VARIATIONAL MESSAGE PASSING WITH STRUCTURED INFERENCE NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a variational message-passing algorithm for a class of models that combine deep models with probabilistic graphical models. Our algorithm is a natural-gradient algorithm whose messages automatically reduce to stochastic-gradients for the deep components of the model. Using a special-structure inference network, our algorithm exploits the structural properties of the model to gain computational efficiency while retaining the simplicity and generality of deep-learning algorithms. By combining the strength of two different types of inference procedures, our approach offers a framework that simultaneously enables structured, amortized, and natural-gradient inference for complex models.

1 INTRODUCTION

To analyze real-world data, machine learning relies on models that can extract useful patterns. Deep Neural Networks (DNNs) are a popular choice for this purpose because they can learn flexible representations. Another popular choice are probabilistic graphical models (PGMs) which can find interpretable structures in the data. Recent work on combining these two types of models hopes to exploit their complimentary strengths and provide powerful models that are also easy to interpret (Johnson et al., 2016; Krishnan et al., 2015; Archer et al., 2015; Fraccaro et al., 2016).

To apply such hybrid models to real-world problems, we need efficient algorithms that can extract useful structure using the model. However, the two fields of deep learning and PGMs traditionally use different types of algorithms. For deep learning, stochastic-gradient (SG) methods are the most popular choice, e.g., those based on back-propagation. These algorithms are not only widely applicable, but can also employ amortized inference to enable fast inference at test time (Rezende et al., 2014; Kingma & Welling, 2013). On the other hand, most popular algorithms for PGMs exploit the model's graphical conjugacy structure to gain computational efficiency, e.g., variational message passing (VMP) (Winn & Bishop, 2005), expectation propagation (Minka, 2001), Kalman filtering, and more recently stochastic variational inference (SVI) (Hoffman et al., 2013). In short, the two fields of deep learning and probabilistic modelling employ fundamentally different inferential strategies and a natural question is, whether we can design algorithms that combine their respective strengths.

There have been several attempts to design such methods in the recent years, e.g., Krishnan et al. (2015); Fraccaro et al. (2016); Archer et al. (2015); Johnson et al. (2016). Our work in this paper is inspired by the previous work of Johnson et al. (2016) that aims to combine message-passing, natural-gradient, and amortized inference. Our proposed algorithm in this paper simplifies and generalizes the algorithm of Johnson et al. (2016).

To do so, we propose Structured Inference Networks (SIN) that combine the strengths of standard inference networks used in deep learning with structured PGM models. We derive conditions under which such inference networks can enable efficient inference procedure. By using a recent VMP method of Khan & Lin (2017), we derive a variational message-passing algorithm whose messages automatically reduce to stochastic-gradients for the deep components of the model, while perform natural-gradient updates for PGM part. Overall, our algorithm enables Structured, Amortized, and Natural-gradient (SAN) updates and therefore we call our algorithm the SAN algorithm. We show that our algorithm give comparable performance to the method of Johnson et al. (2016) while simplifying and generalizing it.

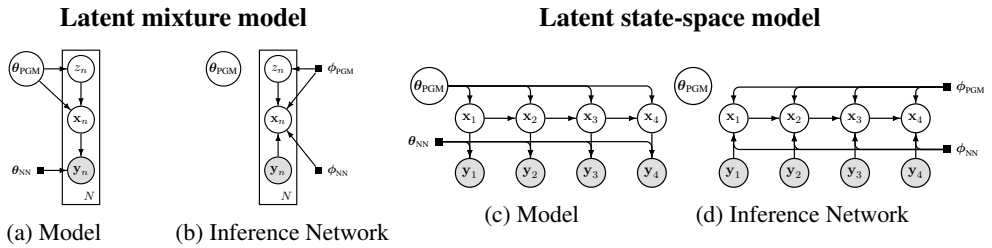


Figure 1: Two examples of model classes that combine probabilistic graphical models with deep neural networks. The two figures on the left show the model and corresponding inference network for the latent mixture model, while the two figures on the right show the same for the latent state-space model. Our inference network mimics the structure of the model by using additional parameters $\boldsymbol{\phi}$. The inference network differs from the model in two ways. First, the arrows from $\mathbf{x}_n$ to $\mathbf{y}_n$ in the model are reversed in the inference network. Second, the model parameters $\boldsymbol{\theta}$ are decoupled in the inference network due to the mean-field assumption. The latter plays a crucial role to improve over the method of Johnson et al. (2016).

2 The Model and Challenges with Its Inference

We consider the modelling of data vectors $\mathbf{y}_n$ by using local latent vectors $\mathbf{x}_n$. Following previous works (Johnson et al., 2016; Archer et al., 2015; Krishnan et al., 2015), we model the output $\mathbf{y}_n$ given $\mathbf{x}_n$ using a neural network with parameters $\boldsymbol{\theta}_{\text{NN}}$, and capture the correlations among data vectors $\mathbf{y} := \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$ using a probabilistic graphical model (PGM) over the latent vectors $\mathbf{x} := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Specifically, we use the following joint distribution:

$$p(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) := \underbrace{\left[\prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\theta}_{\text{NN}})\right]}_{\text{DNN}} \underbrace{\left[p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})\right]}_{\text{PGM}} \underbrace{\left[p(\boldsymbol{\theta}_{\text{PGM}})\right]}_{\text{Hyperprior}}, \tag{1}$$

where $\boldsymbol{\theta}_{\text{NN}}$ and $\boldsymbol{\theta}_{\text{PGM}}$ are parameters of a DNN and PGM respectively, and $\boldsymbol{\theta} := \{\boldsymbol{\theta}_{\text{NN}}, \boldsymbol{\theta}_{\text{PGM}}\}$.

This combination of probabilistic graphical model and neural network is referred to as structured variational auto-encoder (SVAE) by Johnson et al. (2016). SVAE employs a structured prior $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$ to extract useful structure from the data. SVAE therefore differs from VAE (Kingma & Welling, 2013) where the prior distribution over $\mathbf{x}$ is simply a multivariate Gaussian distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|0, \mathbf{I})$ with no special structure. To illustrate this difference, we now give an example of a structured prior which is useful to obtain clustering of the outputs $\mathbf{y}_n$.

Example (Mixture-Model Prior) : Suppose we wish to group the output $\mathbf{y}_n$ into K distinct clusters, in this case the standard Gaussian prior, which is used in VAE, is not a useful prior. In SVAE, we could instead use a mixture-model prior over $\mathbf{x}_n$ (Johnson et al., 2016),

$$p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}}) = \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\theta}_{\text{PGM}}) = \prod_{n=1}^{N} \left[\sum_{k=1}^{K} p(\mathbf{x}_n|z_n = k)\pi_k\right] \tag{2}$$

where $z_n \in \{1, 2, \ldots, K\}$ is the mixture indicator for n'th data example, and π_k are mixing proportions that sum to 1. Each cluster can be modelled, for example, by using a Gaussian mixture model (GMM) where each mixture component $p(\mathbf{x}_n|z_n = k) := \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ is a Gaussian with mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$, in which case $\boldsymbol{\theta}_{\text{PGM}} := \{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \pi_k\}_{k=1}^{K}$. The graphical model of an SVAE with a GMM prior is shown in Figure 1a. Such structured-prior is useful to discover cluster of outputs $\mathbf{y}_n$ by clustering the latent variables $\mathbf{x}_n$.

