Efficient Gradient-Based Inference through Transformations between Bayes Nets and Neural Nets

Kingma, D.P.; Welling, M.

Published in:
JMLR Workshop and Conference Proceedings

Citation for published version (APA):
Hierarchical Bayesian networks and neural networks with stochastic hidden units are commonly perceived as two separate types of models. We show that either of these types of models can often be transformed into an instance of the other, by switching between centered and differentiable non-centered parameterizations of the latent variables. The choice of parameterization greatly influences the efficiency of gradient-based posterior inference; we show that they are often complementary to each other, we clarify when each parameterization is preferred and show how inference can be made robust. In the non-centered form, a simple Monte Carlo estimator of the marginal likelihood can be used for learning the parameters. Theoretical results are supported by experiments.

1. Introduction

Bayesian networks (also called belief networks) are probabilistic graphical models where the conditional dependencies within a set of random variables are described by a directed acyclic graph (DAG). Many supervised and unsupervised models can be considered as special cases of Bayesian networks.

In this paper we focus on the problem of efficient inference in Bayesian networks with multiple layers of continuous latent variables, where exact posterior inference is intractable (e.g. the conditional dependencies between variables are nonlinear) but the joint distribution is differentiable. Algorithms for approximate inference in Bayesian networks can be roughly divided into two categories: sampling approaches and parametric approaches. Parametric approaches include Belief Propagation (Pearl 1982) or the more recent Expectation Propagation (EP) (Minka 2001). When it is not reasonable or possible to make assumptions about the posterior (which is often the case), one needs to resort to sampling approaches such as Markov Chain Monte Carlo (MCMC) (Neal 1993). In high-dimensional spaces, gradient-based samplers such as Hybrid Monte Carlo (Duane et al. 1987) and the recently proposed no-U-turn sampler (Hoffman & Gelman 2011) are known for their relatively fast mixing properties. When just interested in finding a mode of the posterior, vanilla gradient-based optimization methods can be used. The alternative parameterizations suggested in this paper can dramatically improve the efficiency of any of these algorithms.

1.1. Outline of the paper

After reviewing background material in 2 we introduce a generally applicable differentiable reparameterization of continuous latent variables into a differentiable non-centered form in section 3. In section 4 we analyze the posterior dependencies in this reparameterized form. Experimental results are shown in section 6.

2. Background

Notation. We use bold lower case (e.g. \( \mathbf{x} \) or \( \mathbf{y} \)) notation for random variables and instantiations (values) of random variables. We write \( p_{\theta}(\mathbf{x}| \mathbf{y}) \) and \( p_{\theta}(\mathbf{x}) \) to denote (conditional) probability density (PDF) or mass (PMF) functions of variables. With \( \theta \) we denote the vector containing all parameters; each distribution in the network uses a subset of \( \theta \)'s elements. Sets of variables are capitalized and bold, matrices are capitalized and bold, and vectors are written in bold and lower case.

2.1. Bayesian networks

A Bayesian network models a set of random variables \( V \) and their conditional dependencies as a directed acyclic graph, where each variable corresponds to a vertex and each edge to a conditional dependency. Let the distribu-
2.2. Conditionally deterministic variables

A conditionally deterministic variable $v_j$ with parents $pa_j$ is a variable whose value is a (possibly nonlinear) deterministic function $g_j(\cdot)$ of the parents and the parameters: $v_j = g_j(pa_j, \theta)$. The PDF of a conditionally deterministic variable is a Dirac delta function, which we define as a Gaussian PDF $\mathcal{N}(\cdot; \mu, \sigma)$ with infinitesimal $\sigma$:

$$p_\theta(v_j|pa_j) = \lim_{\sigma \to 0} \mathcal{N}(v_j; g_j(pa_j, \theta), \sigma)$$  \hspace{1cm} (2)

which equals $+\infty$ when $v_j = g_j(pa_j, \theta)$ and equals 0 everywhere else such that $\int_{v_j} p_\theta(v_j|pa_j) \, dv_j = 1$.

