerospace and Electronic Systems*, (3):418–425, 1978.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., and Bengio, Y. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pp. 2980–2988, 2015.

Dai, B., Wang, Y., Aston, J., Hua, G., and Wipf, D. Hidden talents of the variational autoencoder. *arXiv preprint arXiv:1706.05148*, 2017.

Doerr, A., Daniel, C., Schiegg, M., Nguyen-Tuong, D., Schaal, S., Toussaint, M., and Trimpe, S. Probabilistic recurrent state-space models. *arXiv preprint arXiv:1801.10395*, 2018.

Eleftheriadis, S., Nicholson, T., Deisenroth, M., and Hensman, J. Identification of gaussian process state space models. In *Advances in Neural Information Processing Systems*, pp. 5309–5319, 2017.

FitzHugh, R. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1 (6):445–466, 1961.

Fraccaro, M., Sønderby, S. K., Paquet, U., and Winther, O. Sequential neural models with stochastic layers. In *Advances in neural information processing systems*, pp. 2199–2207, 2016.

Fraccaro, M., Kamronn, S., Paquet, U., and Winther, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pp. 3604–3613, 2017.

Goyal, A., Sordoni, A., Côté, M.-A., Ke, N., and Bengio, Y. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, pp. 6713–6723, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jang, E., Gu, S., and Poole, B. Categorical Reparameterization with Gumbel-Softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Jayaraman, D., Ebert, F., Efros, A. A., and Levine, S. Time-agnostic prediction: Predicting predictable video frames. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pp. 2946–2954, 2016.

Karl, M., Soelch, M., Bayer, J., and van der Smagt, P. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017a.

Karl, M., Soelch, M., Becker-Ehmck, P., Benbouzid, D., van der Smagt, P., and Bayer, J. Unsupervised real-time control through variational empowerment. *arXiv preprint arXiv:1710.05101*, 2017b.

Kingma, D. and Welling, M. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pp. 3581–3589, 2014.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Krishnan, R. G., Shalit, U., and Sontag, D. Deep Kalman Filters. *arXiv preprint arXiv:1511.05121*, 2015.

Krishnan, R. G., Shalit, U., and Sontag, D. Structured inference networks for nonlinear state space models. In *AAAI*, pp. 2101–2109, 2017.

Linderman, S., Johnson, M., Miller, A., Adams, R., Blei, D., and Paninski, L. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 914–922, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

Maddison, C. J., Mnih, A., and Teh, Y. W. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Marino, J., Cvitkovic, M., and Yue, Y. A general method for amortizing variational filtering. In *Advances in Neural Information Processing Systems*, pp. 7868–7879, 2018.

Nassar, J., Linderman, S., Bugallo, M., and Park, I. M. Tree-structured recurrent switching linear dynamical systems for multi-scale modeling. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Neitz, A., Parascandolo, G., Bauer, S., and Schölkopf, B. Adaptive skip intervals: Temporal abstraction for recurrent dynamical models. In *Advances in Neural Information Processing Systems*, pp. 9838–9848, 2018.

OpenAI. Roboschool: open-source software for robot simulation. 2017. URL `https://github.com/openai/roboschool`.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1278–II–1286. JMLR.org, 2014.

Shabanian, S., Arpit, D., Trischler, A., and Bengio, Y. Variational bi-lstms. *arXiv preprint arXiv:1711.05717*, 2017.

Tomczak, J. and Welling, M. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223, 2018.

van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

Watter, M., Springenberg, J., Boedecker, J., and Riedmiller, M. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pp. 2746–2754, 2015.