Given such models, our goal is to approximate the posterior distribution $p(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y})$. In particular, in a similar spirit to VAE, we would like to approximate the posterior of $\mathbf{x}$ by using an inference network. In VAE, such inference networks are obtained by using functions parameterized using a DNN, as shown below where the generative model for VAE is shown in the left hand side while the

inference network is shown on the right hand side:

$$p(\mathbf{x}|\mathbf{y}, \boldsymbol{\theta}_{\text{NN}}) = \frac{1}{p(\mathbf{y}|\boldsymbol{\theta})} \prod_{n=1}^{N} [p(\mathbf{y}_n|\mathbf{x}_n, \boldsymbol{\theta}_{\text{NN}})\mathcal{N}(\mathbf{x}_n|0, \mathbf{I})] \approx \prod_{n=1}^{N} q(\mathbf{x}_n|f_\phi(\mathbf{y}_n)). \tag{3}$$

The distribution q is typically an exponential-family distribution whose natural-parameters are modelled by using a DNN f_ϕ with parameters ϕ. The same function $f_\phi(\cdot)$ is used for all n which reduces the number of variational parameters and enables sharing of statistical strengths across n. This leads to both faster training and testing (Rezende et al., 2014).

Unfortunately, for SVAE, such inference networks may give inaccurate predictions since they ignore the prior structure of the PGM prior $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$. For example, suppose $\mathbf{y}_n$ is a time-series and we model $\mathbf{x}_n$ using a dynamical system as depicted in Fig. 1c. In this case, the inference network of (3) is not an accurate approximation since it ignores the time-series structure in $\mathbf{x}$. This might result in inaccurate predictions of distant future observations, e.g., prediction for an observation $\mathbf{y}_{10}$ given the past data $\{\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3\}$ would be inaccurate because the inference network has no path connecting $\mathbf{x}_{10}$ to $\mathbf{x}_1, \mathbf{x}_2$, or $\mathbf{x}_3$. In cases like these where the prior structure plays an important role in prediction, it is important to incorporate this structure in the inference network.

A solution to this problem is to use an inference network with the same structure as the model but to replace all its edges by neural networks (Krishnan et al., 2015; Fraccaro et al., 2016). This solution is reasonable when the PGM itself is complex, but this might be too aggressive when the PGM is a simple model, e.g., for the linear dynamical system in Fig. 1c. Using DNNs in such cases would dramatically increase the number of parameters which will lead to a possible deterioration in both speed and performance.

Johnson et al. (2016) propose an alternative algorithm to incorporate the structure of the PGM part. They consider optimizing the standard mean-field lower bound: $\mathcal{L}(\boldsymbol{\lambda}_x, \boldsymbol{\lambda}_\theta) := \mathbb{E}_q[\log p(\mathbf{y}, \mathbf{x}, \boldsymbol{\theta}) - \log q(\mathbf{x}|\boldsymbol{\lambda}_x) q(\boldsymbol{\theta}|\boldsymbol{\lambda}_\theta)]$ when $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$ is a conditionally-conjugate distribution and q are minimal exponential-family distributions with natural parameters $\boldsymbol{\lambda}_x$ and $\boldsymbol{\lambda}_\theta$ respectively. To incorporate an inference network, they define the following surrogate lower-bound where the DNN likelihood in $\mathcal{L}$ is replaced by the inference network of (3),

$$\hat{\mathcal{L}}(\boldsymbol{\lambda}_x, \boldsymbol{\lambda}_\theta, \boldsymbol{\phi}) := \mathbb{E}_{q(\theta)q(x)} \left[\log \left\{ \prod_{n=1}^{N} q(\mathbf{x}_n|f_\phi(\mathbf{y}_n)) p(\mathbf{x}, \boldsymbol{\theta}_{\text{PGM}}) \right\} - \log \{q(\mathbf{x}|\boldsymbol{\lambda}_x) q(\boldsymbol{\theta}|\boldsymbol{\lambda}_\theta)\} \right], \tag{4}$$

and constrain $\boldsymbol{\lambda}_x$ to be its local optimizer, i.e., $\boldsymbol{\lambda}_x^*(\boldsymbol{\lambda}_\theta, \boldsymbol{\phi}) = \arg\max_{\lambda_x} \hat{\mathcal{L}}(\boldsymbol{\lambda}_x, \boldsymbol{\lambda}_\theta, \boldsymbol{\phi})$. When the factors $q(\mathbf{x}_n|f_\phi(\mathbf{y}_n))$ are chosen to be conjugate to $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$, this optimization can be performed efficiently using VMP. To exploit this efficiency, Johnson et al. (2016) propose a two-stage procedure where, in every iteration, they compute $\boldsymbol{\lambda}_x^*(\boldsymbol{\lambda}_\theta, \boldsymbol{\phi})$ for a given $\boldsymbol{\lambda}_\theta$ and $\boldsymbol{\phi}$, and then optimize the mean-field objective $\mathcal{L}(\boldsymbol{\lambda}_x^*(\boldsymbol{\lambda}_\theta, \boldsymbol{\phi}), \boldsymbol{\lambda}_\theta)$. The hope is that $\boldsymbol{\lambda}_x^*$ can be computed efficiently using VMP, while the update of $\boldsymbol{\lambda}_\theta$ can be obtained using a natural-gradient method similar to SVI, and $\boldsymbol{\phi}$ can be optimized using SGD as in VAE.

This two-stage optimization is equivalent to an implicitly-constrained optimization problem. Such problems are typically more difficult to solve than their unconstrained counterparts, especially when the constraints are nonconvex (Heinkenschloss, 2008). Theoretically, the convergence of such methods is difficult to guarantee when the constraints are violated. In practice, this makes the implementation difficult because in every iteration the VMP updates need to run long enough to reach close to a local optimum of the surrogate lower bound.

Another disadvantage of the method of Johnson et al. (2016) is that it applies only under restrictive assumptions on the PGM prior. For example, it does not work for PGMs that contain non-conjugate factors because in that case VMP cannot be used to optimize the surrogate lower bound. In addition, it does not apply directly when $\boldsymbol{\lambda}_x$ is constrained, and in case $q(\mathbf{x}|\boldsymbol{\lambda}_x)$ has additional latent variables (e.g., indicator variables z_n in GMM), the VMP iterations may need to run quite long to reach a local optimum. Due to these issues, the method of Johnson et al. (2016) presents serious implementation challenge for generic PGMs.