2.3. Inference problem under consideration

We are often interested in performing posterior inference, which most frequently consists of either optimization (finding a mode $\arg\max_z p_\theta(z|x)$) or sampling from the posterior $p_\theta(z|x)$. Gradients of the log-posterior w.r.t. the latent variables can be easily acquired using the equality:

$$\nabla_z \log p_\theta(z|x) = \nabla_z \log p_\theta(x, z)$$
$$= \sum_{j=1}^N \nabla_z \log p_\theta(v_j|pa_j)$$  \hspace{1cm} (3)

In words, the gradient of the log-posterior w.r.t. the latent variables is simply the sum of gradients of individual factors w.r.t. the latent variables. These gradients can then be followed to a mode if one is interested in finding a MAP solution. If one is interested in sampling from the posterior, then the gradients can be plugged into a gradient-based sampler such as Hybrid Monte Carlo (Duane et al. 1987); if also interested in learning parameters, the resulting samples can be used for the E-step in Monte Carlo EM (Wei & Tanner 1990) (MCEM).

Problems arise when strong posterior dependencies exist between latent variables. From eq. (3) we can see that the Hessian $H$ of the posterior is:

$$H = \nabla_z \nabla_z^T \log p_\theta(z|x) = \sum_{j=1}^N \nabla_z \nabla_z^T \log p_\theta(v_j|pa_j)$$  \hspace{1cm} (4)

Suppose a factor $\log p_\theta(z_i|z_j)$ connecting two scalar latent variables $z_i$ and $z_j$ exists, and $z_j$ is strongly dependent on $z_j$, then the Hessian’s corresponding element $\frac{\partial^2 \log p_\theta(z_i|z_j)}{\partial z_i \partial z_j}$ will have a large (positive or negative) value. This is bad for gradient-based inference since it means that changes in $z_j$ have a large effect on the gradient $\frac{\partial \log p_\theta(z_i|z_j)}{\partial z_i}$ and changes in $z_i$ have a large effect on the gradient $\frac{\partial \log p_\theta(z_i|z_j)}{\partial z_j}$. In general, strong conditional dependencies lead to ill-conditioning of the posterior, resulting in smaller optimal stepsizes for first-order gradient-based optimization or sampling methods, making inference less efficient.

3. The differentiable non-centered parameterization (DNCP)

In this section we introduce a generally applicable transformation between continuous latent random variables and deterministic units with auxiliary parent variables. In rest of the paper we analyze its ramifications for gradient-based inference.

3.1. Parameterizations of latent variables

Let $z_j$ be some continuous latent variable with parents $pa_j$, and corresponding conditional PDF:

$$z_j|pa_j \sim p_\theta(z_j|pa_j)$$  \hspace{1cm} (5)

This is also known in the statistics literature as the centered parameterization (CP) of the latent variable $z_j$. Let the

![Figure 1. (a) The centered parameterization (CP) of a latent variable $z_j$. (b) The differentiable non-centered parameterization (DNCP) where we have introduced an auxiliary 'noise' variable $\epsilon_j \sim p_\theta(\epsilon_j)$ such that $z_j$ becomes deterministic: $z_j = g_j(pa_j, \epsilon_j, \theta)$. This deterministic variable has an interpretation of a hidden layer in a neural network, which can be differentiated efficiently using the backpropagation algorithm.](image-url)
differentiable non-centered parameterization (DNCP) of the latent variable \( z_j \) be:

\[
\begin{align*}
  z_j &= g_j(p_{\alpha_j}, \epsilon_j, \theta) \quad \text{where} \quad \epsilon_j \sim p(\epsilon_j) \quad (6)
\end{align*}
\]

where \( g_j(\cdot) \) is some differentiable function. Note that in the DNCP, the value of \( z_j \) is deterministic given both \( p_{\alpha_j} \) and the newly introduced auxiliary variable \( \epsilon_j \) which is distributed as \( p(\epsilon_j) \). See figure [1] for an illustration of the two parameterizations.

