BIDIRECTIONAL HELMHOLTZ MACHINES

Jörg Bornschein, Samira Shabanian, Asja Fischer, Yoshua Bengio*
Department of Computer Science and Operations Research
University of Montreal
Montreal, Quebec, Canada
{bornj,shabanis,fischer,<findme>}@iro.umontreal.ca

ABSTRACT

Efficient unsupervised training and inference in deep generative models remains a challenging problem. One basic approach, called Helmholtz machine, involves training a top-down directed generative model together with a bottom-up auxiliary model that is trained to help perform approximate inference. Recent results indicate that better results can be obtained with better approximate inference procedures. Instead of employing more powerful procedures, we here propose to regularize the generative model to stay close to the class of distributions that can be efficiently inverted by the approximate inference model. We achieve this by interpreting both the top-down and the bottom-up directed models as approximate inference distributions and by defining the model distribution to be the geometric mean of these two. We present a lower-bound for the likelihood of this model and we show that optimizing this bound regularizes the model so that the Bhattacharyya distance between the bottom-up and top-down approximate distributions is minimized. We demonstrate that we can use this approach to fit generative models with many layers of hidden binary stochastic variables to complex training distributions and that this method prefers significantly deeper architectures while it supports orders of magnitude more efficient approximate inference than other approaches.

1 INTRODUCTION AND BACKGROUND

Training good generative models and fitting them to complex and high dimensional training data with probability mass in multiple disjunct locations remains a major challenge. This is especially true for models with multiple layers of deterministic or stochastic variables, which is unfortunate because it has been argued previously (Hinton et al., 2006; Bengio, 2009) that deeper generative models have the potential to capture higher-level abstractions and thus generalize better. Although there has been progress in dealing with continous-valued latent variables (Kingma & Welling, 2014), building a hierarchy of representations, especially with discrete-valued latent variables, remains a challenge.

With the Helmholtz machine (Hinton et al., 1995; Dayan et al., 1995), a concept was introduced that proposed to not only fit a powerful but intractable generative model $p(\mathbf{x}, \mathbf{h})$ to the training data, but also to jointly train a parametric approximate inference model $q(\mathbf{h}|\mathbf{x})$. The q model would be used to efficiently perform approximate inference over the latent variables $\mathbf{h}$ of the generative model given an observed example $\mathbf{x}$. This basic idea has been applied and enhanced many times; initially with the wake-sleep algorithm (WS, Hinton et al. (1995); Dayan & Hinton (1996)) and more recently with the variational autoencoder (VAE, Kingma & Welling (2014)), stochastic backpropagation and approximate inference in deep generative models (Rezende et al., 2014), neural variational inference and learning (NVIL, Mnih & Gregor (2014)) and reweighted wake-sleep (RWS, Bornschein & Bengio (2015)).

Recent results indicate that significant improvements can be made when better approximate inference methods are used: Saliman et. al. (2014) for example presented an iterative inference procedure that improves the samples from q by employing a learned MCMC transition operator. Burda et al.

*Yoshua Bengio is a CIFAR Senior Fellow

(2015) present the importance weighted auto encoder (IWAE), an improved VAE that, similarly to RWS, uses multiple samples from q to calculate gradients. And RWS already reported that autoregressive q distributions lead to noticable improvements. In contrast to these previous approaches that aimed at incorporating more powerful inference methods, we here propose to regularize the top-down model p such that it's posterior stays close to the approximate inference distribution q (and vice versa). We archive this by interpreting both p and q as approximate inference models for our actual generative model p^*, which is defined to be the geometric mean over the top-down and bottom-up approximate inference models, i.e., $p^*(\mathbf{x}, \mathbf{h}) = {}^1/_Z \sqrt{p(\mathbf{x}, \mathbf{h}) q(\mathbf{x}, \mathbf{h})}$.

In Section 2 we will show that this definition leads to an objective that can be interpreted as using a regularization term that encurages solutions where p and q are close to each other in terms of the Bhattacharyya distance. In Section 3 we will explain how to perform importance sampling based training and inference. The ability to model complex distributions and the computational efficiency of this approach are demonstrated empirically in Section 4.

2 Model definition and properties

We introduce the concept by defining a joint probability distribution over three variables, an observed vector $\mathbf{x}$ and two latent variable vectors $\mathbf{h}_1$ and $\mathbf{h}_2$. Analogous to a Deep Boltzmann Machine (DBM, Salakhutdinov & Hinton (2009)), we think of these as layers in a neural network with links between $\mathbf{x}$ and $\mathbf{h}_1$ on the one side, and $\mathbf{h}_1$ and $\mathbf{h}_2$ on the other side. We will present the approach for the specific case of an architecture with two hidden layers, but it can be applied to arbitrary graphs of variables without loops. It can especially be used to train architectures with more than two stacked layers of latent variables.