3 STRUCTURED INFERENCE NETWORKS

Our goal is to design an inference network that preserves the PGM's probabilistic structure while enabling inference with a computationally-efficient message-passing algorithm. We propose the following *structured* inference network (SIN) which consists of two types of factors,

$$q(\mathbf{x}|\mathbf{y},\boldsymbol{\phi}) := \frac{1}{\mathcal{Z}(\boldsymbol{\phi})} \underbrace{\left[\prod_{n=1}^{N} q(\mathbf{x}_n|f_{\phi_{\text{NN}}}(\mathbf{y}_n))\right]}_{\text{DNN Factor}} \underbrace{\left[q(\mathbf{x}|\boldsymbol{\phi}_{\text{PGM}})\right]}_{\text{PGM Factor}}. \tag{5}$$

The DNN factor here is similar to (3) while the PGM factor is an exponential-family distribution which has a similar graph structure as the PGM prior $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$. The role of the DNN term is to enable flexibility while the role of the PGM term is to incorporate the model's PGM structure into the inference network. Both factors have their own parameters, where $\boldsymbol{\phi}_{\text{NN}}$ is the parameter of the DNN and $\boldsymbol{\phi}_{\text{PGM}}$ is the natural parameters of the PGM factor. The set of parameters is $\boldsymbol{\phi} = \{\boldsymbol{\phi}_{\text{NN}}, \boldsymbol{\phi}_{\text{PGM}}\}$.

How should we choose the two factors? As we will show soon that, for tractable inference, these factors need to satisfy the following two conditions. The first condition is that the normalizing constant $\mathcal{Z}(\boldsymbol{\phi})$ is easy to evaluate and differentiate with respect to $\boldsymbol{\phi}$. Note that both the factors are normalized distribution, but their product may not be. The second condition is that it is easy to draw samples $\mathbf{x}^*(\boldsymbol{\phi}) \sim q(\mathbf{x}|\mathbf{y},\boldsymbol{\phi})$, where we denote a sample by $\mathbf{x}^*(\boldsymbol{\phi})$ to show the direct dependence on $\boldsymbol{\phi}$. An additional desirable feature is to be able to compute the gradient of $\mathbf{x}^*(\boldsymbol{\phi})$ by using the reparameterization trick, although this is not necessary. Now, we will show that given these two conditions we can easily compute the necessary gradients to derive the message passing algorithm.

We show that when the above two conditions are met, the gradient of the lower bound can be computed easily, and then give details of the message passing algorithm in the next section. For now, we assume that $\boldsymbol{\theta}$ is a deterministic variable for now (we will relax this in the next section). The variational lower bound in this case can be written as follows:

$$\mathcal{L}_{\text{SIN}}(\boldsymbol{\theta},\boldsymbol{\phi}) := \mathbb{E}_q\left[\log \frac{p(\mathbf{y},\mathbf{x}|\boldsymbol{\theta})}{q(\mathbf{x}|\mathbf{y},\boldsymbol{\phi})}\right] = \mathbb{E}_q\left[\log \frac{\prod_n \{p(\mathbf{y}_n|\mathbf{x}_n,\boldsymbol{\theta}_{\text{NN}})\}\, p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})\mathcal{Z}(\boldsymbol{\phi})}{\prod_n \{q(\mathbf{x}_n|f_{\phi_{\text{NN}}}(\mathbf{y}_n))\}\, q(\mathbf{x}|\boldsymbol{\phi}_{\text{PGM}})}\right] \tag{6}$$

$$= \sum_{n=1}^{N} \mathbb{E}_q\left[\log p(\mathbf{y}_n|\mathbf{x}_n,\boldsymbol{\theta}_{\text{NN}})\right] + \mathbb{E}_q[\log p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})] - \sum_{n=1}^{N} \mathbb{E}_q[\log q(\mathbf{x}_n|f_{\phi_{\text{NN}}}(\mathbf{y}_n))]$$
$$- \mathbb{E}_q[\log q(\mathbf{x}|\boldsymbol{\phi}_{\text{PGM}})] + \log \mathcal{Z}(\boldsymbol{\phi}) \tag{7}$$

The first three terms are almost identical to the lower bound for the standard VAE, and the last two terms are a result of adding the PGM structure to the inference network. As a result, if we can generate samples $\mathbf{x}^*(\boldsymbol{\phi})$ from SIN, the gradient can be straightforwardly computed using the reparameterization trick similar to VAE. This is shown below.

$$\frac{\partial \mathcal{L}_{\text{SIN}}}{\partial \theta_{\text{NN}}} \approx N \frac{\partial \log p(\mathbf{y}_n|\mathbf{x}_n^*,\boldsymbol{\theta}_{\text{NN}})}{\partial \theta_{\text{NN}}}, \qquad \frac{\partial \mathcal{L}_{\text{SIN}}}{\partial \theta_{\text{PGM}}} \approx \frac{\partial \log p(\mathbf{x}^*|\boldsymbol{\theta}_{\text{PGM}})}{\partial \theta_{\text{PGM}}}, \tag{8}$$

$$\frac{\partial \mathcal{L}_{\text{SIN}}}{\partial \boldsymbol{\phi}_{\text{NN}}} \approx N\left\{\frac{\partial}{\partial \mathbf{x}_n^*}\left[\log \frac{p(\mathbf{y}_n|\mathbf{x}_n^*,\boldsymbol{\theta}_{\text{NN}})}{q(\mathbf{x}_n^*|f_{\phi_{\text{NN}}}(\mathbf{y}_n))}\right]\frac{\partial \mathbf{x}_n^*}{\partial \boldsymbol{\phi}_{\text{NN}}} - \frac{\partial \log q(\mathbf{x}_n^*|f_{\phi_{\text{NN}}}(\mathbf{y}_n))}{\partial \boldsymbol{\phi}_{\text{NN}}}\right\} + \frac{\partial \log \mathcal{Z}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}_{\text{NN}}}, \tag{9}$$

$$\frac{\partial \mathcal{L}_{\text{SIN}}}{\partial \boldsymbol{\phi}_{\text{PGM}}} \approx \frac{\partial}{\partial \mathbf{x}^*}\left[\log \frac{p(\mathbf{x}^*|\boldsymbol{\theta}_{\text{PGM}})}{q(\mathbf{x}^*|\boldsymbol{\phi}_{\text{PGM}})}\right]\frac{\partial \mathbf{x}^*}{\partial \boldsymbol{\phi}_{\text{PGM}}} + \frac{\partial \log \mathcal{Z}(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}_{\text{PGM}}}, \tag{10}$$

where we drop the explicit dependence of $\mathbf{x}^*(\boldsymbol{\phi})$ over $\boldsymbol{\phi}$. In the first line, the gradient with respect to θ_{NN} is identical to the back-propagation-step through the decoder in VAE, while the gradient with respect to $\boldsymbol{\theta}_{\text{PGM}}$ is the gradient of log-likelihood of PGM. Both quantities can be easily computed. In the second line, the gradient with respect to $\boldsymbol{\phi}_{\text{NN}}$ is identical to the gradient of the encoder in VAE, plus the gradient of $\mathcal{Z}(\boldsymbol{\phi})$. The third line shows the gradient with respect to $\boldsymbol{\phi}_{\text{PGM}}$, which is equal to the gradient through the log-likelihoods of the PGMs, plus the gradient of $\mathcal{Z}(\boldsymbol{\phi})$. Therefore, given that $\mathbf{x}^*$ can be drawn easily and gradient of $\mathcal{Z}(\boldsymbol{\phi})$ can be computed, we can compute all gradients easily. Not only that, we can also reuse the implementation for inference in VAE and PGMs.