By the change of variables, the relationship between the original PDF \( p_{\theta}(z_j|p_{\alpha_j}) \), the function \( g_j(p_{\alpha_j}, \epsilon_j) \) and the PDF \( p(\epsilon_j) \) is:

\[
\begin{align*}
  p(\epsilon_j) &= p_{\theta}(z_j = g_j(p_{\alpha_j}, \epsilon_j)|p_{\alpha_j}) \cdot |\text{det}(J)| \quad (7)
\end{align*}
\]

where \( \text{det}(J) \) is the determinant of Jacobian of \( g_j(\cdot) \) w.r.t. \( \epsilon_j \). If \( z_j \) is a scalar variable, then \( \epsilon_j \) is also scalar and

\[
|\text{det}(J)| = \frac{\partial z_j}{\partial \epsilon_j}.
\]

In the DNCP, the original latent variable \( z_j \) has become deterministic, and its PDF \( p_{\theta}(z_j|p_{\alpha_j}, \epsilon_j) \) can be described as a Dirac delta function (see section [2.2]).

The joint PDF over the random and deterministic variables can be integrated w.r.t. the deterministic variables. If for simplicity we assume that observed variables are always leaf nodes of the network, and that all latent variables are reparameterized such that the only random variables left are the observed and auxiliary variables \( x \) and \( \epsilon \), then the marginal joint PDF \( p_{\theta}(x, \epsilon) \) is obtained as follows:

\[
\begin{align*}
  p_{\theta}(x, \epsilon) &= \int_z p_{\theta}(x, z, \epsilon) \, dz \\
  &= \int_z \prod_j p_{\theta}(x_j|p_{\alpha_j}) \prod_j p_{\theta}(z_j|p_{\alpha_j}, \epsilon_j) \prod_j p(\epsilon_j) \, dz \\
  &= \prod_j p_{\theta}(x_j|p_{\alpha_j}) \prod_j p(\epsilon_j) \int_z \prod_j p_{\theta}(z_j|p_{\alpha_j}, \epsilon_j) \, dz \\
  &= \prod_j p_{\theta}(x_j|p_{\alpha_j}) \prod_j p(\epsilon_j) \\
  \text{where} \quad z_k &= g_k(p_{\alpha_k}, \epsilon_k, \theta)
\end{align*}
\]

(8)

In the last step of eq. (8), the inputs \( p_{\alpha_j} \) to the factors of observed variables \( p_{\theta}(x_j|p_{\alpha_j}) \) are defined in terms of functions \( z_k = g_k(\cdot) \), whose values are all recursively computed from auxiliary variables \( \epsilon \).

3.2. Approaches to DNCPs

There are a few basic approaches to transforming CP of a latent variable \( z_j \) to a DNCP:

1. Tractable and differentiable inverse CDF. In this case, let \( \epsilon_j \sim \mathcal{U}(0, 1) \), and let \( g_j(z_j, p_{\alpha_j}, \theta) = F^{-1}(z_j|p_{\alpha_j}; \theta) \) be the inverse CDF of the conditional distribution. Examples: Exponential, Cauchy, Logistic, Rayleigh, Pareto, Weibull, Reciprocal, Gompertz, Gumbel and Erlang distributions.

2. For any "location-scale" family of distributions (with differentiable log-PDF) we can choose the standard distribution (with location = 0, scale = 1) as the auxiliary variable \( \epsilon_j \), and let \( g_j(\cdot) = \text{location} + \text{scale} \cdot \epsilon_j \). Examples: Gaussian, Uniform, Laplace, Elliptical, Student’s t, Logistic and Triangular distributions.

3. Composition: It is often possible to express variables as functions of component variables with different distributions. Examples: Log-Normal (exponentiation of normally distributed variable), Gamma (a sum over exponentially distributed variables), Beta distribution, Chi-Squared, F distribution and Dirichlet distributions.

When the distribution is not in the families above, accurate differentiable approximations to the inverse CDF can be constructed, e.g. based on polynomials, with time complexity comparable to the CP (see e.g. [Devroye 1986] for some methods).

