Structured Inference Networks for Nonlinear State Space Models

Rahul G. Krishnan, Uri Shalit, David Sontag
New York University
{rahul, shalit, dsontag}@cs.nyu.edu

Abstract

Gaussian state space models have been used for decades as generative models of sequential data. They admit an intuitive probabilistic interpretation, have a simple functional form, and enjoy widespread adoption. We introduce a unified algorithm to efficiently learn a broad class of linear and non-linear state space models, including variants where the emission and transition distributions are modeled by deep neural networks. Our learning algorithm simultaneously learns a compiled inference network and the generative model, leveraging a structured variational approximation parameterized by recurrent neural networks to mimic the posterior distribution. We apply the learning algorithm to both synthetic and real-world datasets, demonstrating its scalability and versatility. We find that using the structured approximation to the posterior results in models with significantly higher held-out likelihood.

1. Introduction

Models of sequence data such as hidden Markov models (HMMs) and recurrent neural networks (RNNs) are widely used in machine translation, speech recognition, and computational biology. Linear and non-linear Gaussian state space models (GSSMs, Fig. 1) are used in applications including robotic planning and missile tracking. However, despite huge progress over the last decade, efficient learning of non-linear models from complex high dimensional time-series remains a major challenge. Our paper proposes a unified learning algorithm for a broad class of GSSMs, and we introduce an inference procedure that scales easily to high dimensional data, compiling approximate (and where feasible, exact) inference into the parameters of a neural network.

In engineering and control, the parametric form of the GSSM model is often known, with typically a few specific parameters that need to be fit to data. The most commonly used approaches for these types of learning and inference problems are often computationally demanding, e.g. dual extended Kalman filter (Wan and Nelson, 1996), expectation maximization (Briegel and Tresp, 1999; Roweis and Ghahramani, 2000) or particle filters (Schön et al., 2011). Our compiled inference algorithm can easily deal with high-dimensions both in the observed and the latent spaces, without compromising the quality of inference and learning.

When the parametric form of the model is unknown, we propose learning *deep Markov models* (DMM), a class of generative models where classic linear emission and transition distributions are replaced with complex multi-layer perceptrons (MLPs). These are GSSMs that retain the Markovian

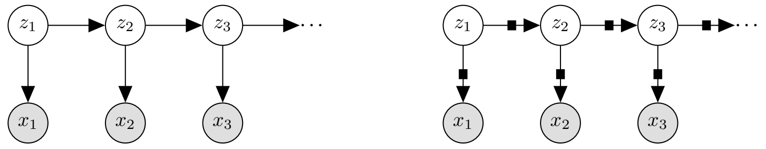


Figure 1: Generative Models of Sequential Data: (**Left**) Hidden Markov Model (HMM), (**Right**) Deep Markov Model (DMM) ■ denotes the neural networks used in DMMs for the emission and transition functions. The code correspond to this work to learn models of the form shown in the figure on the right may be found at: `https://github.com/clinicalml/structuredinference`

structure of HMMs, but leverage the representational power of deep neural networks to model complex high dimensional data. If one augments a DMM model such as the one presented in Fig. 1 with edges from the observations x_t to the latent states of the following time step z_{t+1}, then the DMM can be seen to be similar to, though more restrictive than, stochastic RNNs (Bayer and Osendorfer, 2014) and variational RNNs (Chung et al., 2015).

Our learning algorithm performs stochastic gradient ascent on a variational lower bound of the likelihood. Instead of introducing variational parameters for each data point, we *compile* the inference procedure at the same time as learning the generative model. This idea was originally used in the wake-sleep algorithm for unsupervised learning (Hinton et al., 1995), and has since led to state-of-the-art results for unsupervised learning of deep generative models (Kingma and Welling, 2014; Mnih and Gregor, 2014; Rezende et al., 2014).

Specifically, we introduce a new family of *structured inference networks*, parameterized by recurrent neural networks, and evaluate their effectiveness in three scenarios: (1) when the generative model is known and fixed, (2) in parameter estimation when the functional form of the model is known and (3) for learning deep Markov models. By looking at the structure of the true posterior, we show both theoretically and empirically that inference for a latent state should be performed using information *from its future*, as opposed to recent work which performed inference using only information from the past (Chung et al., 2015; Gan et al., 2015; Gregor et al., 2015). We also outperform mean-field based approaches. Finally, we learn a DMM on a polyphonic music dataset and on a dataset of electronic health records (a complex high dimensional setting with missing data). We use the model learned on health records to ask queries such as "what would have happened to patients had they not received treatment", and show that our model correctly identifies the way certain medications affect a patient's health.

Related Work: Learning GSSMs with MLPs for the transition distribution was considered by (Raiko and Tornio, 2009). They approximate the posterior with non-linear dynamic factor analysis (Valpola and Karhunen, 2002), which scales quadratically with the observed dimension and is impractical for large-scale learning.

Recent work has also considered variational learning of time-series data using inference or recognition networks. Archer et al. (2015) propose using a block-diagonal Gaussian approximation to the posterior, parameterized by neural networks. Johnson et al. (2016) consider learning structured time-series models by approximating the posterior distribution with conditional random fields. Bayer and Osendorfer (2014) and Fabius and van Amersfoort (2014) create a stochastic variant of RNNs by making the hidden state of the RNN at every time step be a function of independently sampled latent variables. Chung et al. (2015) apply a similar model to speech data, sharing parameters between the RNNs for the generative model and the inference network. Gan et al. (2015) learn a model with discrete random variables, using a structured inference network that only considers information from the past, similar to Chung et al. (2015) and Gregor et al. (2015). In contrast to these works, we use information from the future within a structured inference network, which we show to be preferable both theoretically and practically. Additionally, we systematically evaluate the impact of the different variational approximations on learning. Watter et al. (2015) construct a first-order Markov model using inference networks. However, their learning algorithm is based on data tuples over consecutive time steps. This makes the strong assumption that the posterior distribution can be recovered based on observations at the current and next time-step. As we show, for generative models like the one in Fig. 1, the posterior distribution at any time step is a function of *all* future observations.

This paper expands an earlier work (Krishnan et al. 2015). The current paper instantiates the inference scheme presented in (Krishnan et al., 2015) [Thm. 5.1] with a specific neural architecture, evalutes several experimental benchmarks, and explores in depth the impact of inference on learning. We also revamped the names of the underlying inference scheme, based on suggestions made by reviewers.

2. Background

Gaussian State Space Models: We consider both inference and learning in a class of latent variable models given by:

$$z_t \sim \mathcal{N}(G_\alpha(z_{t-1}, \Delta_t), S_\beta(z_{t-1}, \Delta_t)) \textit{ (Transition)} \qquad x_t \sim \Pi(F_\kappa(z_t)) \textit{ (Emission)} \tag{1}$$

We assume that the distribution of the latent states is a multivariate Gaussian with a mean and covariance which are differentiable functions of the previous latent state. The multivariate observations x_t are distributed according to a distribution Π (e.g., independent Bernoullis if the data is binary) whose parameters are a function of the corresponding latent state z_t. Collectively, we denote by $\theta = \{\alpha, \beta, \kappa\}$ the parameters of the generative model. Eq. 1 subsumes a large family of linear and non-linear Gaussian state space models. For example, by setting $G_\alpha(z_{t-1}) = G_t z_{t-1}, S_\beta = \Sigma_t, F_\kappa = F_t z_t$ we obtain linear state space models. The functional forms and initial parameters for $G_\alpha, S_\beta, F_\kappa$ may be pre-specified.

Variational Learning: Using recent advances in variational inference we optimize a variational lower bound on the data log-likelihood. The key technical innovation is the introduction of an *inference network* or *recognition network* (Hinton et al., 1995), a neural network which approximates the intractable posterior. This is a parametric conditional distribution that is optimized to perform inference. Throughout this paper we will use θ to denote the parameters of the generative model, and ϕ to denote the parameters of the inference network.

Let $p(x, z) = p_\theta(z)p_\theta(x|z)$ be a generative model for the set of observations x. The posterior distribution $p_\theta(z|x)$ is typically intractable. Using the well-known variational principle, we posit an approximate posterior distribution $q_\phi(z|x)$ to obtain the following lower bound on the marginal likelihood:

$$\log p_\theta(x) \geq \mathop{\mathbb{E}}_{q_\phi(z|x)} [\log p_\theta(x|z)] - \mathrm{KL}(\, q_\phi(z|x) || p_\theta(z)\,), \tag{2}$$

where the inequality is by Jensen's inequality. Kingma and Welling (2014); Rezende et al. (2014) use a neural net (with parameters ϕ) to parameterize q_ϕ. The challenge in the resulting optimization problem is that the lower bound (2) includes an expectation w.r.t. q_ϕ, which implicitly depends on the network parameters ϕ. This difficulty is overcome by using *stochastic backpropagation*. We use a Normally distributed variational approximation: ie. $q_\phi(z|x) \sim \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))$, where $\mu_\phi(x), \Sigma_\phi(x)$ are functions of the observation x which parameterize the Normal distribution. Using this assumption, a simple transformation allows one to obtain unbiased Monte Carlo estimates of the gradients of $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$ with respect to ϕ. The KL term in (2) can be estimated similarly since it is also an expectation. If we assume the prior $p_\theta(z)$ is also Normally distributed, the KL and its gradients may be obtained analytically.