When $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$ and $q(\mathbf{x}|\boldsymbol{\phi}_{\text{PGM}})$ factorize into simple factors, which is expected since both are structured graphical models, we can further reduce computation by computing stochastic approximations of their gradients. The only tricky part is the computation of $\mathcal{Z}(\boldsymbol{\phi})$ which might require a

full inference through the graphical model. If the graphical model is too complex (e.g., Gaussian process prior), then this might be difficult since an exact Bayesian inference in such models is not computationally efficient. In this paper, we therefore only focus on PGMs where $\mathcal{Z}(\boldsymbol{\phi})$ is tractable.

We now give many examples and illustrate the choices of the two factors for a variety of PGM priors.

When $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$ is a conjugate exponential-family distribution, choosing the two factors is a very easy task. In this case, we can let $q(\mathbf{x}|\boldsymbol{\phi}_{\text{PGM}}) = p(\mathbf{x}|\boldsymbol{\phi}_{\text{PGM}})$, i.e., the second factor is the same distribution as the PGM prior but with different set of parameters $\boldsymbol{\phi}_{\text{PGM}}$. To illustrate this, we give an example below when the PGM prior is a linear dynamical system.

Example (SIN for Linear Dynamical System (LDS)) : When $\mathbf{y}_n$ is a time series, we can model the latent $\mathbf{x}_n$ using an LDS. $p(\mathbf{x}|\boldsymbol{\theta}) := \mathcal{N}(\mathbf{x}_0|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{A}\mathbf{x}_{n-1}, \mathbf{Q})$, where $\mathbf{A}$ is the transition matrix, $\mathbf{Q}$ is the covariance for the process noise, and $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ are the mean and covariance of the initial distribution. Therefore, $\boldsymbol{\theta}_{\text{PGM}} := \{\mathbf{A}, \mathbf{Q}, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\}$. In our inference network, we choose $q(\mathbf{x}|\boldsymbol{\phi}) = p(\mathbf{x}|\boldsymbol{\phi})$ as show below, where $\boldsymbol{\phi}_{\text{PGM}} := \{\bar{\mathbf{A}}, \bar{\mathbf{Q}}, \bar{\boldsymbol{\mu}}_0, \bar{\boldsymbol{\Sigma}}_0\}$ and, since the PGM is a Gaussian, we choose the DNN factor to be a Gaussian as well:

$$q(\mathbf{x}|\mathbf{y}, \boldsymbol{\phi}) := \frac{1}{\mathcal{Z}(\boldsymbol{\phi})} \underbrace{\left[\prod_{n=1}^{N} \mathcal{N}\left(\mathbf{x}_n|\mathbf{m}_n, \mathbf{V}_n\right)\right]}_{\text{DNN Factor}} \underbrace{\left[\mathcal{N}(\mathbf{x}_0|\bar{\boldsymbol{\mu}}_0, \bar{\boldsymbol{\Sigma}}_0) \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\bar{\mathbf{A}}\mathbf{x}_{n-1}, \bar{\mathbf{Q}})\right]}_{\text{LDS Factor}}, \quad (11)$$

where $\mathbf{m}_n := \mathbf{m}_{\phi_{\text{NN}}}(\mathbf{y}_n)$ and $\mathbf{V}_n := \mathbf{V}_{\phi_{\text{NN}}}(\mathbf{y}_n)$ are mean and covariance parameterized by a DNN with parameter $\boldsymbol{\phi}_{\text{NN}}$. The graphical model and SIN are shown in Figure 1c. The above SIN is a conjugate model where the marginal likelihood and distributions can be computed in $O(N)$ using the forward-backward algorithm, a.k.a. Kalman smoother (Bishop, 2006). We can also compute the gradient of $\mathcal{Z}(\boldsymbol{\phi})$ as shown in (Kokkala et al., 2015).

When the PGM prior have additional structure and latent variables, e.g., the GMM prior has cluster indicators z_n, we might want to incorporate this into the network, as shown in the example below.

Example (SIN for GMM prior): The prior shown in (2) has additional structure where we marginalize over the indicator variable z_n. To mimic this structure in SIN, we choose the PGM factor as shown below with parameters $\boldsymbol{\phi}_{\text{PGM}} := \{\bar{\boldsymbol{\mu}}_k, \bar{\boldsymbol{\Sigma}}_k, \bar{\pi}_k\}_{k=1}^{K}$, while keeping the DNN part to be a Gaussian distribution similar to the LDS case.

$$q(\mathbf{x}|\mathbf{y}, \boldsymbol{\phi}) := \frac{1}{\mathcal{Z}(\boldsymbol{\phi})} \prod_{n=1}^{N} \underbrace{\left[\mathcal{N}\left(\mathbf{x}_n|\mathbf{m}_n, \mathbf{V}_n\right)\right]}_{\text{DNN Factor}} \underbrace{\left[\sum_{k=1}^{K} \mathcal{N}(\mathbf{x}_n|\bar{\boldsymbol{\mu}}_k, \bar{\boldsymbol{\Sigma}}_k)\bar{\pi}_k\right]}_{\text{GMM Factor}}, \quad (12)$$

The model and SIN are shown in Figure 1a. Fortunately, due to conjugacy of the Gaussian and Multinomial distribution, we can marginalize $\mathbf{x}_n$ to get a closed-form expression for $\log \mathcal{Z}(\boldsymbol{\phi}) := \sum_n \log \sum_k \mathcal{N}(\mathbf{m}_n|\bar{\boldsymbol{\mu}}_k, \mathbf{V}_n + \bar{\boldsymbol{\Sigma}}_k)\ \bar{\pi}_k$. We an also easily sample by first sampling from the marginal distribution of z_n given by $q(z_n = k|\mathbf{y}, \boldsymbol{\phi}) \propto \mathcal{N}\left(\mathbf{m}_n|\bar{\boldsymbol{\mu}}_k, \mathbf{V}_n + \bar{\boldsymbol{\Sigma}}_k\right)\ \bar{\pi}_k$. Given z_n, we can easily sample $\mathbf{x}_n$ from the following marginal: $q(\mathbf{x}_n|z_n = k, \mathbf{y}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{x}_n|\widetilde{\boldsymbol{\mu}}_n, \widetilde{\boldsymbol{\Sigma}}_n)$, where $\widetilde{\boldsymbol{\Sigma}}_n^{-1} = \mathbf{V}_n^{-1} + \bar{\boldsymbol{\Sigma}}_k^{-1}$ and $\widetilde{\boldsymbol{\mu}}_n = \widetilde{\boldsymbol{\Sigma}}_n(\mathbf{V}_n^{-1}\mathbf{m}_n + \bar{\boldsymbol{\Sigma}}_k^{-1}\boldsymbol{\mu}_k)$.