For the exact approaches above, the CP and DNCP forms have equal time complexities. In practice, the difference in CPU time depends on the relative complexity of computing derivatives of \( \log p_{\theta}(z_j|p_{\alpha_j}) \) versus computing \( g_j(\cdot) \) and derivatives of \( \log p(\epsilon_j) \), which can be easily verified to be similar in most cases below. Iterations with the DNCP form were slightly faster in our experiments.

3.3. DNCP and neural networks

It is instructive to interpret the DNCP form of latent variables as “hidden units” of a neural network. The network of hidden units together form a neural network with inserted noise \( \epsilon \), which we can differentiate efficiently using the backpropagation algorithm ([Rumelhart et al. 1986]).

There has been recent increase in popularity of deep neural networks with stochastic hidden units (e.g. [Krizhevsky et al. 2012] [Goodfellow et al. 2013] [Bengio 2013]). Often, the parameters \( \theta \) of such neural networks are optimized towards maximum-likelihood objectives. In that case, the neural network can be interpreted as a probabilistic model \( \log p_{\theta}(t|x, \epsilon) \) computing a conditional distribution over some target variable \( t \) (e.g. classes) given some input \( x \). In ([Bengio & Thibodeau-Lafuer 2013]), stochastic hidden units are used for learning the parameters of a Markov chain transition operator that samples from the data distribution.

For example, in ([Hinton et al. 2012]) a ‘dropout’ regularization method is introduced where (in its basic ver-
(a) (b)

Figure 2. (a) An illustrative hierarchical model in its centered parameterization (CP). (b) The differentiable non-centered parameterization (DNCP), where \( z_1 = g_1(\epsilon_1, \theta) \), \( z_2 = g_2(z_1, \epsilon_2, \theta) \) and \( z_3 = g_3(z_2, \epsilon_3, \theta) \), with auxiliary latent variables \( \epsilon_k \sim p_0(\epsilon_k) \). The DNCP exposes a neural network within the hierarchical model, which we can differentiate efficiently using backpropagation.

Table 1. Limiting behaviour of squared correlations between \( z \) and its parent \( y_i \) when \( z \) is in the centered (CP) and non-centered (DNCP) parameterization.

<table>
<thead>
<tr>
<th></th>
<th>( \rho_{y_i,z}^2 ) (CP)</th>
<th>( \rho_{y_i,\epsilon}^2 ) (DNCP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lim_{\sigma \to 0} )</td>
<td>1</td>
<td>( \beta w_i^2 )</td>
</tr>
<tr>
<td>( \lim_{\sigma \to +\infty} )</td>
<td>0</td>
<td>( \frac{\beta w_i^2}{\beta w_i^2 + \alpha} )</td>
</tr>
<tr>
<td>( \lim_{\beta \to 0} )</td>
<td>( \frac{\beta w_i^2}{\beta w_i^2 - \alpha \sigma^2} )</td>
<td>0</td>
</tr>
<tr>
<td>( \lim_{\beta \to +\infty} )</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( \lim_{\sigma \to 0} )</td>
<td>( \frac{1 - \beta \sigma^2}{\beta \sigma^2 - 1} )</td>
<td>( 0 )</td>
</tr>
</tbody>
</table>

3.4. A differentiable MC likelihood estimator

We showed that many hierarchical continuous latent-variable models can be transformed into a DNCP \( p_\theta(x, \epsilon) \), where all latent variables (the introduced auxiliary variables \( \epsilon \)) are root nodes (see eq. \( \theta \)). This has an important implication for learning since (contrary to a CP) the DNCP can be used to form a differentiable Monte Carlo estimator of the marginal likelihood:

\[
\log p_\theta(x) \approx \log \frac{1}{L} \sum_{l=1}^{L} \prod_j p_\theta(x_j | p_{\theta_j}^{(l)})
\]

where the parents \( p_{\theta_j}^{(l)} \) of the observed variables are either root nodes or functions of root nodes whose values are sampled from their marginal: \( \epsilon^{(l)} \sim p(\epsilon) \). This MC estimator can be differentiated w.r.t. \( \theta \) to obtain an MC estimate of the log-likelihood gradient \( \nabla_\theta \log p_\theta(x) \), which can be plugged into stochastic optimization methods such as Adagrad for approximate ML or MAP. When performed one datapoint at a time, we arrive at our on-line Maximum Monte Carlo Likelihood (MMCL) algorithm.