3. A Factorized Variational Lower Bound

We leverage stochastic back-propagation to learn generative models given by Eq. 1, corresponding to the graphical model in Figure 1. Our insight is that for the purpose of inference, we can use the Markov properties of the generative model to guide us in deriving a structured approximation to the posterior. Specifically, the posterior factorizes as:

$$p(\vec{z}|\vec{x}) = p(z_1|\vec{x}) \prod_{t=2}^{T} p(z_t|z_{t-1}, x_t, \ldots, x_T). \tag{3}$$

This factorization follows from conditional independencies in the model and is detailed in Appendix C. We directly mimic the structure of the posterior with the following factorization of the variational approximation:

$$\begin{aligned} & q_\phi(\vec{z}|\vec{x}) = \prod_{t=1}^{T} q_\phi(z_t|z_{t-1}, x_t, \ldots, x_T) \\ & \text{s.t.} \quad q_\phi(z_t|z_{t-1}, x_t, \ldots, x_T) \sim \mathcal{N}\left(\mu_\phi(z_{t-1}, x_t, \ldots, x_T), \Sigma_\phi(z_{t-1}, x_t, \ldots, x_T)\right) \end{aligned} \tag{4}$$

where μ_ϕ and Σ_ϕ are functions parameterized by neural nets. Although q_ϕ has the option to condition on all information across time, Eq. 3 suggests that in fact it suffices to condition on information from the future and the previous latent state. The previous latent state serves as a summary statistic for information from the past.

Deriving a Variational Lower Bound: For a generative model (with parameters θ) and an inference network (with parameters ϕ), we are interested in $\max_\theta \log p_\theta(\vec{x})$. For ease of exposition, we instantiate the derivation of the variational bound for a single data point $\vec{x}$ though we learn θ, ϕ from a dataset.

The lower bound in Eq. 2 has an analytic form of the KL term only for the simplest of transition models G_α, S_β between z_{t-1} and z_t (Eq. 1). One could estimate the gradient of the KL term by sampling from the variational model, but that results in high variance estimates and gradients. We use a different factorization of the KL term, leading to the variational lower bound we use as our objective function:

$$\begin{aligned} \mathcal{L}(\vec{x}; (\theta, \phi)) = & \sum_{t=1}^{T} \underset{q_\phi(z_t|\vec{x})}{\mathbb{E}} [\log p_\theta(x_t|z_t)] - \text{KL}(q_\phi(z_1|\vec{x}) || p_\theta(z_1)) \\ & - \sum_{t=2}^{T} \underset{q_\phi(z_{t-1}|\vec{x})}{\mathbb{E}} \left[\text{KL}(q_\phi(z_t|z_{t-1}, \vec{x}) || p_\theta(z_t|z_{t-1}))\right]. \end{aligned} \tag{5}$$

The derivation of the bound and the factorization of the KL divergence is detailed in Appendix A,B. The key point is the resulting objective function has stable analytic gradients, since the KL divergence does not need to be approximated with Monte-Carlo estimation.

Learning with Gradient Descent: The objective in Eq. 6 is differentiable in the parameters of the model (θ, ϕ). If the generative model θ is fixed, we perform gradient ascent of (6) in ϕ. Otherwise, we perform gradient ascent in both ϕ and θ. We use stochastic backpropagation (Kingma and Welling, 2014; Rezende et al., 2014) for estimating the gradient w.r.t. ϕ. Note that the expectations are only taken with respect to the variables z_{t-1}, z_t, which are the sufficient statistics of the Markov model. This is in contrast to the variational bound obtained by Chung et al. (2015) in Eq. 7 of their paper, where expectations are taken over all past latent states. For the KL terms in Eq. 6, we use the fact that the prior $p_\theta(z_t|z_{t-1})$ and the variational approximation to the posterior $q_\phi(z_t|z_{t-1}, \vec{x})$ are both normally distributed, and hence their KL divergence may be estimated analytically (see Appendix B).

4. Structured Inference Networks

We now detail the way we construct the variational approximation q_ϕ, and specifically how we model the mean and diagonal covariance functions μ_ϕ and Σ_ϕ using recurrent neural networks (RNNs). This parameterization cannot in general be expected to be equal to $p_\theta(z|x)$, but in many cases is a reasonable approximation. We use RNNs due to their high model capacity and ability to scale well to large datasets.

Table 1 details the different choices for inference networks that we evaluate. The Deep Kalman Smoother, **DKS**, corresponds exactly to the functional form suggested by Eq. 3, and is our proposed variational approximation. The **DKS** smoothes information from the past (z_t) and future ($x_t, \dots x_T$) to form the approximate posterior.

We also evaluate other possibilities for the variational models (inference networks) q_ϕ: two are mean-field models (denoted **MF**) and two are structured models (denoted **ST**). They are distinguished by whether they use information from the past (denoted **L**, for left), the future (denoted **R**, for right), or both (denoted **LR**). See Figure 2 for an illustration of two of these methods. Each conditions on a different subset of the observations to summarize information in the input sequence $\vec{x}$. **DKS** corresponds to **ST-R**.

The hidden states of the RNN are used to parameterize the parameters of the variational distribution, which go through what we call the "combiner function". We obtain the mean μ_q and diagonal covariance σ_q^2 for the posterior at each time-step in a manner akin to Gaussian belief propagation. Specifically, we interpret the hidden states of the forward and backward RNNs as parameterizing the mean and variance of two Gaussian-distributed "messages" summarizing the observations from the past and the future, respectively. We then multiply these two Gaussians, performing a variance-weighted average of the means. All operations should be understood to be performed element-wise on the corresponding vectors.

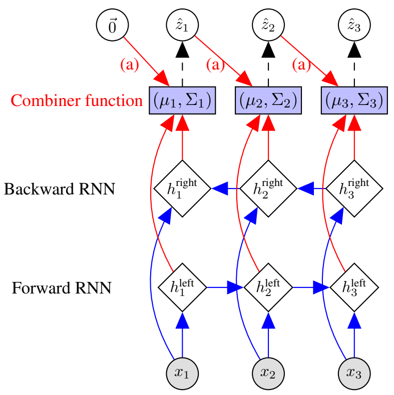
Combiner Function for Mean Field Approximations: For the **MF-LR** inference network, the mean μ_t and covariance Σ_t of the variational distribution $q_\phi(z_t|\vec{x})$ are predicted using the output of the RNN (not conditioned on z_{t-1}) as follows, where $\text{Softplus}(x) = \log(1 + \exp(x))$:

$$\mu_{\text{r}} = W_{\mu_{\text{r}}}^{\text{right}} h_t^{\text{right}} + b_{\mu_{\text{r}}}^{\text{right}}; \quad \sigma_{\text{r}}^2 = \text{Softplus}(W_{\sigma_{\text{r}}^2}^{\text{right}} h_t^{\text{right}} + b_{\sigma_{\text{r}}^2}^{\text{right}})$$
$$\mu_{\text{l}} = W_{\mu_{\text{l}}}^{\text{left}} h_t^{\text{left}} + b_{\mu_{\text{l}}}^{\text{left}}; \quad \sigma_{\text{l}}^2 = \text{Softplus}(W_{\sigma_{\text{l}}^2}^{\text{left}} h_t^{\text{left}} + b_{\sigma_{\text{l}}^2}^{\text{left}})$$
$$\mu_q = \frac{\mu_{\text{r}}\sigma_{\text{l}}^2 + \mu_{\text{l}}\sigma_{\text{r}}^2}{\sigma_{\text{r}}^2 + \sigma_{\text{l}}^2}; \quad \sigma_q^2 = \frac{\sigma_{\text{r}}^2\sigma_{\text{l}}^2}{\sigma_{\text{r}}^2 + \sigma_{\text{l}}^2}$$

Combiner Function for Structured Approximations: Since the posterior $p_\theta(z_t \mid z_{t-1}, \vec{x}) \propto p_\theta(z_t \mid x_1, \ldots, x_{t-1}, z_{t-1}) p_\theta(x_t, \ldots, x_T \mid z_t)$, a similar reasoning as above suggests that we should use a weighted sum of information from the left and right. The combiner functions for the structured approximations are implemented as:

$$\textit{(For } \textbf{ST-LR}\textit{)} \quad h_{\text{combined}} = \frac{1}{3}(\text{Tanh}(W z_{t-1} + b) + h_t^{\text{left}} + h_t^{\text{right}}),$$
$$\textit{(For } \textbf{DKS}\textit{)} \quad h_{\text{combined}} = \frac{1}{2}(\text{Tanh}(W z_{t-1} + b) + h_t^{\text{right}}),$$
$$\mu_q = W_{\mu_q} h_{\text{combined}} + b_{\mu_q} \quad \textit{(Posterior Means and Covariances)}$$
$$\sigma_q^2 = \text{Softplus}(W_{\sigma_q^2} h_{\text{combined}} + b_{\sigma_q^2})$$

The combiner function uses the Tanh non-linearity from z_{t-1} to approximate the transition function (alternatively, one could share parameters with the generative model), and here we use a simple weighting between the components.