In all of the above examples, we are able to satisfy the two conditions even when using the same structure as the model. However, this may not be possible for general conditionally-conjugate exponential family distribution. In such cases, we can still obtain samples from a tractable structured mean-field approximation using VMP. We illustrate this on the switching state-space model in Appendix B. A drawback of our method in this case is that we need to run VMP long enough to get a sample, very similar to the method of Johnson et al. (2016). The advantage, however, is that this process is completely independent of $\boldsymbol{\theta}$, therefore we do not have to compute gradient of $\mathcal{Z}(\boldsymbol{\phi})$ with respect to $\boldsymbol{\theta}$, which makes the implementation easier. We also expect the convergence to not be a big problem because, even if we stop early, we are simply generating samples from an approximate posterior. In addition, we can handle non-conjugate factors as we show in our next example.

When the PGM prior contains some non-conjugate factors, we might replace them by their closest conjugate approximations while making sure that the inference network captures the useful structure present in the posterior distribution. We illustrate this point for a Student's t mixture model.

> **Example (SIN for Student's t Mixture Model) :** To handle outliers in the data, we might want to use the Student-t mixture component in the mixture model shown in (2), i.e., we set $p(\mathbf{x}_n|z_n = k) = \mathcal{T}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k, \gamma_k)$ with mean $\boldsymbol{\mu}_k$, scale matrix $\boldsymbol{\Sigma}_k$ and degree of freedom $\boldsymbol{\gamma}_k$. The Student's t-distribution is not conjugate to the multinomial distribution, therefore, if we use it as the PGM factor in SIN, we will not be able to satisfy both conditions easily. Even though our model contains a t-distribution components, we can still use the SIN shown in (12) that uses a GMM factor. Our method therefore simplifies inference by choosing an inference network which has a simpler form than the original model.

In theory, one can do this even when all factors are non-conjugate, however, the approximation error might be quite large in some cases for this approximation to be useful. We experimented this for non-linear dynamical system and found that capturing non-linearity was essential for extremely non-linear dynamics.

4 Variational Message Passing with Inference Networks

In the previous section, we introduced the structured inference network and described how to compute the gradient of the lower bound. We will now show that when $\boldsymbol{\theta}_{\text{PGM}}$ is a random variable with a prior distribution, we can use VMP udpates to estimate its variational approximation, while continuing to use stochastic-gradient updates for the rest of the parameters. In this section, we assume an exponential-family prior $p(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\eta}_{\text{PGM}})$ with natural parameter $\boldsymbol{\eta}_{\text{PGM}}$. Our goal is to approximate the posterior distribution over $\mathbf{x}$ and $\boldsymbol{\theta}_{\text{PGM}}$, and compute point estimates of $\boldsymbol{\theta}_{\text{NN}}$, $\boldsymbol{\phi}_{\text{NN}}$, and $\boldsymbol{\phi}_{\text{PGM}}$.

Similar to Johnson et al. (2016), we want to perform natural-gradient updates for $\boldsymbol{\theta}_{\text{PGM}}$. For this reason, we will use the method of Khan & Lin (2017). This method is a generalization of VMP and SVI, and enables us to derive natural-gradient updates for $\boldsymbol{\theta}_{\text{PGM}}$. The method uses a mirror-descent algorithm in the mean-parameter space of the exponential family. When the Kullback-Leibler (KL) divergence is used for the mirror-descent step, the step is equivalent to the natural-gradient descent in the natural-parameter space. Khan & Lin (2017) show that the mirror-descent update is equivalent to the SVI update when all the neighbours of a node are conjugate. In our setup, their method is useful since it enables natural-gradient updates even when the PGM part is not conjugate.

We start by deriving the variational lower bound. We use the following mean-field variational approximation: $p(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y}) \approx q(\mathbf{x}|\mathbf{y}, \boldsymbol{\phi})q(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\lambda}_{\text{PGM}})$ where the first term is SIN introduced in the previous section, and the second term is an exponential-family distribution with natural parameter $\boldsymbol{\lambda}_{\text{PGM}}$. The variational lower bound corresponding to the mean-field approximation can be expressed in terms of $\mathcal{L}_{\text{SIN}}$ derived in the previous section.

$$\mathcal{L}(\boldsymbol{\lambda}_{\text{PGM}}, \boldsymbol{\phi}_{\text{PGM}}, \boldsymbol{\theta}_{\text{NN}}, \boldsymbol{\phi}_{\text{NN}}) := \mathbb{E}_{q(\theta_{\text{PGM}}|\lambda_{\text{PGM}})}\left[\mathcal{L}_{\text{SIN}}(\boldsymbol{\theta}, \boldsymbol{\phi})\right] - \mathbb{D}_{KL}[q(\boldsymbol{\theta}_{\text{PGM}}) \,\|\, p(\boldsymbol{\theta}_{\text{PGM}})]. \tag{13}$$

We will use a mirror-descent update with the KL divergence for $q(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\lambda}_{\text{PGM}})$ because we want natural-gradient updates for it. We denote the mean parameter of this distribution by $\boldsymbol{\mu}_{\text{PGM}}$. For the rest of the parameters, we will use the usual Euclidean distance. Denoting the values at iteration t with a superscipt t and using Eq. 19 in (Khan & Lin, 2017) with these divergences, we get:

$$\max_{\mu_{\text{PGM}}} \; \langle \boldsymbol{\mu}_{\text{PGM}}, \nabla_{\mu_{\text{PGM}}}\mathcal{L}_t\rangle - \frac{1}{\beta_1}\mathbb{D}_{KL}[q(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\mu}_{\text{PGM}}) \,\|\, q(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\mu}^t_{\text{PGM}})] \tag{14}$$

$$\max_{\theta_{\text{NN}}} \; \langle \boldsymbol{\theta}_{\text{NN}}, \nabla_{\theta_{\text{NN}}}\mathcal{L}_t\rangle - \frac{1}{\beta_2}\|\boldsymbol{\theta}_{\text{NN}} - \boldsymbol{\theta}^t_{\text{NN}}\|^2. \tag{15}$$

$$\max_{\phi_{\text{PGM}}, \phi_{\text{NN}}} \; \langle \boldsymbol{\phi}_{\text{PGM}}, \nabla_{\phi_{\text{PGM}}}\mathcal{L}_t\rangle + \langle \boldsymbol{\phi}_{\text{NN}}, \nabla_{\phi_{\text{NN}}}\mathcal{L}_t\rangle - \frac{1}{\beta_3}\|\boldsymbol{\phi}_{\text{PGM}} - \boldsymbol{\phi}^t_{\text{PGM}}\|^2 - \frac{1}{\beta_4}\|\boldsymbol{\phi}_{\text{NN}} - \boldsymbol{\phi}^t_{\text{NN}}\|^2. \tag{16}$$

where β_1 to β_4 are scalars, $\langle, \rangle$ is an inner product, and $\nabla\mathcal{L}_t$ is the gradient at the value in iteration t.