4. Effects of parameterizations on posterior dependencies

What is the effect of the proposed reparameterization on the efficiency of inference? If the latent variables have linear-Gaussian conditional distributions, we can use the metric of squared correlation between the latent variable and any of its children in their posterior distribution. If after reparameterization the squared correlation is decreased, then in general this will also result in more efficient inference.

For non-linear Gaussian conditional distributions, the log-PDF can be locally approximated as a linear-Gaussian using a second-order Taylor expansion. Results derived for the linear case can therefore also be applied to the non-linear case; the correlation computed using this approximation is a local dependency between the two variables. Denote by \( z \) a scalar latent variable we are going to reparameterize, and by \( y \) its parents, where \( y_i \) is one of the parents. The log-PDF of the corresponding conditional dis-
Figure 3. Plots of the log-posteriors of the illustrative linear-Gaussian model discussed in sec. 4.4. Columns: different choices of \( \sigma_z \) ranging from a low prior dependency (\( \sigma_z = 50 \)) to a high prior dependency (\( \sigma_z = 0.02 \)). First row: CP form. Second row: DNCP form. The posterior correlation \( \rho \) between the variables is also displayed. In the original form a larger prior dependency leads to a larger posterior dependency (see top row). The dependency in the DNCP posterior is inversely related to the prior dependency between \( z_1 \) and \( z_2 \) (bottom row).

The correlation \( \rho \) between two jointly Gaussian distributed variables \( A \) and \( B \) is given by: \( \rho = \sigma_{AB}^2/(\sigma_A \sigma_B) \). Using the equation above, the squared correlation can be computed from the elements of the Hessian matrix:

\[
\rho^2 = \frac{(\sigma_{AB})^2}{\sigma_A^2 \sigma_B^2} = \frac{(H_{AB}/\det(H))^2/((-H_A/\det(H))(-H_B/\det(H)))}{H_{AB}^2/(H_A H_B)}
\]

Important to note is that derivatives of the log-posterior w.r.t. the latent variables are equal to the derivatives of log-joint w.r.t. the latent variables, therefore,

\[
H = \nabla_z \nabla^T_z \log p_\theta(z|x) = \nabla_z \nabla^T_z \log p_\theta(x, z)
\]

The following shorthand notation is used in this section:

\[
L = \log p_\theta(x, z) \quad \text{(sum of all factors)}
\]

\[
z = \text{the variable to be reparameterized}
\]

\[
y = z's \text{ parents}
\]

\[
L(z) = \log p_\theta(z|y) \quad (z's \text{ factor})
\]

\[
L(z \rightarrow) = \text{the factors of } z's \text{ children}
\]

\[
L(z \leftarrow) = \text{all factors minus } z's \text{ factor}
\]

4.1. Squared Correlations

4.1.1. Centered case

In the CP case, the relevant Hessian elements are as follows:

\[
H_{yi, yj} = \frac{\partial^2 L}{\partial y_i \partial y_j} = \alpha + \frac{\partial^2 L}{\partial y_i \partial y_j} = \alpha - w_i^2/\sigma^2
\]

\[
H_{zz} = \frac{\partial^2 L}{\partial z \partial z} = \beta + \frac{\partial^2 L}{\partial z \partial z} = \beta - 1/\sigma^2
\]

\[
H_{yi, z} = \frac{\partial^2 L}{\partial y_i \partial z} = \frac{\partial^2 L}{\partial y_i \partial z} = w_i/\sigma^2
\]

Therefore, using eq. 10, the squared correlation between \( y_i \) and \( z \) is:

\[
\rho^2_{y_i, z} = \frac{(H_{yi, z})^2}{H_{yi, yj} H_{zz}} = \frac{w_i^2/\sigma^4}{(\alpha - w_i^2/\sigma^2)/(\beta - 1/\sigma^2)}
\]
4.1.2. Non-centered case