Inf. Network	Variational Approximation	Implementation
MF-LR	$q(z_t \mid x_1, \ldots x_T)$	BRNN
MF-L	$q(z_t \mid x_1, \ldots x_t)$	RNN
ST-L	$q(z_t \mid z_{t-1}, x_1, \ldots x_t)$	RNN & comb.fxn
DKS	$q(z_t \mid z_{t-1}, x_t, \ldots x_T)$	RNN & comb.fxn
ST-LR	$q(z_t \mid z_{t-1}, x_1, \ldots x_T)$	BRNN & comb.fxn

Figure 2 & Table 1: Structured Inference Networks: **MF-LR** and **ST-LR** variational approximations for a sequence of length 3, using a bi-directional recurrent neural net (BRNN). The BRNN takes as input the sequence $(x_1, \ldots x_3)$, and through a series of non-linearities denoted by the blue arrows it forms a sequence of hidden states summarizing information from the left and right (h_t^{left} and h_t^{right}) respectively. Then through a further sequence of non-linearities which we call the "combiner function" (marked (a) above), and denoted by the red arrows, it outputs two vectors μ and Σ, parameterizing the mean and diagonal covariance of $q_\phi(z_t \mid z_{t-1}, \vec{x})$ of Eq. 4. Samples $\hat{z}_t$ are drawn from $q_\phi(z_t \mid z_{t-1}, \vec{x})$, as indicated by the black dashed arrows. For the structured variational models **ST-LR**, the samples $\hat{z}_t$ are fed into the computation of μ_{t+1} and Σ_{t+1}, as indicated by the red arrows with the label (a). The mean-field model does *not* have these arrows, and therefore computes $q_\phi(z_t \mid \vec{x})$. We use $\hat{z}_0 = \vec{0}$. **Table of Inference Networks:** BRNN refers to a Bidrectional RNN and comb.fxn is shorthand for combiner function.

5. Deep Markov Models

Following Raiko et al. (2006), we apply the ideas of deep learning to non-linear continuous state space models. Where the transition and emission function have an unknown functional form, we parameterize $G_\alpha, S_\beta, F_\kappa$ from Eq. 1 with deep neural networks. See Figure 1 (right) for an illustration of the graphical model.

Emission Function: We parameterize the emission function F_κ using a two-layer MLP (multi-layer perceptron), $\text{MLP}(x, \text{NL}_1, \text{NL}_2) = \text{NL}_2(W_2\text{NL}_1(W_1x + b_1) + b_2))$, where NL denotes non-linearities such as ReLU, sigmoid, or Tanh units applied element-wise to the input vector. For modeling binary data, $F_\kappa(z_t) = \text{Sigmoid}(W_{\text{emission}}\text{MLP}(z_t, \text{ReLU}, \text{ReLU}) + b_{\text{emission}})$ parameterizes the mean probabilities of independent Bernoullis.

Gated Transition Function: Instead of MLPs, we use a gated transition function inspired by Gated Recurrent Units (Chung et al., 2014). Gated recurrent units (GRUs) are a neural architecture that parameterizes the recurrence equation in the RNN with gating units to control the flow of information from one hidden state to the next, conditioned on the observation. Unlike GRUs, in the DMM, the transition function is not conditional on any of the observations. All the information must be encoded in the completely stochastic latent state. To achieve this goal, we create a Gated Transition Function. We would like the model to have the flexibility to choose a linear transition for some dimensions while having a non-linear transitions for the others. We adopt the following parameterization, where $\mathbb{I}$ denotes the identity function and $\odot$ denotes element-wise multiplication: Using the notation for MLP from the main paper. How G_α and S_β are defined as a function of z_{t-1}.

$$
\begin{aligned}
&g_t = \text{MLP}(z_{t-1}, \text{ReLU}, \text{Sigmoid}) \quad \textit{(Gating Unit)}\\
&h_t = \text{MLP}(z_{t-1}, \text{ReLU}, \mathbb{I}) \quad \textit{(Proposed mean)}\\
&\textit{(Transition Mean } G_\alpha \textit{ and } S_\beta\textit{)}\\
&\mu_t(z_{t-1}) = (1 - g_t) \odot (W_{\mu_p} z_{t-1} + b_{\mu_p}) + g_t \odot h_t\\
&\sigma_t^2(z_{t-1}) = \text{Softplus}(W_{\sigma_p^2}\text{ReLU}(h_t) + b_{\sigma_p^2})
\end{aligned}
$$

Note that the architecture shares the bottom layer of the mean and covariance functions. In our experiments, we initialize W_{μ_p} to be the identity function and b_{μ_p} to 0. The parameters of the emission and transition function form the set θ.

6. Evaluation

Our models and learning algorithm are implemented in Theano (Theano Development Team, 2016). We use Adam (Kingma and Ba, 2015) with a learning rate of 0.0008 to train the DMM. Our code may be found at `https://github.com/clinicalml/structuredinference`.

6.1 Datasets

We evaluate on three datasets.

Synthetic: We consider simple linear and non-linear GSSMs. To train the inference networks we use $N = 5000$ datapoints of length $T = 25$. We consider both one and two dimensional systems for inference and parameter estimation. We compare our results using the training value of the variational bound $\mathcal{L}(\vec{x};(\theta,\phi))$ (Eq. 6) and the RMSE $= \sqrt{\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T}\sum_{t=1}^{T}[\mu_\phi(x_{i,t}) - z^*_{i,t}]^2}$, where z^* correspond to the true underlying zs that generated the data.

Polyphonic Music: We train DMMs on polyphonic music data (Boulanger-lewandowski et al., 2012). An instance in the sequence comprises an 88-dimensional binary vector corresponding to the notes of a piano. We learn for 2000 epochs and report results based on early stopping using the validation set. We report held-out negative log-likelihood (NLL) in the format "a (b) {c}". a is an importance sampling based estimate of the NLL (details in appendix); $b = \frac{1}{\sum_{i=1}^{N} T_i}\sum_{i=1}^{N} -\mathcal{L}(\vec{x};\theta,\phi)$ where T_i is the length of sequence i. This is an upper bound on the NLL, which facilitates comparison to RNNs; From inspecting the code for TSBN (Gan et al., 2015) we found that they report $c = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\mathcal{L}(\vec{x};\theta,\phi)$. We compute this to facilitate comparison with their work.

Electronic Health Records (EHRs): The dataset comprises 5000 diabetic patients using data from a major health insurance provider. The observations of interest are: A1c level (hemoglobin A1c, a protein for which a high level indicates that the patient is diabetic) and glucose (blood sugar). We bin glucose into quantiles and A1c into clinically meaningful bins. The observations also include age, gender and ICD-9 diagnosis codes for co-morbidities of diabetes such as congestive heart failure, chronic kidney disease and obesity. There are 48 binary observations for a patient at every time-step. We group each patient's data (over 4 years) into three month intervals, yielding a sequence of length 18.

6.2 Compiling Exact Inference

We investigate whether inference networks can accurately compile exact posterior inference into the network parameters ϕ for linear GSSMs when exact inference is feasible. For this experiment we optimize Eq. 6 over ϕ, while θ is fixed to a synthetic distribution given by a one-dimensional GSSM. We compare results obtained by the various approximations we propose to those obtained by Kalman smoothing (Duckworth, 2016) which performs *exact inference*. Fig. 4 depicts our results. The proposed **DKS** (i.e., **ST-R**) and **ST-LR** outperform the mean-field based variational method **MF-L** that only looks at information from the past. **MF-LR**, however, is often able to catch up when it comes to RMSE, highlighting the role that information from the future plays when performing posterior inference, as is evident in the posterior factorization (3). Both **DKS** and **ST-LR** converge to the RMSE of the exact Smoothed KF, and their bound on the likelihood becomes tight.

6.3 Inference for Parameter Estimation

On synthetic non-linear datasets (see Appendix E) we find, similarly, that the structured variational approximations are more capable of generalizing inference to unseen data while being able to match inference using a smoothed Unscented Kalman Filter (Wan et al., 2000). Finally, Fig. 3 illustrates a toy instance where we perform parameter estimation in a synthetic, two-dimensional, non-linear GSSM.