Let $p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$ be a joint probability distribution constructed in a specific way from two constituent distributions $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$ and $q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$,

$$p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = \frac{1}{Z} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)} \;, \tag{1}$$

where Z is a normalization constant and p and q are directed graphical models from $\mathbf{h}_2$ to $\mathbf{x}$ and vice versa,

$$p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = p(\mathbf{h}_2)\; p(\mathbf{h}_1|\mathbf{h}_2)\; p(\mathbf{x}|\mathbf{h}_1) \quad \text{and} \quad q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = q(\mathbf{x})\; q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1) \;.$$

We assume that the prior distribution $p(\mathbf{h}_2)$ and all conditional distributions belong to parametrized families of distributions which can be evaluated and sampled from efficiently. For $q(\mathbf{x})$ we do not assume an explicit form but define it to be the marginal

$$\begin{aligned} q(\mathbf{x}) = p^*(\mathbf{x}) &= \sum_{\mathbf{h}_1, \mathbf{h}_2} p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = \frac{\sqrt{q(\mathbf{x})}}{Z} \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{h}_1|\mathbf{x}) q(\mathbf{h}_2|\mathbf{h}_1)} \\ &= \left(\frac{1}{Z} \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{h}_1|\mathbf{x}) q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 . \end{aligned} \tag{2}$$

The normalization constant Z guarantees that $\sum_{\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2} p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = 1$. Using the Cauchy-Schwarz inequality $\left| \sum_y f(y) g(y) \right|^2 \leq \sum_y |f(y)|^2 \times \sum_y |g(y)|^2$ and identifying $\sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}$ with $f(y)$ and $\sqrt{q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}$ with $g(y)$, it becomes clear that $Z = \sum_{\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)} \leq 1$ for arbitrary p and q. Furthermore, we see that $Z = 1$ only if $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$. We can therefore obtain a lower bound on the marginal probability $p^*(\mathbf{x})$ by defining

$$\tilde{p}^*(\mathbf{x}) = \left(\sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 = Z^2\; p^*(\mathbf{x}) \leq p^*(\mathbf{x}) \;. \tag{3}$$

This suggests that the model distribution $p^*(\mathbf{x})$ can be fitted to some training data by maximizing the bound of the log-likelihood (LL) $\log \tilde{p}^*(\mathbf{x})$ instead of $\log p^*(\mathbf{x})$, as we elaborate in the following section. Since $\log \tilde{p}^*(\mathbf{x})$ can reach the maximum only when $Z \to 1$, the model is implicitly pressured to find a maximum likelihood solution that yields $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \approx q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \approx p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$.

2.1 ALTERNATIVE VIEW BASED ON THE BHATTACHARYYA DISTANCE

Recalling the Bhattacharyya distance $D_B(p,q) = -\log \sum_y \sqrt{p(y)q(y)}$ (for which holds $D_B(p,q) \geq 0$ for arbitrary distributions p, q and $D_B(p,q) = 0$ only if $p = q$) the model LL $\log p^*(\mathbf{x})$ can be decomposed into two terms

$$\begin{aligned}\log p^*(\mathbf{x}) &= 2\log \sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)q(\mathbf{h}_1|\mathbf{x})q(\mathbf{h}_2|\mathbf{h}_1)} \\ &\quad - 2\log \sum_{\mathbf{x}',\mathbf{h}_1',\mathbf{h}_2'} \sqrt{p(\mathbf{x}',\mathbf{h}_1',\mathbf{h}_2')q(\mathbf{x}',\mathbf{h}_1',\mathbf{h}_2')} \\ &= \log \tilde{p}^*(\mathbf{x}) - 2\log Z = \log \tilde{p}^*(\mathbf{x}) + 2\, D_B(p,q) \geq \log \tilde{p}^*(\mathbf{x}) \ , \end{aligned} \tag{4}$$

where we clearly see that the proposed training objective $\log \tilde{p}^*(\mathbf{x})$ corresponds to the correct (but intractable) LL $\log p^*(\mathbf{x})$ minus 2 times the Bhattacharyya distance $D_B(p,q)$, i.e., it is maximzing the true LL and minimizing the distance between p and q. We can compare this to the variational approach, where the marginal probability $\log p(\mathbf{x})$ of some model containing latent variables $\mathbf{h}$ is rewritten in terms of the KL-divergence $D_{KL}(q(\mathbf{h}|\mathbf{x}) \,||\, p(\mathbf{h}|\mathbf{x})) = \sum_h q(\mathbf{h}|\mathbf{x}) \log \frac{q(\mathbf{h}|\mathbf{x})}{p(\mathbf{h}|\mathbf{x})} \geq 0$ to obtain a lower bound