As shown in Khan & Lin (2017), the maximization for $\boldsymbol{\lambda}_{\text{PGM}}$ can be obtained in closed-form:

$$\boldsymbol{\lambda}_{\text{PGM}} \leftarrow (1 - \beta_1)\boldsymbol{\lambda}_{\text{PGM}} + \beta_1 \nabla_{\mu_{\text{PGM}}}\mathbb{E}_{q(\theta_{\text{PGM}}|\mu_{\text{PGM}})}[\log p(\mathbf{x}^*|\boldsymbol{\theta}_{\text{PGM}})]. \tag{17}$$

Algorithm 1 Update for our SAN algorithm

Require: Data $\mathbf{y}$, Step-sizes $\beta_1, \beta_2, \beta_3, \beta_4$
1: Initialize $\boldsymbol{\lambda}_{\text{PGM}}, \boldsymbol{\phi}_{\text{PGM}}, \boldsymbol{\theta}_{\text{NN}}, \boldsymbol{\phi}_{\text{NN}}$.
2: **repeat**
3: Compute the posterior $q(\mathbf{x}|\mathbf{y}, \boldsymbol{\phi})$ for SIN of (5) either using an exact expression or VMP.
4: Sample $\mathbf{x}^* \sim q(\mathbf{x}|\mathbf{y}, \boldsymbol{\phi})$, and compute $\nabla_\phi \mathcal{Z}$ and $\nabla_\phi \mathbf{x}^*$.
5: Update $\boldsymbol{\lambda}_{\text{PGM}}$ using the natural-gradient step given in (17).
6: Update $\boldsymbol{\phi}_{\text{PGM}}$ using the gradient given in (10).
7: Update $\boldsymbol{\theta}_{\text{NN}}$ and $\boldsymbol{\phi}_{\text{NN}}$ using the gradient given in (8)-(9) with $\boldsymbol{\theta}_{\text{PGM}} \sim q(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\lambda}_{\text{PGM}})$.
8: **until** Convergence

This update is a natural-gradient update since the gradient with respect to $\boldsymbol{\mu}_{\text{PGM}}$ is the natural gradient. When the prior on $\boldsymbol{\theta}_{\text{PGM}}$ is conjugate to $p(\mathbf{x}|\boldsymbol{\theta}_{\text{PGM}})$, this gradient can be computed using the VMP or SVI implementation. Otherwise, it can be approximated either using stochastic gradients or using the reparameterization trick, as discussed in (Khan & Lin, 2017).

It is easy to see that the update of the rest of the parameters is equal to a stochastic-gradient descent update. We can verify this by simply taking the derivative of (15) or (16), and setting it to zero. The gradient $\nabla \mathcal{L}_t$ can be obtained by using a stochastic approximation by first sampling $\boldsymbol{\theta}_{\text{PGM}} \sim q(\boldsymbol{\theta}_{\text{PGM}}|\boldsymbol{\lambda}_{\text{PGM}})$, and then approximating the gradient as follows: $\nabla \mathcal{L}(\boldsymbol{\lambda}_{\text{PGM}}, \boldsymbol{\phi}_{\text{PGM}}, \boldsymbol{\theta}_{\text{NN}}, \boldsymbol{\phi}_{\text{NN}}) \approx \nabla \mathcal{L}_{\text{SIN}}(\boldsymbol{\theta}, \boldsymbol{\phi})$. Therefore, the update can be obtained using the gradients given in (8)-(10). In practice, we use adaptive-gradient methods to update (instead of using SGD).

The final algorithm is outlined in Algorithm 1. Since our algorithm enables Structured, Amortized, and Natural-gradient (SAN) updates, we call it the SAN algorithm. Our updates conveniently separate the PGM and DNN computations. Step 3-6 operate on the PGM part, for which we can use existing implementation for the PGM. Step 7 operates on the DNN part, for which we can reuse VAE implementation. Our algorithm not only generalizes previous works, but also simplifies the implementation by enabling reusing existing software.

5 Experiments and Results

The main goal of our experiment is to show that our SAN algorithm gives similar results to the method of Johnson et al. (2016). The advantage of our algorithm over the method of Johnson et al. (2016) is that our algorithm is simpler and more general than their method, and does not involve a double-loop procedure. For this reason, we apply our algorithm to the two examples considered in Johnson et al. (2016), namely the latent GMM and latent LDS. In this section, we show results on GMM. An additional result for LDS is included in Appendix A. Our results show that the SAN algorithm can learn complex representations with interpretable structure, similar to the method of Johnson et al. (2016).

For latent GMM, we compare SAN to three algorithms: first, the standard variational EM algorithm applied to GMM (referred to as 'GMM'), second, the VAE approach of Kingma & Welling (2013), and third, the SVAE approach of Johnson et al. (2016). We use two datasets. The first dataset is the synthetic two-dimensional Pinwheel dataset ($N = 5000$ and $D = 2$), and the second dataset is the Auto dataset ($N = 392$ and $D = 6$, available in the UCI repository). The latter contains information about cars. The dataset also contains a five-class label which indicates the number of cylinders in a car. We use these labels to validate our results. For both datasets we use 70% data for training.

For all methods, we tune their stepsizes, the number of mixture components, and the latent dimensionality on a validation set. We train the GMM baseline on the full dataset, and, for VAE and SVAE, we use minibatches of size 64. Our neural-network likelihood consists of two layers with 50 hidden units and an output layer of dimensionality 6 for the Auto dataset and 2 for the Pinwheel dataset.