In the DNCP case, the Hessian elements are:

\[ H_{y_iy_i} = \frac{\partial^2 L}{\partial y_i \partial y_i} = \alpha + \frac{\partial}{\partial y_i} \frac{\partial L(z \rightarrow)}{\partial z} = \alpha + w_i^2 \beta \]

\[ H_{\epsilon \epsilon} = \frac{\partial^2 L}{\partial \epsilon \partial \epsilon} = \frac{\partial^2 L(z \rightarrow)}{\partial \epsilon \partial \epsilon} + \frac{\partial^2 \log p(\epsilon)}{\partial \epsilon \partial \epsilon} = \sigma^2 \beta - 1 \]

The squared correlation between \(y_i\) and \(\epsilon\) is therefore:

\[ \rho^2_{y_iz} = \frac{(H_{y_iz})^2}{H_{y_iz}H_{\epsilon \epsilon}} = \frac{\sigma^2 w_i^2 \beta^2}{(\alpha + w_i^2 \beta)(\alpha 1 \beta - 1)} \]  

(13)

4.2. Correlation inequality

We can now compare the squared correlation, between \(z\) and some parent \(y_i\), before and after the reparameterization. Assuming \(\alpha < 0\) and \(\beta < 0\) (i.e. \(L(z)\) and \(L(z \rightarrow)\) are concave, e.g. exponential families):

\[
\begin{align*}
\frac{w_i^2 / \sigma^4}{\alpha - w_i^2 / \sigma^2} &> \frac{\sigma^2 w_i^2 \beta^2}{(\alpha + w_i^2 \beta)(\alpha 1 \beta - 1)} \\
\frac{w_i^2 / \sigma^4}{\alpha - w_i^2 / \sigma^2} &> \frac{\sigma^2 w_i^2 \beta^2}{(\alpha + w_i^2 \beta)(\alpha 1 \beta - 1)} \\
\frac{1 / \sigma^4}{\alpha - w_i^2 / \sigma^2} &> \frac{\beta^2}{(\alpha + w_i^2 \beta)} \\
\sigma^{-2} &> -\beta
\end{align*}
\]  

(15)

Thus we have shown the surprising fact that the correlation inequality takes on an extremely simple form where the parent-dependent values \(\alpha\) and \(w_i\) play no role; the inequality only depends on two properties of \(z\): the relative strenghts of \(\sigma\) (its noisiness) and \(\beta\) (its influence on children’s factors). Informally speaking, if the noisiness of \(z\)’s conditional distribution is large enough compared to other factors’ dependencies on \(z\), then the reparameterized form is beneficial for inference.

4.3. A beauty-and-beast pair

Additional insight into the properties of the CP and DNCP can be gained by taking the limits of the squared correlations \(12\) and \(14\). Limiting behaviour of these correlations is shown in table 1. As becomes clear in these limits, the CP and DNCP often form a beauty-and-beast pair: when posterior correlations are high in one parameterization, they are low in the other. This is especially true in the limits of \(\sigma \rightarrow 0\) and \(\beta \rightarrow -\infty\), where squared correlations converge to either 0 or 1, such that posterior inference will be extremely inefficient in either CP or DNCP, but efficient in the other. This difference in shapes of the log-posterior is illustrated in figure 3.

4.4. Example: Simple Linear Dynamical System

Take a simple model with scalar latent variables \(z_1\) and \(z_2\), and scalar observed variables \(x_1\) and \(x_2\). The joint PDF is defined as \(p(x_1, x_2, z_1, z_2) = p(z_1) p(x_1|z_1) p(z_2|z_1) p(x_2|z_2)\), where \(p(z_1) = \mathcal{N}(0, 1)\), \(p(x_1|z_1) = \mathcal{N}(z_1, \sigma^2)\), \(p(z_2|z_1) = \mathcal{N}(z_1, \sigma^2)\) and \(p(x_2|z_2) = \mathcal{N}(z_2, \sigma^2)\). Note that the parameter \(\sigma_z\) determines the dependency between the latent variables, and \(\sigma_z\) determines the dependency between latent and observed variables.