$$\begin{aligned}\log p(\mathbf{x}) &= \mathop{\mathbb{E}}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x})} \left[\log p(\mathbf{x},\mathbf{h}) - \log q(\mathbf{h}|\mathbf{x})\right] + D_{KL}(q(\mathbf{h}|\mathbf{x}) \,||\, p(\mathbf{h}|\mathbf{x})) \\ &\geq \mathop{\mathbb{E}}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x})} \left[\log p(\mathbf{x},\mathbf{h}) - \log q(\mathbf{h}|\mathbf{x})\right] \ . \end{aligned} \tag{5}$$

Analogous to variational methods that maximize the lower bound (5), we can thus maximize $\log \tilde{p}^*(\mathbf{x})$, and it will tighten the bound as $D_B(p,q)$ approaches zero. While this seems very similar to the variational lower bound, we should highlight that there are some important conceptual differences: 1) The KL-divergence in variational methods measures the distance between distributions *given some training data*. The Bhattacharyya distance here in contrast quantifies a property of the model $p^*(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$ independently of any training data. In fact, we saw that $D_B(p,q) = -\log Z$. 2) The variational lower bound is typically used to construct approximate inference algorithms. We here use our bound $\tilde{p}^*(\mathbf{x})$ just to remove the normalization constant Z from our target distribution $p^*(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$. Even after applying the lower-bound, we still have to tackle the inference problem which manifests itself in form of the full combinatorial sum over $\mathbf{h}_1$ and $\mathbf{h}_2$ in equation (3). Although it seems intuitively reasonable to use a variational approximation on top of the bound $\tilde{p}^*(\mathbf{x})$ will here not follow this direction but rather use importance sampling to perform approximate inference and learning (see section 3). Combining a variational method with the bound $\tilde{p}^*(\mathbf{x})$ is therefore subject to future work.

We can also argue that optimizing $\log \tilde{p}^*(\mathbf{x})$ instead of $\log p^*(\mathbf{x})$ is beneficial in the light of the original goal we formulated in section 1: To learn a generative model $p^*(\mathbf{x})$ that is regularized to be close to the model q which we use to perform approximate inference for p^*. Let us assume we have two equally well trained models $p^*_{\boldsymbol{\theta}_1}$ and $p^*_{\boldsymbol{\theta}_2}$, i.e., in expectation over the empirical distribution $\mathbb{E}\left[\log p^*_{\boldsymbol{\theta}_1}(\mathbf{x})\right] = \mathbb{E}\left[\log p^*_{\boldsymbol{\theta}_2}(\mathbf{x})\right]$, but the expected bound $\tilde{p}^*(\mathbf{x})$ for the first model is closer to the LL than the expected bound for the second model: $\mathbb{E}\left[\log \tilde{p}^*_{\boldsymbol{\theta}_1}(\mathbf{x})\right] > \mathbb{E}\left[\log \tilde{p}^*_{\boldsymbol{\theta}2}(\mathbf{x})\right]$. Using equation (4) we see that $D_B(p_{\boldsymbol{\theta}_1}, q_{\boldsymbol{\theta}_1}) < D_B(p_{\boldsymbol{\theta}_2}, q_{\boldsymbol{\theta}_2})$ which indicates that $q_{\boldsymbol{\theta}_1}$ is closer to $p^*_{\boldsymbol{\theta}_1}$ than $q_{\boldsymbol{\theta}_2}$ is to $p^*_{\boldsymbol{\theta}_2}$ (when we measure their distance using the Bhattacharyya distance). According to our original goal, we thus prefer solution $p^*_{\boldsymbol{\theta}_1}$, where the bound $\tilde{p}^*(\mathbf{x})$ is maximized and the distance $D_B(p,q)$ minimized.

Note that the decomposition (4) also emphasizes why our recursive definition $q(\mathbf{x}) = \sum_{\mathbf{h}_1,\mathbf{h}_2} p^*(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$ is a consistent and reasonable one: minimizing $D_B(p,q)$ during learning means that the joint distributions $p(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$ and $q(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$ approach each other. This implies that the marginals $p(\mathbf{h}_l)$ and $q(\mathbf{h}_l)$ for all layers l become more similar. This also implies $p(\mathbf{x}) \approx q(\mathbf{x})$ in the limit of $D_B(p,q) \to 0$; a requirement that most *simple* parametrized distributions $q(\mathbf{x})$ could never fulfill.