In Figure 2, we compare the performance during training. In the left figure, we compare to SVAE and GMM, where we see that SAN converges faster than SVAE. Both methods achieve similar performance upon convergence and perform better than GMM. In the right figure, we compare to VAE and observe similar trends. In this figure, the performance of GMM is represented as a constant because training converges after a few iterations already. We found that the implementation

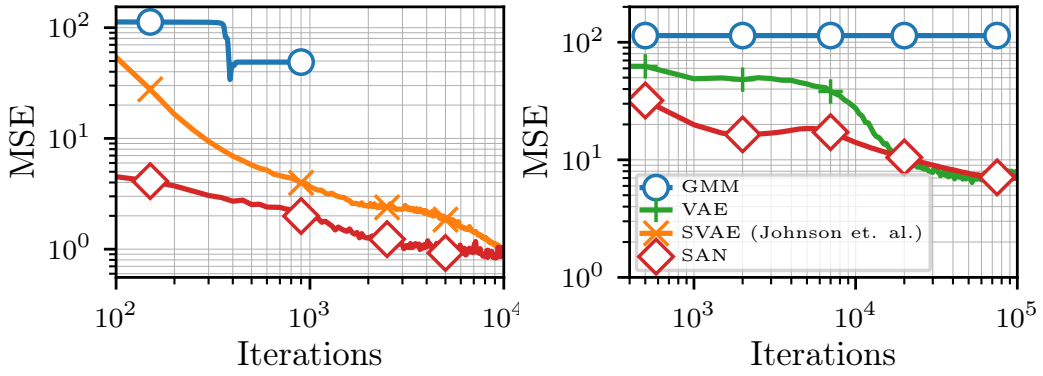
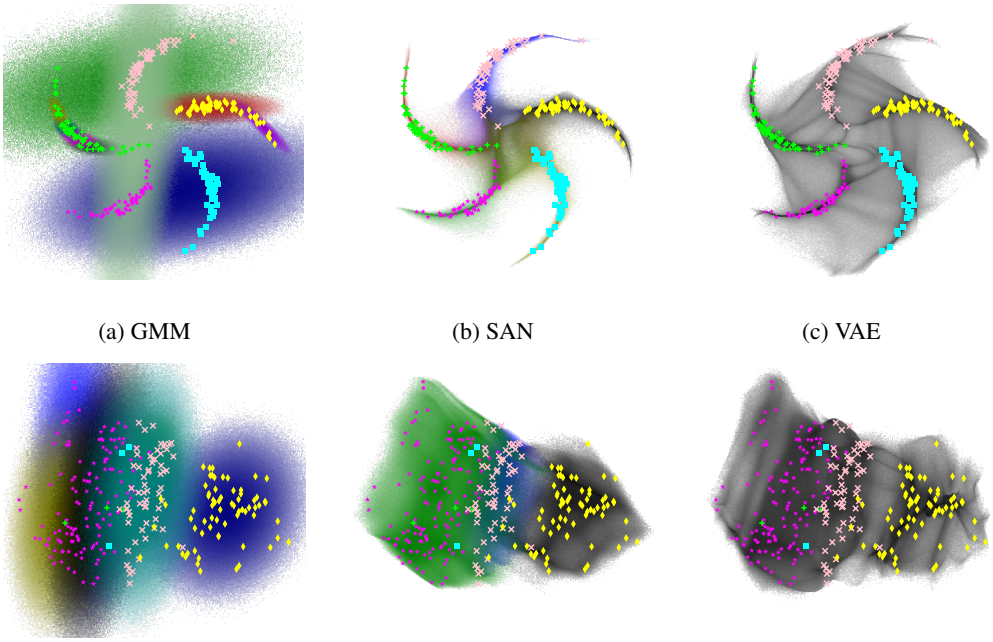


Figure 2: Reconstruction mean-squared error (MSE) computed on the Pinwheel dataset (left) and the Auto dataset (right). The performance has been measured every 500 iterations but the markers are shown only at iteration 500, 2000, 7000 and 75000. In the left figure, we compare to SVAE and GMM, where we see that SAN converges faster than SVAE, although both methods achieve similar error but perform better than GMM. In the right figure, we compare to VAE, where we see very similar trends. The performance of the GMM is represented as a constant because training converges after a few iterations already.



(a) GMM (b) SAN (c) VAE

(d) GMM (e) SAN (f) VAE

Figure 3: These plots show the learned representations $p(\mathbf{y})$ in the background of the Pinwheel dataset (3a, 3b, 3c) and the Auto dataset (3d, 3e, 3f). In the foreground, we show 300 data samples which are colored according to the true labels (not used during the training). Different colors signify a mixture component and color intensities indicate the density. We use $K = 10$ mixture components to train all models. For the Auto dataset, we show only the first two principle components.

provided by Johnson et al. (2016) does not perform well on the Auto dataset which is why we have not included it in the comparison. We also compared test log-likelihood and imputation error which show very similar trends.

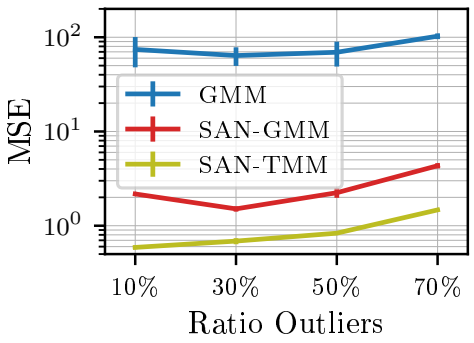


Figure 4: Reconstruction mean-squared error (MSE) computed on the Pinwheel dataset with an increasing number of outliers. We show the mean-MSE over three runs with the corresponding standard deviation. We compare the non-conjugate latent TMM against GMM and latent GMM. We see that the latent TMM clearly outperforms the other models for every noise level. Also, with increasing noise level, the performance drop for the latent TMM is less dramatic compared to the other models (mind that the y-axis is logarithmic).

In Figure 3, we show samples generated from the learned distribution $p(\mathbf{y})$ for all methods. On the top of this plot, we show the data with true labels. These labels were not used during training. The plots (a)-(c) show results on Pinwheel dataset, while plots (d)-(e) shows results on the Auto dataset. For the Auto dataset, each label corresponds to the number of cylinders present in a car. We observe that SAN can learn meaningful partition of the data space, while VAE does not posses this capability. We also observe that SAN and VAE can learn flexible patterns while GMM does not do so.

These results show that with SAN we are able to combine complementary characteristics of VAE and GMM. For both datasets, our method learns flexible representations containing meaningful structural information.

We also consider a non-conjugate model, the latent Student's t mixture model (latent TMM) discussed in Section 3. The goal of this experiment is to show that, unlike the method of Johnson et al. (2016), our method works even when the model contains non-conjugate factors. The Student's t-distribution is very similar to the Gaussian distribution but is more robust to outliers thanks to its heavy tails. To show the latent TMM's advantage over the latent GMM, we add artificial outliers to the Pinwheel dataset by introducing random Gaussian noise. We fix the degree of freedom for the Student's t-distribution to 5 and use the structured inference network as explained in Section 3. We test our model on four noisy data sets with different levels of noise and report the test MSE averaged over three runs in Figure 4. We compare to the GMM baseline, as well as to SAN algorithm applied to latent GMM (instead of latent TMM). In Figure 4, we can see that latent TMM is more robust than latent GMM and GMM. The performance of each of the three tested models degrades with an increasing number of outliers, but the drop of the latent TMM is far less dramatic (mind the log-scale of the y-axis). In fact, with 70% of outliers, the latent TMM still performs better than the latent GMM with only 10% of outliers. This experiment illustrates that a conjugate SIN can be used for inference on a model with a nonconjugate PGM.