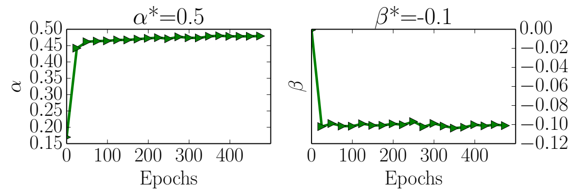


Figure 3: Learning parameters α, β in a synthetic non-linear GSSM. $N = 5000, T = 25$ $\vec{z}_t \sim \mathcal{N}([0.2z^0_{t-1} + \tanh(\alpha z^1_{t-1}); 0.2z^1_{t-1} + \sin(\beta z^0_{t-1})], 1.0)$ $\vec{x}_t \sim \mathcal{N}(0.5\vec{z}_t, 0.1)$ where $\vec{z}$ denotes a vector, [] denotes concatenation and superscript denotes indexing.

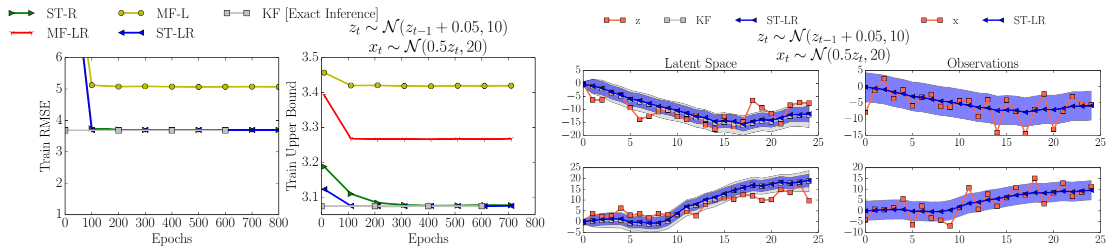


Figure 4: Synthetic Evaluation: Compiled inference for a *fixed* linear GSSM: $z_t \sim \mathcal{N}(z_{t-1} + 0.05, 10)$, $x_t \sim \mathcal{N}(0.5z_t, 20)$. The training set comprised $N = 5000$ one-dimensional observations of sequence length $T = 25$. **(Left)** RMSE with respect to true z that generated the data and variational bound during training. **(Right)** Visualizing inference in two sequences; z is the latent variable that generated the observation x. "Observations" denotes application of the emission function to the posterior shown in "Latent Space". Shading denotes standard deviations.

Table 2: Comparing Inference Networks: Test negative log-likelihood on polyphonic music of different inference networks trained on a DMM with a fixed structure (lower is better). The numbers inside parentheses are the variational bound.

Inference Network	JSB	Nottingham	Piano	Musedata
DKS (i.e., **ST-R**)	6.605 (7.033)	3.136 (3.327)	8.471 (8.584)	7.280 (7.136)
ST-L	7.020 (7.519)	3.446 (3.657)	9.375 (9.498)	8.301 (8.495)
ST-LR	6.632 (7.078)	3.251 (3.449)	8.406 (8.529)	7.127 (7.268)
MF-LR	6.701 (7.101)	3.273 (3.441)	9.188 (9.297)	8.760 (8.877)

6.4 Mean-Field vs Structured Inference

Table 2 shows the results of learning a DMM on the polyphonic music dataset using **MF-LR**, **ST-L**, **DKS** and **ST-LR**. **ST-L** is a structured variational approximation that only considers information from the past and, up to implementation details, is comparable to the one used by Gregor et al. (2015). Comparing the negative log-likelihoods of the learned models, we see that the looseness in the variational bound (which we first observed in the synthetic setting in Fig. 4 top right) significantly affects the ability to learn. **ST-LR** and **DKS** substantially outperform **MF-LR** and **ST-L**. This adds credence to the idea that by taking into consideration the factorization of the posterior, one can perform better inference and, consequently, learning, in real-world, high dimensional settings. Note that the **DKS** network has half the parameters of the **ST-LR** and **MF-LR** networks.

6.5 A Generalization of the DMM

To display the efficacy of our inference algorithm to model variants beyond first-order Markov Models, we further augment the DMM with edges from x_{t-1} to z_t and from x_{t-1} to x_t. We refer to the resulting generative model as DMM-Augmented (Aug.). Augmenting the DMM with additional edges realizes a richer class of generative models. The baselines we compare to in Table 3 also have more complex generative models than the DMM. STORN has edges from x_{t-1} to z_t given by the recurrence update and TSBN has edges from x_{t-1} to z_t as well as from x_{t-1} to x_t. HMSBN shares

Table 3: Evaluation against Baselines: Test negative log-likelihood (lower is better) on Polyphonic Music Generation dataset. **Table Legend**: RNN (Boulanger-lewandowski et al., 2012), LV-RNN (Gu et al., 2015), STORN (Bayer and Osendorfer, 2014), TSBN, HMSBN (Gan et al., 2015).

Methods	JSB	Nottingham	Piano	Musedata
DMM	6.388 (6.926) {6.856}	2.770 (2.964) {2.954}	7.835 (7.980) {8.246}	6.831 (6.989) {6.203}
DMM-Aug.	6.288 (6.773) {6.692}	2.679 (2.856) {2.872}	7.591 (7.721) {8.025}	6.356 (6.476) {5.766}
HMSBN	(8.0473) {7.9970}	(5.2354) {5.1231}	(9.563) {9.786}	(9.741) {8.9012}
STORN	6.91	2.85	7.13	6.16
RNN	8.71	4.46	8.37	8.13
TSBN	{7.48}	{3.67}	{7.98}	{6.81}
LV-RNN	3.99	2.72	7.61	6.89

the same structural properties as the DMM, but is learned using a simpler inference network. We show that **DKS** can be used *as is* for inference on a more complex generative model than DMM, while making gains in held-out likelihood. All following experiments use **DKS** for posterior inference.

In Table 3, as we increase the complexity of the generative model, we obtain better results across all datasets. The DMM outperforms both RNNs and HMSBN everywhere, outperforms STORN on JSB, Nottingham and outperform TSBN on all datasets except Piano. Compared to LV-RNN (that optimizes the inclusive KL-divergence), DMM-Aug obtains better results on all datasets except JSB. This showcases our flexible, structured inference network's ability to learn powerful generative models that compare favourably to other state of the art models. We provide audio files for samples from the learned DMM models in the code repository.

6.6 EHR Patient Data

Learning models from large observational health datasets is a promising approach to advancing precision medicine. Such models could be used, for example, to understand which medications work best, for whom. In this section, we show that the DMM, trained on EHR data using the **DKS** may be used for precisely such an application. We highlight some of the challenges we overcome to perform learning in this data:

- We make use of the time-varying drug prescription u_t for each patient. We augment the DMM's transition function as $z_t \sim \mathcal{N}(G_\alpha(z_{t-1}, u_{t-1}, \Delta_t), S_\beta(z_{t-1}, u_{t-1}, \Delta_t)$ (cf. (1)), where u_t is a binary indicator vector of eight diabetic drugs including Metformin and Insulin. Metformin is the most commonly prescribed first-line anti-diabetic drug.

- A subset of the observations (such as A1C and Glucose values) is frequently missing in the data. We marginalize them out during learning, which is straightforward within the probabilistic semantics of our Bayesian network.

- The choice of emission and transition function to use for such data is not well understood. In Figure 5 (right), we experiment with variants of DMMs and find that using MLPs (rather than linear functions) in the emission and transition function yield the best (in terms of held-out likelihood) generative models.

Modeling the effect of Diabetic Medications: Since our cohort comprises diabetic patients, we ask what *would* have happened to a patient had diabetic drugs not been prescribed? We perform inference using held-out patient data leading up to the time k of first prescription of Metformin. From the posterior mean, we perform ancestral sampling tracking two latent trajectories: (1) the factual: where we sample new latent states conditioned on the medication u_t the patient had actually received and (2) the counterfactual: where we sample conditioned on not receiving any drugs for all remaining timesteps (i.e u_k set to the zero-vector). We reconstruct the patient observations $x_k, \ldots, x_T$, threshold the predicted values of A1C levels into high and low and visualize the average number of highs A1C levels we observe among the synthetic patients in both scenarios. This is an example of performing do-calculus (Pearl, 2009) in order to estimate model-based counterfactual effects.

The results are shown in Figure 5. We see the model learns that, on average, patients who were prescribed diabetes medication had more controlled levels of A1C than patients who did not receive any medication. Despite being an aggregate effect, this is interesting because it is a phenomenon that coincides with our intuition but was confirmed by the model in an entirely unsupervised manner. Note that in our model, most diabetic patients are indeed prescribed medications, making the counterfactual prediction harder. The ability of this model to answer such queries opens up possibilities into building personalized neural models of healthcare. Further experiments, samples from the learned generative model and implementation details may be found in the appendix.