3 INFERENCE AND TRAINING WITH IMPORTANCE SAMPLING

Algorithm 1 Learning $p^*(\mathbf{x})$ using importance sampling with q as proposal

for number of training iterations **do**
- Sample $\mathbf{x}$ from the training distribution (i.e. $\mathbf{x} \sim \mathcal{D}$)
- **for** $k = 1, 2, \ldots, K$ **do**
  - Sample $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1^{(k)}|\mathbf{x})$; for each layer $l = 2$ to L sample $\mathbf{h}_l^{(k)} \sim q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}^{(k)})$
  - Compute $q(\mathbf{h}^{(k)}|\mathbf{x})$ and $p(\mathbf{x}, \mathbf{h}^{(k)})$ for $\mathbf{h}^{(k)} = (\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_L^{(k)})$
- **end for**
- Compute unnormalized importance weights $\omega_k = \sqrt{p(\mathbf{x}, \mathbf{h}^{(k)}) / q(\mathbf{h}^{(k)}|\mathbf{x})}$
- Normalize the weights $\tilde{\omega}_k = \omega_k / \sum_{k'} \omega_{k'}$
- Update parameters of p and q: Gradient decent with gradient estimator $2 \sum_k \tilde{\omega}_k \frac{\partial \log p^*(\mathbf{x}, \mathbf{h}^{(k)})}{\partial \theta}$

end for

Based on the construction of $p^*(\mathbf{x})$ outlined in the previous section, we can define a wide range of possible models. Furthermore, we have a wide range of potential training and appropriate inference methods we could employ to maximize $\log \tilde{p}^*(\mathbf{x})$.

In this text we concentrate on binary latent and observed variables $\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2$ and model all our conditional distributions by simple sigmoid belief network layers, e.g., $p(\mathbf{x} \,|\, \mathbf{h}_1) = \prod_i \mathcal{B}(x_i \,|\, \sigma(W_i\, \mathbf{h}_1 + b_i))$ where $\mathcal{B}(x_i \,|\, c)$ refers to the Bernoulli distribution with $P(x_i = 1) = c$, W_i are the connection weights between the latent variables $\mathbf{h}_1$ and the visible variable x_i; b_i is the bias of x_i, and $\sigma(\cdot)$ is the sigmoid function. For our top-level prior $p(\mathbf{h}_2)$, we use a factorized Bernoulli distribution: $p(\mathbf{h}_2) = \prod_i \mathcal{B}(h_{2,i} \,|\, \sigma(b_{2,i}))$.

We form an estimate of $\tilde{p}^*(\mathbf{x})$ by using importance sampling instead of the exhaustive sum over $\mathbf{h}_1$ and $\mathbf{h}_2$ in equation (3). We use $q(\mathbf{h}_1|\mathbf{x})q(\mathbf{h}_2|\mathbf{h}_1)$ as the proposal distribution which is by construction easy to evaluate and to sample from:

$$\tilde{p}^*(\mathbf{x}) = \left(\sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 = \left(\mathop{\mathbb{E}}_{\substack{\mathbf{h}_2 \sim q(\mathbf{h}_2|\mathbf{h}_1) \\ \mathbf{h}_1 \sim q(\mathbf{h}_1|\mathbf{x})}} \left[\sqrt{\frac{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}{q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1)}} \right] \right)^2$$

$$\simeq \left(\frac{1}{K} \sum_{k=1}^{K} \sqrt{\frac{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})}{q(\mathbf{h}_1^{(k)}|\mathbf{x})\; q(\mathbf{h}_2^{(k)}|\mathbf{h}_1^{(k)})}} \right)^2 \quad \text{with} \quad \begin{matrix} \mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1|\mathbf{x}) \\ \mathbf{h}_2^{(k)} \sim q(\mathbf{h}_2|\mathbf{h}_1^{(k)}) \end{matrix} \quad . \tag{6}$$

Using the same approach, we can also derive the well known estimator for the marginal probability of a datapoint under the top-down generative model p:

$$p(\mathbf{x}) = \mathop{\mathbb{E}}_{\substack{\mathbf{h}_2 \sim q(\mathbf{h}_2|\mathbf{h}_1) \\ \mathbf{h}_1 \sim q(\mathbf{h}_1|\mathbf{x})}} \left[\frac{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}{q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1)} \right] \simeq \frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})}{q(\mathbf{h}_1^{(k)}|\mathbf{x})\; q(\mathbf{h}_2^{(k)}|\mathbf{h}_1^{(k)})} \quad . \tag{7}$$

Comparing (6) and (7) and making use of Jensen's inequality it becomes clear that $p(\mathbf{x}) \geq \tilde{p}^*(\mathbf{x})$.