We reparameterize \(z_2\) such that it is conditionally deterministic given a new auxiliary variable \(\epsilon_2\). Let \(p(\epsilon_2) = \mathcal{N}(0, 1)\). Let \(z_2 = g(z_1, \epsilon_2, \sigma_z) = z_1 + \sigma_z \cdot \epsilon_2\) and let \(\epsilon_1 = z_1\). See figure 3 for plots of the original and auxiliary posterior log-PDFs, for different choices of \(\sigma_z\), along with the resulting posterior correlation \(\rho\).

For what choice of parameters does the reparameterization yield smaller posterior correlation? We use equation \(15\) and plug in \(\sigma \leftarrow \sigma_z\) and \(-\beta \leftarrow \sigma^2 / \sigma_z^2\), which results in:

\[ \rho^2_{\epsilon_1, \epsilon_2} > \rho^2_{z_1, z_2} \Rightarrow \sigma^2_z < \sigma^2_x \]

i.e. the posterior correlation in DNCP form \(\rho^2_{z_1, z_2}\) is smaller when the latent-variable noise parameter \(\sigma^2_z\) is smaller than the observed-variable noise parameter \(\sigma^2_x\). Less formally, this means that the DNCP is preferred when the latent variable is more strongly coupled to the data (likelihood) then to its parents.
Auxiliary variables are used for data augmentation (see (Van Dyk & Meng, 2001) or slice sampling (Neal, 2003)) where, in contrast with our method, sampling is performed in a higher-dimensional augmented space. Auxiliary variables are used in a similar form under the name exogenous variables in Structural Causal Models (SCMs) (Pearl, 2000). In SCMs the functional form of exogenous variables is more restricted than our auxiliary variables. The concept of conditionally deterministic variables has been used earlier in e.g. (Cobb & Shenoy, 2005), although not as a tool for efficient inference in general Bayesian networks with continuous latent variables. Recently, (Raiko et al., 2012) analyzed the elements of the Hessian w.r.t. the parameters in neural network context.

The differentiable reparameterization of latent variables in this paper was introduced earlier in (Kingma & Welling, 2013) and independently in (Bengio, 2013), but these publications lacked a theoretic analysis of the impact on the efficiency of inference. In (Kingma & Welling, 2013), the reparameterization trick was used in an efficient algorithm for stochastic variational inference and learning.

6. Experiments

6.1. Nonlinear DBN

From the derived posterior correlations in the previous sections we can conclude that depending on the parameters of the model, posterior sampling can be extremely inefficient in one parameterization while it is efficient in the other. When the parameters are known, one can choose the best parameterization (w.r.t. posterior correlations) based on the correlation inequality (15).

In practice, model parameters are often subject to change, e.g. when optimizing the parameters with Monte Carlo EM; in these situations where there is uncertainty over the value of the model parameters, it is impossible to choose the best parameterization in advance. The “beauty-beast” duality from section 4.3 suggests a solution in the form of a very simple sampling strategy: mix the two parameterizations. Let \( Q_{CP}(z'|z) \) be the MCMC/HMC proposal distribution based on \( p_\theta(z|x) \) (the CP), and let \( Q_{DNCP}(z'|z) \) be the proposal distribution based on \( p_\theta(e|x) \) (the DNCP). Then the new MCMC proposal distribution based on the mixture is:

\[
Q(z'|z) = \rho \cdot Q_{CP}(z'|z) + (1-\rho) \cdot Q_{DNCP}(z'|z)
\]

(16)

where we use \( \rho = 0.5 \) in experiments. The mixing efficiency might be half that of the oracle solution (where the
optimal parameterization is known), nonetheless when taking into account the uncertainty over the parameters, the expected efficiency of the mixture proposal can be better than a single parameterization chosen ad hoc.