7. Discussion

We introduce a general algorithm for scalable learning in a rich family of latent variable models for time-series data. The space complexity of our learning algorithm depends neither on the sequence length T nor on the training set size N, offering massive savings compared to classical variational inference methods. Since we use RNNs only in the inference network, it should be possible to continue to increase their capacity and condition on different modalities that might be relevant to posterior inference without worry of overfitting the data. Finally, we showcased an application of the learning algorithm to modeling longitudinal patient data in electronic health records and inferring treatment effect.

Acknowledgements

The Tesla K40s used for this research were donated by the NVIDIA Corporation. The authors gratefully acknowledge support by the DARPA Probabilistic Programming for Advancing Machine Learning (PPAML) Program under AFRL prime contract no. FA8750-14-C-0005, ONR #N00014-13-1-0646, and NSF CAREER award #1350965. We thank the anonymous reviewers for the comments.

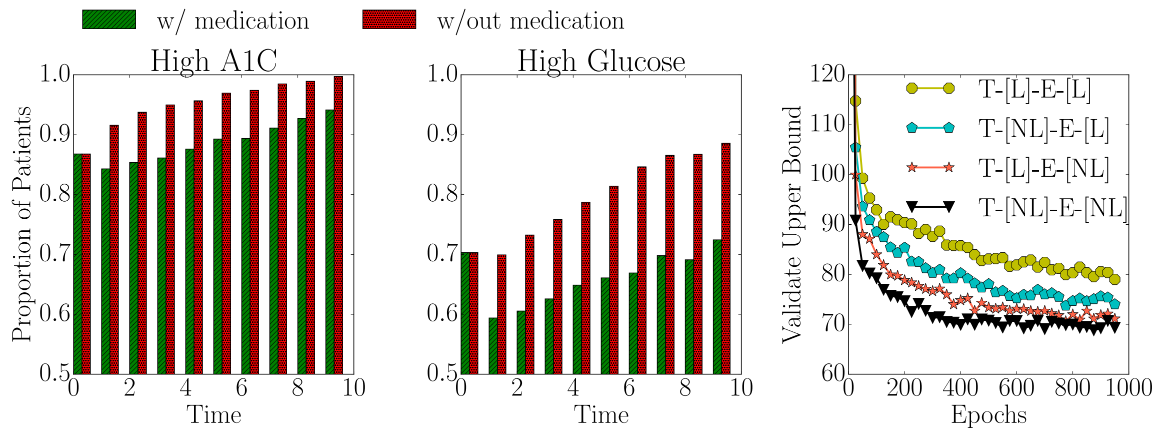


Figure 5: (Left Two Plots) Estimating Counterfactuals with DMM: The x-axis denotes the number of 3-month intervals after prescription of Metformin. The y-axis denotes the proportion of patients (out of a test set size of 800) who, after their first prescription of Metformin, experienced a high level of A1C. In each tuple of bar plots at every time step, the left aligned bar plots (green) represent the population that recieved diabetes medication while the right aligned bar plots (red) represent the population that did not recieve diabetes medication. **(Rightmost Plot)** Upper bound on negative-log likelihood for different DMMs trained on the medical data. (T) denotes “transition”, (E) denotes “emission”, (L) denotes “linear” and (NL) denotes “non-linear”.

References

Evan Archer, Il Memming Park, Lars Buesing, John Cunningham, and Liam Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Nicolas Boulanger-lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML*, 2012.

Thomas Briegel and Volker Tresp. Fisher scoring and a mixture of modes approach for approximate inference and learning in nonlinear state space models. In *NIPS*, 1999.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *NIPS*, 2015.

Daniel Duckworth. Kalman filter, kalman smoother, and em library for python. `https://pykalman.github.io/`, 2016. Accessed: 2016-02-24.

Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv:1412.6581*, 2014.

Zhe Gan, Chunyuan Li, Ricardo Henao, David E Carlson, and Lawrence Carin. Deep temporal sigmoid belief networks for sequence modeling. In *NIPS*, 2015.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, 2015.

Shixiang Gu, Zoubin Ghahramani, and Richard E Turner. Neural adaptive sequential monte carlo. In *NIPS*, 2015.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Structured VAEs: Composing probabilistic graphical models and variational autoencoders. *arXiv preprint arXiv:1603.06277*, 2016.

C. Kaae Sønderby, T. Raiko, L. Maaløe, S. Kaae Sønderby, and O. Winther. How to Train Deep Variational Autoencoders and Probabilistic Ladder Networks. *ArXiv e-prints*, February 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, 2011.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *ICML*, 2014.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Tapani Raiko and Matti Tornio. Variational bayesian learning of nonlinear hidden state-space models for model predictive control. *Neurocomputing*, 72(16):3704–3712, 2009.

Tapani Raiko, Matti Tornio, Antti Honkela, and Juha Karhunen. State inference in variational bayesian nonlinear state-space models. In *International Conference on ICA and Signal Separation*, pages 222–229. Springer, 2006.

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Sam Roweis and Zoubin Ghahramani. An EM algorithm for identification of nonlinear dynamical systems. 2000.

Thomas B Schön, Adrian Wills, and Brett Ninness. System identification of nonlinear state-space models. *Automatica*, 47(1):39–49, 2011.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. abs/1605.02688, May 2016. URL `http://arxiv.org/abs/1605.02688`.

Harri Valpola and Juha Karhunen. An unsupervised ensemble learning method for nonlinear dynamic state-space models. *Neural computation*, 14(11):2647–2692, 2002.

Eric Wan, Ronell Van Der Merwe, et al. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium*. IEEE, 2000.

Eric A. Wan and Alex T. Nelson. Dual kalman filtering methods for nonlinear prediction, smoothing and estimation. In *NIPS*, 1996.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015.

Appendix

Appendix A. Lower Bound on the Likelihood of data

We can derive the bound on the likelihood $\mathcal{L}(\vec{x};(\theta,\phi))$ as follows:

$$\log p_\theta(\vec{x}) \geq \int_{\vec{z}} q_\phi(\vec{z}|\vec{x}) \log \frac{p_\theta(\vec{z})p_\theta(\vec{x}|\vec{z})}{q_\phi(\vec{z}|\vec{x})} d\vec{z} \quad \textit{(Using } x_t \perp\!\!\!\perp x_{\neg t}|\vec{z} \textit{)}$$
$$= \mathop{\mathbb{E}}_{q_\phi(\vec{z}|\vec{x})}[\log p_\theta(\vec{x}|\vec{z})] - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z})) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{q_\phi(z_t|\vec{x})}[\log p_\theta(x_t|z_t)] - \mathrm{KL}(q_\phi(\vec{z}|\vec{x})||p_\theta(\vec{z})) \quad (6)$$
$$= \mathcal{L}(\vec{x};(\theta,\phi))$$

In the following we omit the dependence of q on $\vec{x}$, and omit the subscript ϕ. We can show that the KL divergence between the approximation to the posterior and the prior simplifies as:

$$\mathrm{KL}(q(z_1,\ldots,z_T)||p(z_1,\ldots,z_T)) = \int_{z_1}\ldots\int_{z_T} q(z_1)\ldots q(z_T|z_{T-1}) \log \frac{p(z_1,z_2,\ldots,z_T)}{q(z_1)\ldots q(z_T|z_{T-1})}$$

(Factorization of the variational distribution)

$$= \int_{z_1}\ldots\int_{z_T} q(z_1)\ldots q(z_T|z_{T-1}) \log \frac{p(z_1)p(z_2|z_1)\ldots p(z_T|z_{T-1})}{q(z_1)\ldots q(z_T|z_{T-1})}$$

(Factorization of the prior)

$$= \int_{z_1}\ldots\int_{z_T} q(z_1)\ldots q(z_T|z_{T-1}) \log \frac{p(z_1)}{q(z_1)} + \sum_{t=2}^{T}\int_{z_1}\ldots\int_{z_T} q(z_1)\ldots q(z_T|z_{T-1}) \log \frac{p(z_t|z_{t-1})}{q(z_t|z_{t-1})}$$
$$= \int_{z_1} q(z_1) \log \frac{p(z_1)}{q(z_1)} + \sum_{t=2}^{T}\int_{z_{t-1}}\int_{z_t} q(z_t) \log \frac{p(z_t|z_{t-1})}{q(z_t|z_{t-1})}$$

(Each expectation over z_t is constant for $t \notin \{t, t-1\}$)

$$= \mathrm{KL}(q(z_1)||p(z_1)) + \sum_{t=2}^{T} \mathop{\mathbb{E}}_{q(z_{t-1})}[\mathrm{KL}(q(z_t|z_{t-1})||p(z_t|z_{t-1}))] \quad (7)$$

For evaluating the marginal likelihood on the test set, we can use the following Monte-Carlo estimate:

$$p(\vec{x}) \approxeq \frac{1}{S}\sum_{s=1}^{S} \frac{p(\vec{x}|\vec{z}^{(s)})p(\vec{z}^{(s)})}{q(\vec{z}^{(s)}|\vec{x})} \quad \vec{z}^{(s)} \sim q(\vec{z}|\vec{x}) \quad (8)$$

This may be derived in a manner akin to the one depicted in Appendix E (Rezende et al., 2014) or Appendix D (Kingma and Welling, 2014).