Analogous to the parameter updates in RWS (Bornschein & Bengio, 2015), we can derive an importance sampling based estimate for the LL gradient with respect to the parameters of p and q (jointly denoted by $\boldsymbol{\theta}$) and use it to optimize our proposed regularized objective (see Appendix A for more details):

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log \tilde{p}^*(\mathbf{x}) \simeq 2 \sum_{k=1}^{K} \tilde{\omega}_k \frac{\partial}{\partial \boldsymbol{\theta}} \log \sqrt{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}) q(\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}|\mathbf{x})} \tag{8}$$

with samples $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1 \,|\, \mathbf{x})$, $\mathbf{h}_2^{(k)} \sim q(\mathbf{h}_2 \,|\, \mathbf{h}_1^{(k)})$, for $k = 1, \ldots K$, and importance weights

$$\tilde{\omega}_k = \frac{\omega_k}{\sum_{k'} \omega_{k'}} \quad , \qquad \text{where} \qquad \omega_k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})}{q(\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}|\mathbf{x})}} \quad . \tag{9}$$

In contrast to VAEs and IWAEs, the updates do not require any form of backpropagation through more than one layer because, as far as the gradient computation $\frac{\partial}{\partial \boldsymbol{\theta}} \log p^*(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})$ is concerned, these samples are considered fully observed. The gradient approximation (8) computes the

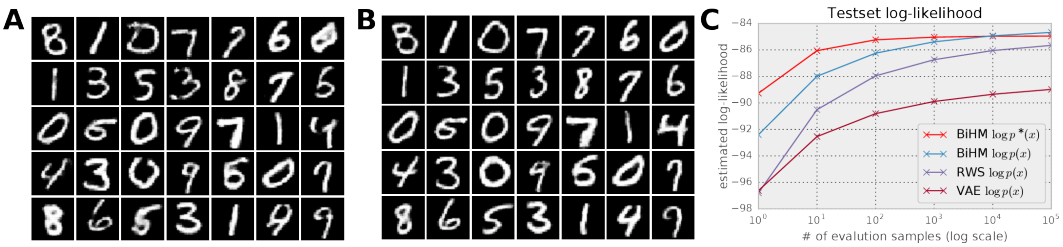


Figure 1: MNIST experiments: **A**) Random samples from top-down model $p(\mathbf{x})$. **B**) Generally improved samples after running 10 iterations of algorithm 2 (see Appendix B) starting from the samples in **A**, i.e., approximately from the joint model $p^*(\mathbf{x})$. In **A** and **B** we show expected samples instead of sampling from the bottom-most Bernoulli distribution. **C**) Sensitivity of the test set LL estimates to the number of samples K. We plot the $\log p(\mathbf{x})$ and $\log p^*(\mathbf{x})$ estimates of our best BiHM model together with the $\log p(\mathbf{x})$ estimates of our best RWS and VAE models.

weighted average over the individual gradients. These properties are basically inherited from the RWS training algorithm. But in contrast to RWS, and in contrast to most other algorithms which employ a generative model p and an approximate inference model q, we here automatically obtain parameter updates for both p and q because we optimize p^* which contains both. The resulting training method is summarized in algorithm 1.

Sampling and inpainting We now discuss two general approaches for approximate sampling from a BiHM. One can either easily and efficiently sample from the directed model p, or one can use Gibbs sampling to draw higher-quality samples from the undirected model p^*. For the latter, importance resampling is used to approximately draw samples from the conditional distributions, e.g. from $p^*(\mathbf{h}_1 \,|\, \mathbf{x}, \mathbf{h}_2)$. We here choose to draw the proposal samples from the mixture distribution $1/2\, p(\mathbf{h}_1|\mathbf{h}_2) + 1/2\, q(\mathbf{h}_1|\mathbf{x})$, which ensures that we have a symmetric chance of covering the high probability configurations of $p^*(\mathbf{h}_1|\mathbf{x}, \mathbf{h}_2)$ induced by p and q. The importance weights we use to resample a final sample from $p^*(\mathbf{h}_1|\mathbf{x}, \mathbf{h}_2)$ are thus given by