We applied a Hybrid Monte Carlo (HMC) sampler to a Dynamic Bayesian Network (DBN) with nonlinear transition probabilities with the same structure as the illustrative model in figure 2. The prior and conditional probabilities are: \( z_1 \sim \mathcal{N}(0, \mathbf{1}) \), \( z_i | z_{i-1} \sim \mathcal{N}(\tanh(W_xz_{i-1} + b_z), \sigma^2_z \mathbf{I}) \) and \( x_t | z_t \sim \text{Bernoulli}(\text{sigmoid}(W_xz_{t-1})) \). The parameters were initialized randomly by sampling from \( \mathcal{N}(0, \mathbf{1}) \). Based on the derived limiting behaviour (see table 1) we can expect that such a network in CP can have very large posterior correlations if the variance of the latent variables \( \sigma^2_z \) is very small, resulting in slow sampling.

To validate this result, we performed HMC inference with different values of \( \sigma^2_z \), sampling the latent variables while holding the parameters fixed. For HMC we used 10 leapfrog steps per sample, and the stepsize was automatically adjusted while sampling to obtain a HMC acceptance rate of around 0.9. At each sampling run, the first 1000 HMC samples were throwed away (burn-in); the subsequent 4000 HMC samples were kept. To estimate the efficiency of sampling, we computed the effective sample size (ESS); see e.g. (Kass et al., 1998) for a discussion on ESS.

**Results.** See table 2 and figure 4 for results. It is clear that the choice of parameterization has a large effect on posterior dependencies and the efficiency of inference. Sampling was very inefficient for small values of \( \sigma_z \) in the CP, which can be understood from the limiting behaviour in table 1.

### 6.2. Generative multilayer neural net

As explained in section 3.4, a hierarchical model in DNCP form can be learned using a MC likelihood estimator which can be differentiated and optimized w.r.t. the parameters \( \theta \). We compare this Maximum Monte Carlo Likelihood (MMCL) method with the MCEM method for learning the parameters of a 4-layer hierarchical model of the MNIST dataset, where \( x \mid z_3 \sim \text{Bernoulli}(\text{sigmoid}(W_xz_3 + b_x)) \) and \( z_1 \mid z_{t-1} \sim \mathcal{N}(\tanh(W_xz_{t-1} + b_z), \sigma_{z_t}^2 \mathbf{I}) \). For MCEM, we used HMC with 10 leapfrog steps followed by a weight update using Adagrad (Duchi et al., 2010). For MMCL, we used \( L \in \{10, 100, 500\} \). We observed that DNCP was a better parameterization than CP in this case, in terms of fast mixing. However, even in the DNCP, HMC mixed very slowly when the dimensionality of latent space became too high. For this reason, \( z_1 \) and \( z_2 \) were given a dimensionality of 3, while \( z_3 \) was 100-dimensional but noiseless (\( \sigma_{z_3}^2 = 0 \)) such that only \( z_3 \) and \( z_2 \) are random variables that require posterior inference by sampling. The model was trained on a small (1000 datapoints) and large (50000 datapoints) version of the MNIST dataset.

**Figure 5.** Performance of MMCL versus MCEM in terms of the marginal likelihood, when learning the parameters of a generative multilayer neural network (see section 6.2).

**Results.** We compared train- and testset marginal likelihood. See figure 5 for experimental results. As was expected, MCEM attains asymptotically better results. However, despite its simplicity, the on-line nature of MMCL means it scales better to large datasets, and (contrary to MCEM) is trivial to implement.

### 7. Conclusion

We have shown how Bayesian networks with continuous latent variables and generative neural networks are related through two different parameterizations of the latent variables: CP and DNCP. A key result is that the differentiable non-centered parameterization (DNCP) of a latent variable is preferred, in terms of its effect on decreased posterior correlations, when the variable is more strongly linked to the data (likelihood) then to its parents. Through theoretical analysis we have also shown that the two parameterizations are complementary to each other: when posterior correlations are large in one form, they are small in the other. We have also illustrated that this theoretical result can be exploited in practice by designing a MCMC strategy that mixes between both parameterizations, making it robust to situations where MCMC can otherwise be inefficient.

**Acknowledgments**

The authors thank the reviewers for their excellent feedback and Joris Mooij, Ter Meeds and Taco Cohen for invaluable discussions and input.
Transformations between Bayes Nets and Neural Nets

References