6 Discussion and Conclusion

We propose a message-passing algorithm for the models that use both deep networks and probabilistic graphical models. We discussed the challenges involved when designing an algorithm to exploit the structural properties. Our algorithm can automatically adjust its messages and adapt it to the type of the node. For the DNN part, the messages reduce to computation of stochastic gradients, while on the PGM part, the messages are equal to natural gradients. Experimental comparisons confirm that our method gives similar results to existing methods, while using simpler updates.

Our experimental results in this paper are limited to small scale data. We found that it is non-trivial to implement a message-passing framework that goes well with the deep learning framework. Our current implementation is based on AutoGrad and TensorFlow, both of which are slow due to additional communication overhead due to messages passing. We are going to pursue this direction in the future and investigate good platforms to integrate the capabilities of these two different flavors of algorithms.

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.

Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*, pp. 2199–2207, 2016.

Zoubin Ghahramani and Geoffrey E Hinton. Variational learning for switching state-space models. *Neural computation*, 12(4):831–864, 2000.

Matthias Heinkenschloss. Numerical solution of implicitly constrained optimization problems. Technical report, Rice University, Department of Computational and Applied Mathematics, 2008.

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in Neural Information Processing Systems*, pp. 2946–2954, 2016.

Mohammad Emtiyaz Khan and Wu Lin. Conjugate-computation variational inference: Converting variational inference in non-conjugate models to inferences in conjugate models. *arXiv preprint arXiv:1703.04265*, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Juho Kokkala, Arno Solin, and Simo Särkkä. Sigma-point filtering and smoothing based parameter estimation in nonlinear dynamic systems. *arXiv preprint arXiv:1504.06173*, 2015.

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

T. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2001.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

John Winn and Christopher M Bishop. Variational message passing. *Journal of Machine Learning Research*, 6(Apr):661–694, 2005.

A RESULTS FOR LATENT LINEAR DYNAMICAL SYSTEM

In this experiment, we apply our SAN algorithm to the latent LDS discussed in Section 3. For comparison, we compare our method, Structured Variational Auto-Encoder (SVAE) (Johnson et al., 2016), and LDS on the Dot dataset used in Johnson et al. (2016). Our results show that our method achieves comparable performance to SVAE. For LDS, we perform batch learning for all model parameters using the EM algorithm. For SVAE and SAN, we perform mini-batch updates for all model parameters. We use the same neutral network architecture as in Johnson et al. (2016), which contains two hidden layers with tanh activation function. We repeat our experiments 10 times and measure model performance in terms of the following mean absolute error for τ-steps ahead prediction. The error measures the absolute difference between the ground truth and the generative outputs by averaging across generated results.

$$\sum_{n=1}^{N}\sum_{t=1}^{T-\tau}\frac{1}{N(T-\tau)d}\left\{||\mathbf{y}^*_{t+\tau,n}-\mathbb{E}_{p(y_{t+\tau,n}|y_{1:t,n})}\left[\mathbf{y}_{t+\tau,n}\right]||_1\right\} \tag{18}$$

where N is the number of testing time series with T time steps, d is the dimensionality of observation $\mathbf{y}$, and observation $\mathbf{y}^*_{t+\tau,n}$ denotes the ground-truth at time step $t+\tau$.

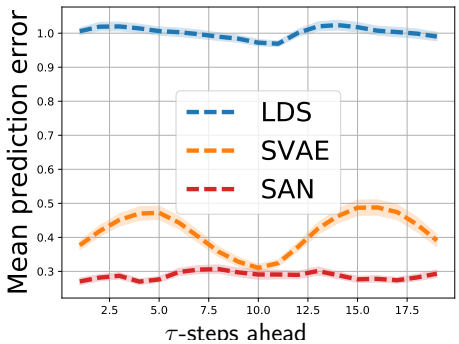


Figure 5: Prediction error, where shadows denote the standard errors across 10 runs

From Figure 5, we can observe that our method performs as good as SVAE and outperforms LDS. Our method is slightly robust than SVAE. In Figure 6, there are generated images obtained from all methods. From Figure 6, we also see that our method performs as good as SAVE and is able to recover the ground-truth observation.

B SIN FOR SWITCHING LDS

In SLDS, we introduce discrete variable $z_n \in \{1, 2, \ldots, K\}$ that are samples using a Markov model: $p(z_n = i|z_{n-1} = j) = \pi_{ij}$ such that π_{ij} sum to 1 over all i given j. The transition for LDS is defined conditional on z_n: $p(\mathbf{x}_n|\mathbf{x}_{n-1}, z_n = i, \boldsymbol{\theta}_{\text{PGM}}) := \mathcal{N}(\mathbf{x}_n|\mathbf{A}_i\mathbf{x}_{n-1}, \mathbf{Q}_i)$ where $\mathbf{A}_i$ and $\mathbf{Q}_i$ are parameters for the i'th indicator. These two dynamics put together define the SLDS prior $p(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}_{\text{PGM}})$. We can use the following SIN which uses the SLDS prior as the PGM factor but with parameters $\boldsymbol{\phi}_{\text{PGM}}$ instead of $\boldsymbol{\theta}_{\text{PGM}}$. The expression for $q(\mathbf{x}, \mathbf{z}|\mathbf{y}, \boldsymbol{\phi})$ is shown below:

$$\frac{1}{\mathcal{Z}(\boldsymbol{\phi})}\underbrace{\left[\prod_{n=1}^{N}\mathcal{N}\left(\mathbf{x}_n|\mathbf{m}_n, \mathbf{V}_n\right)\right]}_{\text{DNN Factor}}\underbrace{\left[\mathcal{N}(\mathbf{x}_0|\bar{\boldsymbol{\mu}}_{0,z_0}, \bar{\boldsymbol{\Sigma}}_{0,z_0})\prod_{n=1}^{N}\mathcal{N}(\mathbf{x}_n|\bar{\mathbf{A}}_{z_n}\mathbf{x}_{n-1}, \bar{\mathbf{Q}}_{z_n})p(z_n|z_{n-1})\right]}_{\text{SLDS Factor}},$$

Even though the above model is a conditionally-conjugate model, the partition function is not tractable and sampling is also not possible. However, we can use a structured mean-field approximation. First, we can combine the DNN factor with the Gaussian observation of SLDS factor and then use a mean-field approximation $q(\mathbf{x}, \mathbf{z}|\mathbf{y}, \boldsymbol{\phi}) \approx q(\mathbf{x}|\boldsymbol{\lambda}_x)q(\mathbf{z}|\boldsymbol{\lambda}_z)$, e.g., using the method of Ghahramani & Hinton (2000). This will give us a structured approximation where the edges between $\mathbf{y}_n$ and $\mathbf{x}_n$ and z_n and z_{n-1} are maintained but $\mathbf{x}_n$ and z_n independent of each other.

(a) SAN (b) SVAE (c) LDS (d) Ground Truth

Figure 6: Generated images, where each column of pixels represents an observation, each row of pixels represents one time step, and each vertical white line denotes the first time step to generate images