$$\omega^{(k)} = \frac{\sqrt{p(\mathbf{h}_1^{(k)}|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1^{(k)})q(\mathbf{h}_1^{(k)}|\mathbf{x})q(\mathbf{h}_2|\mathbf{h}_1^{(k)})}}{p(\mathbf{h}_1^{(k)}|\mathbf{h}_2) + q(\mathbf{h}_1^{(k)}|\mathbf{x})} \quad , \tag{10}$$

where $\mathbf{h}_1^{(k)}$ is randomly drawn from $p(\mathbf{h}_1|\mathbf{h}_2)$ or $q(\mathbf{h}_1|\mathbf{x})$ (see Appendix B for the derivation). For $p^*(\mathbf{x}|\mathbf{h}_1)$ we choose to approximate the sample by drawing the proposal samples from $p(\mathbf{x}|\mathbf{h}_1)$. For Gibbs sampling, we iteratively update all odd layers followed by all even layers until we consider the chain to be in equilibrium (pseudo code can be found in algorithm 2 in Appendix B).

Equipped with approximate sampling procedures for the conditional distributions, it is straightforward to construct an algorithm for inpainting: Given a corrupted input datapoint $\tilde{\mathbf{x}}$, we first initialize a Markov chain by drawing $\mathbf{h}_1, \mathbf{h}_2 \sim q(\mathbf{h}_1, \mathbf{h}_2|\mathbf{x})$ and then run the Gibbs sampling procedure. Whenever we sample the bottom layer $\mathbf{x} \sim p^*(\mathbf{x}|\mathbf{h}_1)$ (approximately), we keep the non-corrupted elements of $\tilde{\mathbf{x}}$ fixed. Note that this method approximately samples reconstructions $\mathbf{x} \sim p^*(\mathbf{x})$ that are consistent with $\tilde{\mathbf{x}}$; it does not provide a MAP reconstruction which would maximize $\log p^*(\mathbf{x})$ given $\tilde{\mathbf{x}}$.

Estimating the partition function Z To compute $p^*(\mathbf{x}) = \frac{1}{Z^2}\tilde{p}^*(\mathbf{x})$ and to monitor the training progress it is desirable to estimate the normalization constant Z. In stark contrast to undirected models like RBMs or DBMs, we can here derive an *unbiased* importance sampling estimator for Z^2:

$$Z^2 = \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{h}\sim p(\mathbf{x},\mathbf{h})}\left[\sqrt{\frac{q(\mathbf{h}|\mathbf{x})}{p(\mathbf{x},\mathbf{h})}} \mathop{\mathbb{E}}_{\mathbf{h}'\sim q(\mathbf{h}|\mathbf{x})}\left[\sqrt{\frac{p(\mathbf{h}',\mathbf{x})}{q(\mathbf{h}'|\mathbf{x})}}\right]\right] = \mathop{\mathbb{E}}_{\substack{\mathbf{x},\mathbf{h}\sim p(\mathbf{x},\mathbf{h}) \\ \mathbf{h}'\sim q(\mathbf{h}'|\mathbf{x})}}\left[\sqrt{\frac{p(\mathbf{x},\mathbf{h}')q(\mathbf{h}|\mathbf{x})}{p(\mathbf{x},\mathbf{h})q(\mathbf{h}'|\mathbf{x})}}\right] \quad . \tag{11}$$

We denote the number of samples used to approximate the outer expectation and the inner expectation with K_{outer} and K_{inner} respectively. In the experimental section, we show that we obtain high quality estimates for Z^2 with K_{inner}=1 and a relatively small number of samples K_{outer}. By taking

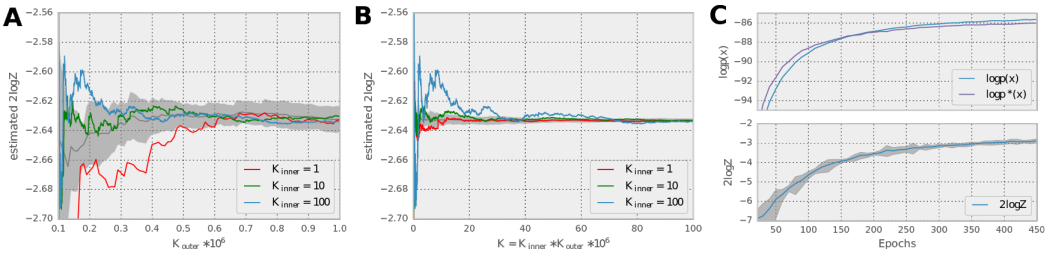


Figure 2: $2 \log Z$ estimates for different values of K_{inner} as a function of **A**) the number of samples K_{outer}, **B**) the total number of samples $K_{\text{inner}} \cdot K_{\text{outer}}$ for the BiHM trained on MNIST; the gray region shows the mean and the standard deviation for 10 runs with K_{inner}=1. This shows that, from the point of view of total computation, convergence is fastest with K_{inner}=1; and that we obtain a high quality estimate of the partition function with only a few million samples. **C**) Evolution of the estimates of $\log p(\mathbf{x})$, $\log p^*(\mathbf{x})$, and $2 \log Z$ during training on MNIST.