The log likelihood on the test set is computed using:

$$\log p(\vec{x}) \approxeq \log \frac{1}{S}\sum_{s=1}^{S} \exp\log\left[\frac{p(\vec{x}|\vec{z}^{(s)})p(\vec{z}^{(s)})}{q(\vec{z}^{(s)}|\vec{x})}\right] \quad (9)$$

Eq. 9 may be computed in a numerically stable manner using the log-sum-exp trick.

Appendix B. KL divergence between Prior and Posterior

Maximum likelihood learning requires us to compute:

$$\mathrm{KL}(q(z_1,\ldots,z_T)||p(z_1,\ldots,z_T)) = \mathrm{KL}(q(z_1)||p(z_1)) + \sum_{t=2}^{T-1} \mathop{\mathbb{E}}_{q(z_{t-1})}[\mathrm{KL}(q(z_t|q_{t-1})||p(z_t|z_{t-1}))] \tag{10}$$

The KL divergence between two multivariate Gaussians q, p with respective means and covariances $\mu_q, \Sigma_q, \mu_p, \Sigma_p$ can be written as:

$$\mathrm{KL}(q||p) = \frac{1}{2}(\underbrace{\log\frac{|\Sigma_p|}{|\Sigma_q|}}_{(a)} - D + \underbrace{\mathrm{Tr}(\Sigma_p^{-1}\Sigma_q)}_{(b)} + \underbrace{(\mu_p-\mu_q)^T\Sigma_p^{-1}(\mu_p-\mu_q)}_{(c)}) \tag{11}$$

The choice of q and p is suggestive. using (10) & (11), we can derive a closed form for the KL divergence between $q(z_1 \ldots z_T)$ and $p(z_1 \ldots z_T)$. μ_q, Σ_q are the outputs of the variational model. Our functional form for μ_p, Σ_p is based on our generative and can be summarized as:

$$\mu_{p1} = 0 \qquad \Sigma_{p1} = \mathbb{1} \qquad\qquad \mu_{pt} = G(z_{t-1}, u_{t-1}) = G_{t-1} \qquad \Sigma_{pt} = \Delta\vec{\sigma}$$

Here, Σ_{pt} is assumed to be a learned diagonal matrix and Δ a scalar parameter.

Term (a) For $t = 1$, we have:

$$\log\frac{|\Sigma_{p1}|}{|\Sigma_{q1}|} = \log|\Sigma_{p1}| - \log|\Sigma_{q1}| = -\log|\Sigma_{q1}| \tag{12}$$

For $t > 1$, we have:

$$\log\frac{|\Sigma_{pt}|}{|\Sigma_{qt}|} = \log|\Sigma_{pt}| - \log|\Sigma_{qt}| = D\log(\Delta) + \log|\vec{\sigma}| - \log|\Sigma_{qt}| \tag{13}$$

Term (b) For $t = 1$, we have:

$$\mathrm{Tr}(\Sigma_{p1}^{-1}\Sigma_{q1}) = \mathrm{Tr}(\Sigma_{q1}) \tag{14}$$

For $t > 1$, we have:

$$\mathrm{Tr}(\Sigma_{pt}^{-1}\Sigma_{qt}) = \frac{1}{\Delta}\mathrm{Tr}(\mathrm{diag}(\vec{\sigma})^{-1}\Sigma_{qt}) \tag{15}$$

Term (c) For $t = 1$, we have:

$$(\mu_{p1}-\mu_{q1})^T\Sigma_{p1}^{-1}(\mu_{p1}-\mu_{q1}) = ||\mu_{q1}||^2 \tag{16}$$

For $t > 1$, we have:

$$(\mu_{pt}-\mu_{qt})^T\Sigma_{pt}^{-1}(\mu_{pt}-\mu_{qt}) = \Delta(G_{t-1}-\mu_{qt})^T\,\mathrm{diag}(\vec{\sigma})^{-1}(G_{t-1}-\mu_{qt}) \tag{17}$$

Rewriting (10) using (12), (13), (14), (15), (16), (17), we get:

$$\mathrm{KL}(q(z_1,\ldots,z_T)||p(z_1,\ldots,z_T)) = \frac{1}{2}((T-1)D\log(\Delta)\log|\vec{\sigma}| - \sum_{t=1}^{T}\log|\Sigma_{qt}|$$

$$+ \mathrm{Tr}(\Sigma_{q1}) + \frac{1}{\Delta}\sum_{t=2}^{T}\mathrm{Tr}(\mathrm{diag}(\vec{\sigma})^{-1}\Sigma_{qt}) + ||\mu_{q1}||^2 + \Delta\sum_{t=2}^{T}\mathop{\mathbb{E}}_{z_{t-1}}\left[(G_{t-1}-\mu_{qt})^T\,\mathrm{diag}(\vec{\sigma})^{-1}(G_{t-1}-\mu_{qt})\right])$$

Algorithm 1 Learning a DMM with Stochastic Gradient Descent. We use Monte-Carlo (MC) estimates over K samples from the recognition network during learning to evaluate expectations in the bound and gradients. During training we use $K = 1$ and aggregate gradients across mini-batches.

```
**Inputs**: Dataset $\mathcal{D} := \left[\vec{x}^1, \dots, \vec{x}^N\right]$
        Inference Model: $q_\phi(\vec{z}|\vec{x})$
        Generative Model: $p_\theta(\vec{x}|\vec{z}), p_\theta(\vec{z})$
while $notConverged()$ do
    1. Sample datapoint: $(\vec{x}, \vec{u}) \sim \mathcal{D}$
    2. Estimate posterior parameters (Evaluate $\mu_\phi, \Sigma_\phi$)
    3. Sample $\hat{\vec{z}}^k \sim q_\phi(\vec{z}|\vec{x}), k = 1, \dots, K$
    4. Estimate conditional likelihood: $p_\theta(\vec{x}|\hat{\vec{z}}^k)$ & KL
    5. Evaluate $\mathcal{L}(\vec{x}; (\theta, \phi))$ using $K$ samples
    6. Estimate MC approx. to $\nabla_\theta \mathcal{L}$ with $K$ samples
    7. Estimate MC approx. to $\nabla_\phi \mathcal{L}$ with $K$ samples
    (Use stochastic backpropagation to move gradients with respect to $q_\phi$ inside expectation)
    8. Update $\theta, \phi$ using ADAM (Kingma and Ba, 2015)
end while
```

Appendix C. Learning Algorithm

Algorithm 1 depicts an overview of the learning algorithm. We outline the algorithm for a mini-batch of size one, but in practice gradients are averaged across stochastically sampled mini-batches of the training set to mitigate the effect of using a single sample ($K = 1$) during training for estimating expectations and their corresponding gradients. We take a gradient step in θ and ϕ, typically with an adaptive learning rate such as Kingma and Ba (2015). For the results in Table 3 in the main paper, as in (Kaae Sønderby et al., 2016), we found annealing the KL divergence in the variational bound ($\mathcal{L}(\vec{x}; (\theta, \phi))$) from 0 to 1 over 5000 parameter updates got better results.

Factorization of the posterior distribution We use the independence statements implied by the graphical model in Figure 1 of the main paper to note that $p(\vec{z}|\vec{x})$, the true posterior, factorizes as:

$$p(\vec{z}|\vec{x}) = p(z_1|\vec{x}) \prod_{t=2}^{T} p(z_t|z_{t-1}, \vec{x})$$

Now, we notice that $z_t \perp\!\!\!\perp x_1, \dots, x_{t-1}|z_{t-1}$, yielding the desired result.

Appendix D. Polyphonic Music Generation

Experimental Setup: For the polyphonic experiments, we used two-layer MLPs in the emission and the (gated) transition function. The hidden dimension was set to be 100 for the emission distribution and 200 in the transition function. We typically used an RNN sizes from one of $\{400, 600\}$ and a latent dimension of size 100.

Samples: Figure 6 depicts mean probabilities of samples from the generative model trained JSB Chorales (Boulanger-lewandowski et al., 2012). MP3 songs corresponding to two different samples from the best DMM model learned on each of the four polyphonic data sets may be found in the code repository.

Experiments with NADE: We also experiment with Neural Autoregressive Density Estimators (NADE) (Larochelle and Murray, 2011) in the emission distribution for DMM-Aug and denote it DMM-Aug-NADE. In Table 4, we see that DMM-Aug-NADE performs comparably to the state of the art RNN-NADE on JSB, Nottingham and Piano.

Table 4: Experiments with NADE Emission: Test negative log-likelihood (lower is better) on Polyphonic Music Generation dataset. **Table Legend**: RNN-NADE (Boulanger-lewandowski et al., 2012)

Methods	JSB	Nottingham	Piano	Musedata
DMM-Aug.-NADE	5.118 (5.335) {5.264}	2.305 (2.347) {2.364}	7.048 (7.099) {7.361}	6.049 (6.115) {5.247}
RNN-NADE	5.19	2.31	7.05	5.60

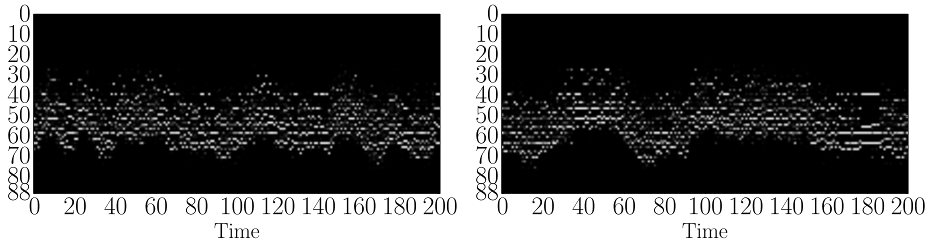


Figure 6: Two samples from the DMM trained on JSB Chorales

Appendix E. Experimental Results on Synthetic Data

Experimental Setup: We used an RNN size of 40 in the inference networks used for the synthetic experiments.

Linear SSMs : Figure 7 (N=500, T=25) depicts the performance of inference networks using the same setup as in the main paper, only now using held out data to evaluate the RMSE and the upper bound. We find that the results echo those in the training set, and that on unseen data points, the inference networks, particularly the structured ones, are capable of generalizing compiled inference.

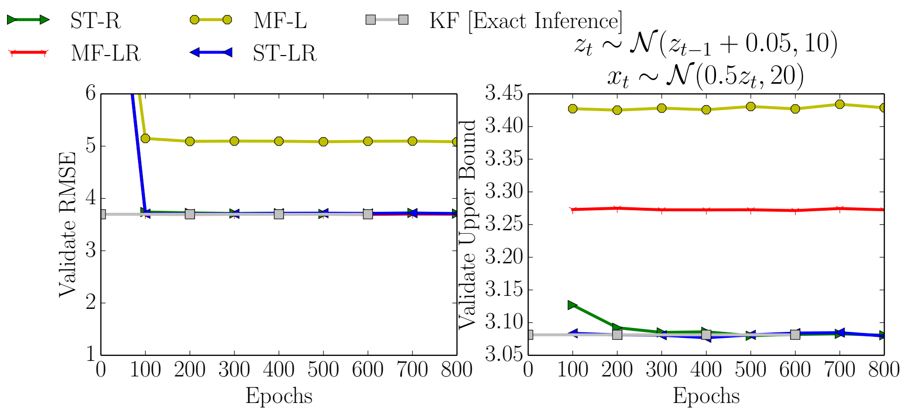
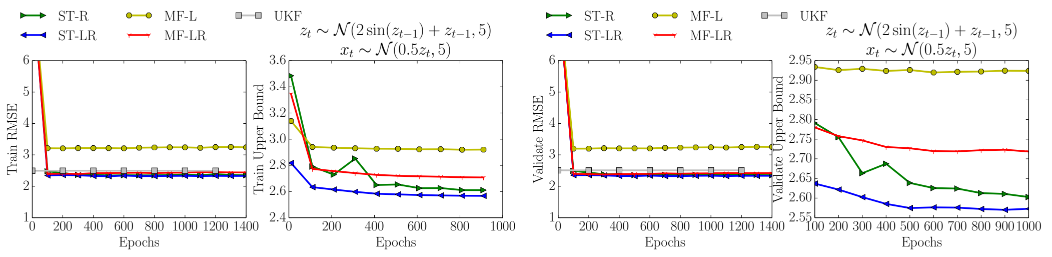


Figure 7: Inference in a Linear SSM on Held-out Data: Performance of inference networks on held-out data using a generative model with Linear Emission and Linear Transition (same setup as main paper)

Non-linear SSMs : Figure 8 considers learning inference networks on a synthetic non-linear dynamical system ($N = 5000, T = 25$). We find once again that inference networks that match the posterior realize both faster convergence and better training (and validation) accuracy.

Visualizing Inference: In Figure 9 we visualize the posterior estimates obtained by the inference network. We run posterior inference on the training set 100 times and take the empirical expectation of the posterior means and covariances of each method. For the linear generative model we used above, we compare with



(a) Performance on training data **(b)** Performance on held-out data

Figure 8: Inference in a Non-linear SSM: Performance of inference networks trained with data from a Linear Emission and Non-linear Transition SSM

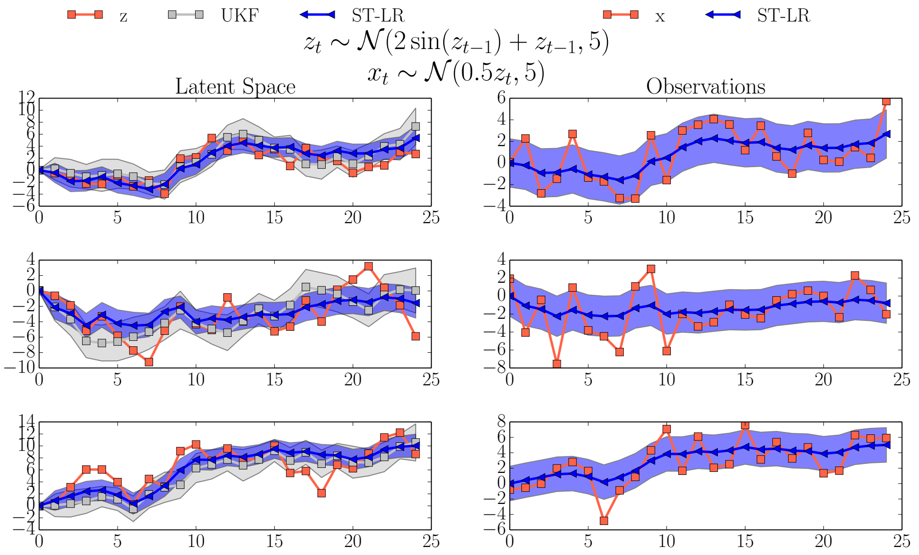


Figure 9: Inference on Non-linear Synthetic Data: Visualizing inference on training data. Generative Models: (a) Linear Emission and Non-linear Transition z^* denotes the latent variable that generated the observation. x denotes the true data. We compare against the results obtained by a smoothed Unscented Kalman Filter (UKF) (Wan et al., 2000). The column denoted “Observations" denotes the result of applying the emission function of the respective generative model on the posterior estimates shown in the column “Latent Space". The shaded areas surrounding each curve μ denotes $\mu \pm \sigma$ for each plot.

the exact posterior means obtained from a Smoothed Kalman Filter. We also generate data from a non-linear model in which case we compare with the Unscented Kalman Filter (UKF) Wan et al. (2000). Both baselines have access to the underlying generative model parameters which they use to perform inference.

Appendix F. Generative Models of Medical Data

Data from Electronic Health Records is noisy, high dimensional and difficult to characterize easily. Patient records are rarely contiguous over large parts of the dataset and data is often missing (not at random). Modeling such data is a challenging task. We use the DMM for this since:

- The generative model of a DMM naturally expresses the idea of a time-evolving latent state of the patient (z_t)
- The deep neural networks are capable of representing arbitrarily complex non-linear functions, justifying their use in modeling the unknown emission and transition functions

Graphical Model: As described in the main paper, we augment the DMM with an additional edge every time step from an external input u_t that represents the (binary) set of medications that are prescribed to the patient. Figure 10 represents the generative model when $T = 4$. The additional dotted edges in the Bayesian network exist but never need to be modeled since x_t and u_t are always assumed to be observed in the scenarios that we evaluate. A natural line of follow up work would be to consider learning when u_t is missing or latent.

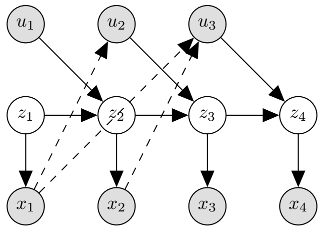


Figure 10: DMM for Medical Data: The DMM is augmented with external observations u_t representing medications presented to the patient. z_t is the latent state of the patient. x_t are the observations that we model Since both u_t and x_t are always assumed observed, the conditional distribution $p(u_t|x_1, \ldots, x_{t-1})$ may be ignored during learning.