the logarithm, we obtain a biased estimator for $2 \log Z$, which will, unfortunately, underestimate $2 \log Z$ on average due to the concavity of the logarithm and the variance of the Z^2 estimate. This can lead to overestimates for $\log p^*(\mathbf{x})$ (see equation (6)) if we are not careful. Fortunately, the bias on the estimated $\log Z$ is induced only by the concavity of the logarithm; the underlying estimator for Z^2 is unbiased. We can thus effectively minimize the bias by minimizing the variance of the Z^2 estimate (e.g. by taking more samples). This is a much better situation than for Z-estimating methods that rely on Markov chains in high dimensional spaces, which might miss entire modes because of mixing issues.

4 Experimental results

In this section we present experimental results obtained when applying the algorithm to various binary datasets. Our main goal is to ensure that the theoretical properties discussed in section 2 translate into a robust algorithm that yields competitive results even when used with simple sigmoid belief network layers as conditional distributions. We train all models using Adam (Kingma & Ba, 2014) with a mini-batch size of 100. We initialize the weights according to (Glorot & Bengio, 2010), set the biases to -1, and use L_1 regularization λ=10^{-3} on all the weights. Our implementation is available at `https://github.com/jbornschein/bihm`.

UCI binary datasets To ascertain that a BiHM with importance sampling as training method works in general, we applied it to the 8 binary datasets from the UCI dataset repository that were evaluated e.g. in (Larochelle & Murray, 2011). The architectures, layer sizes, and learning rates used for these experiments can be found in Appendix C. We generally observe that BiHM prefers deeper architectures than RWS to obtain the best results. The results are summarized in table 4.

Binarized MNIST We use the MNIST dataset that was binarized according to Murray & Salakhutdinov (2009) and downloaded in binarized form (Larochelle, 2011). Compared to RWS, we again observe that BiHMs prefer significantly deeper and narrower models. Our best model consists of 1 visible and 12 latent layers with 300,200,100,75,50,35,30,25,20,15,10,10 latent variables. We follow the same experimental procedure as in the RWS paper: First train the model with K=10 samples and a learning rate of 10^{-3} until convergence and then fine-tune the parameters with K=100 samples and a learning rate of 3×10^{-4}. All layers are actually used to model the empirical distribution; we confirmed that training shallower models (obtained by leaving out individual layers) decreases the performance. We obtain test set log-loglikelihoods of $\log p^*(\mathbf{x}) \simeq$ **-84.8** $\pm$ **0.23** and $\log p(\mathbf{x}) \simeq$ **-84.5** $\pm$ **0.22**. The next section presents a more detailed analysis of these estimates and their dependency on the number of samples from the proposal distribution $q(\mathbf{h}|\mathbf{x})$. Note that even though this model is relatively deep, it is not particularly large, with about $700,000$ parameters in total. The DBMs in Salakhutdinov & Hinton (2009) contain about $900,000$ and 1.1 million parameters; a variational autoencoder with two deterministic, 500 units wide encoder and decoder layers, and with 100 top level latent units contains more than 1.4 million parameters. To highlight the models ability to generate crisp (non-blurry) digits we use algorithm 2 to draw samples. The results are visualized in Fig. 1 **B**. Fig. 1 **A** shows samples obtained when drawing from the top-

Figure 3: Inpainting of binarized MNIST digits. The left column in each block shows the original digit randomly sampled from the MNIST test set; the first column shows the masked version presented to the algorithm, and the next three column show different, independent reconstructions after 100 Gibbs iterations. (see section 3). All images were selected randomly.