.

Learning in the Presence of Missing Data: In the original patient data, A1C and Glucose values are not measured at every visit. To deal with this difficulty, we marginalize out missing values during learning. The sub-network of the original graph we are concerned with is the emission function since missingness affects our ability to evaluate $\log p(x_t|z_t)$. However, as we will show, this marginalization is not difficult since the missing random variables are leaves in the Bayesian sub-network (comprised of the emission function). To illustrate this, consider the case where we are modelling two observations at time t, namely m_t, o_t. The log-likelihood of the data (m_t, o_t) conditioned on the latent variable z_t decomposes as $\log p(m_t, o_t|z_t) = \log p(m_t|z_t) + \log p(o_t|z_t)$ since the random variables are conditionally independent given their parent.

Now, if m is missing and we wish to marginalize it out, while o_t is observed then our log-likelihood is now: $\log \int_m p(m_t, o_t|z_t) = \log(\int_m p(m_t|z_t)p(o_t|z_t)) = \log p(o_t|z_t)$ (since $\int_m p(m_t|z_t) = 1$) i.e we effectively ignore the missing observations when estimating the log-likelihood of the data.

In our data, we have indicators denoting whether or not the A1C values and Glucose values were observed which we use as markers of missingness. During batch learning, at every time-step t, we obtain a matrix $B = \log p(x_t|z_t)$ of size (batch-size $\times$ 48) comprising the log-likelihoods of every dimension for patients in the batch. We multiply this with a matrix of M. M has the same dimensions as B and has a 1 if the patient's A1C value was observed and a 0 otherwise. For dimensions that are never missing, M is always 1.

Model Parameterizations for Medical Data: We first investigate the different choices to parameterize the emission and transition distribution for the medical data. We consider linear and non-linear functions. In the linear case, the mean and log-covariances in the transition function are parameterized as linear functions of the previous latent state. The non-linear functions are parameterized by two layer MLPs. The results in Figure

11 suggest that a non-linear emission and non-linear transition is important in order to be able to model the data, though a linear transition function might work almost as well. Based on this insight, we use non-linear functions in the emission and transition distribution of the DMM. The hidden dimension was set as 200 for the emission and transition functions. We used an RNN size of 400 and a latent dimension of size 50.

Sampling a Patient: We visualize samples from the DMM in Figure 12 The model captures correlations within timesteps as well as variations in A1C level and Glucose level across timesteps. It also appears to capture the rare occurrences of comorbidities that are found amongst diabetic patients.

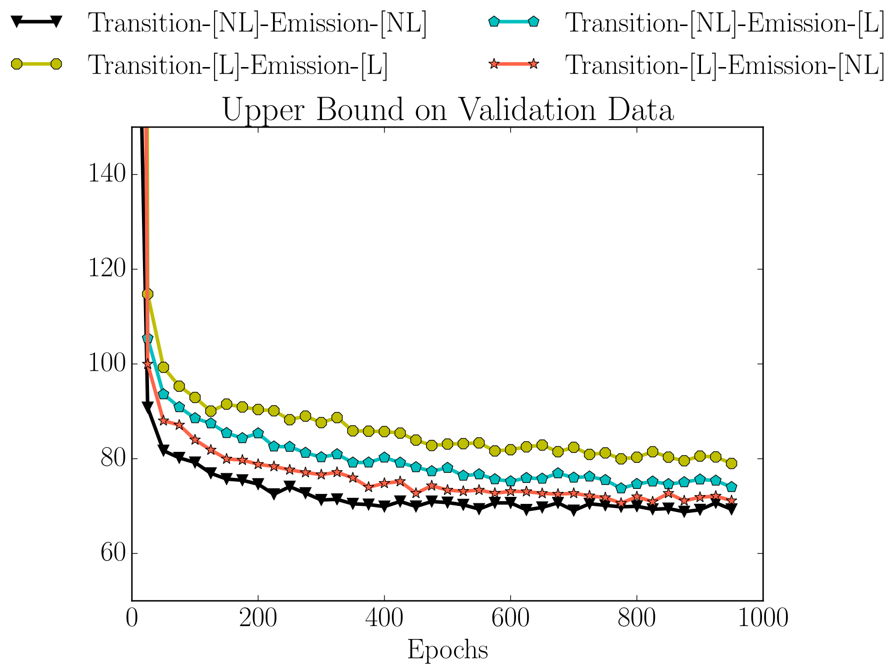


Figure 11: Linear and Non-linear SSMs on EHR Data: Held-out Log-likelihoods on the Medical Data. We plot the upper bound on the likelihood (lower is better). [L] denotes Linear and [NL] denotes non-linear parameterized by a deep-neural network.

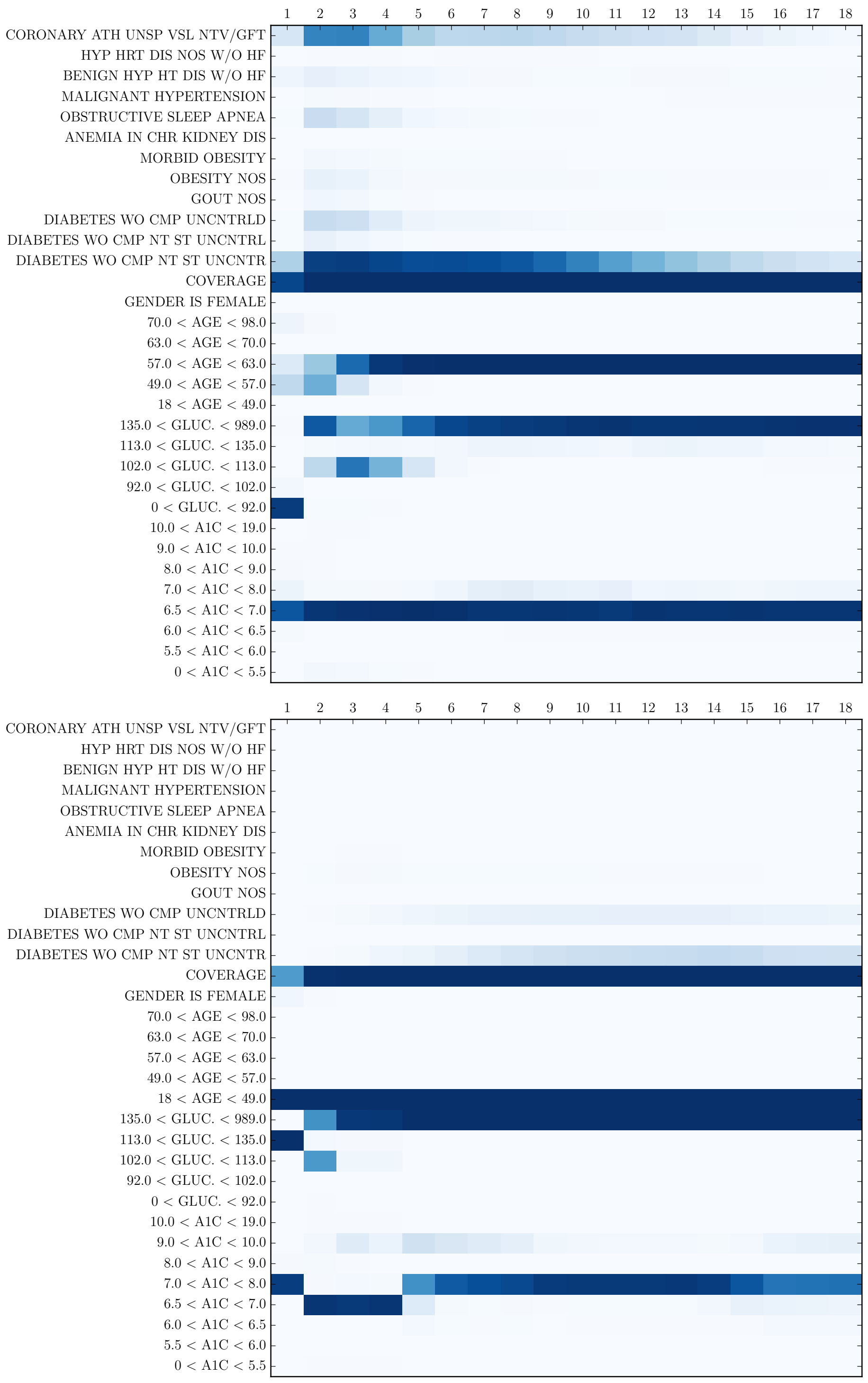


Figure 12: Generated Samples Samples of a patient from the model, including the most important observations. The x-axis denotes time and the y-axis denotes the observations. The intensity of the color denotes its value between zero and one