Model	ADULT	CONNECT4	DNA	MUSHROOMS	NIPS-0-12	OCR-LETTERS	RCV1	WEB
auto regressive								
NADE	13.19	11.99	84.81	9.81	273.08	27.22	46.66	28.39
EoNADE	13.19	12.58	82.31	9.68	272.38	27.31	46.12	27.87
DARN	13.19	11.91	81.04	9.55	274.68	28.17	46.10	28.83
RWS - NADE	13.16	11.68	84.26	9.71	271.11	26.43	46.09	27.92
non AR								
RBM	16.26	22.66	96.74	15.15	277.37	43.05	48.88	29.38
RWS - SBN	13.65	12.68	90.63	9.90	272.54	29.99	46.16	28.18
BiHM								
$-\log p(\mathbf{x})$	13.78	12.43	86.49	9.40	272.66	27.10	46.12	28.14
$-\log p^*(\mathbf{x})$	13.82	12.31	86.92	9.40	272.71	27.30	46.98	28.22
$-2\log Z$	0.20	0.27	0.56	0.09	1.97	1.87	0.41	0.54

Table 1: Negative log-likelihood (NLL) on various binary datasets from the UCI repository: The top rows quote results from shallow models with autoregressive weights between their units within one layer. The second block shows results from non-autoregressive models (quoted from Bornschein & Bengio (2015)). In the third block we show the results obtained by training a BiHMs. We report the estimated test set NLL when evaluating just the top-down model, $\log p(\mathbf{x})$, and when evaluating $\log p^*(\mathbf{x})$. Our BiHM models consistently obtain similar or better results than RWS while they prefer deeper architectures.

down generative model $p(\mathbf{x})$ before running any Gibbs iterations. Fig. 3 visualizes the results when running the inpainting algorihm to reconstruct partially occluded images. Our sourcecode package contains additional results and animations.

Toronto Face Database We also trained models on the 98,058 examples from the unlabeled section of the Toronto face database (TFD, Susskind et al. (2010)). Each training example is of size 48×48 pixels and we interpret the gray-level as Bernoulli probability for the bottommost layer. We observe that training proceeds rapidly during the first few epochs but mostly only learns the mean-face. During the next few hundred epochs training proceeds much slower but the estimated log-likelihood $\log p^*(\mathbf{x})$ increases steadily. Fig. 4 **A** shows random samples from a model with 1000,700,700,300 latent variables in 4 hidden layers. It was trained with a learning rate of $3 \cdot 10^{-5}$; all other hyperparameters were set to the same values as before. Fig. 4 **B** shows the results from inpainting experiments with this model.

4.1 Analysis of IS-based estimates

Estimating the partition function In Fig. 2 **A** we plot $2\log Z$ estimates (equation (11)) over the number of outer samples K_{outer} for our best MNIST model and for 3 different choices of K_{inner}, i.e.,$K_{\text{inner}} \in \{1, 10, 100\}$. In Fig. 2 **B** we plot the estimates over the total number of samples $K_{\text{outer}} \cdot K_{\text{inner}}$. We observe that choosing K_{inner} =1 and using only about 10 million samples results in high quality estimates for $2\log Z$ with an standard error far below 0.1 nats. Estimating based on 10 million samples takes less than 2 minutes on a GTX980 GPU. Fig. 2 **C** shows the development of the $2\log Z$ estimate during learning and in relation to the LL estimates.

Importance sampling efficiency A widely used metric to estimate the quality of an IS estimator is the *effective sampling size* (ESS), given by $\widehat{ess} = (\sum_{k=1}^{K} \omega_k)^2 / (\sum_{k=1}^{K} \omega_k^2)$ (see, e.g., Robert & Casella (2009)). We compute the ESS over the MNIST test set for K=100,000 proposal samples from $q(\mathbf{h}|\mathbf{x})$. For our best RWS model, a model with 5 stochastic layers (400,300,200,100,10), we obtain $\widehat{ess} \simeq 0.10\% \pm 0.06$; for the BiHM model we obtain $\widehat{ess} \simeq 11.9\% \pm 1.1$. When we estimate the ESS for using $q(\mathbf{h}|\mathbf{x})$ from the BiHM as a proposal distribution for $p(\mathbf{h}|\mathbf{x})$, we obtain

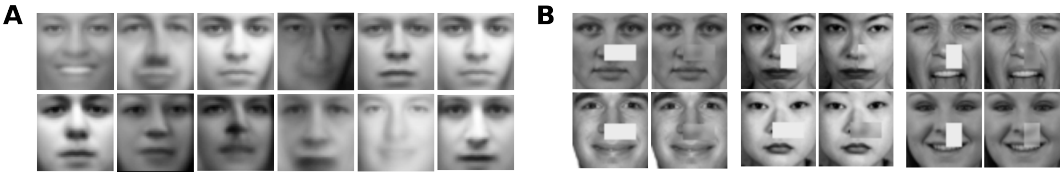


Figure 4: Results after training on TFD: **A**) Random selection of 12 samples drawn from $p^*(\mathbf{x})$ using algorithm 2 (100 iterations of Gibbs sampling; see Appendix E for more samples). **B**) The left column in each block shows the input; the right column shows a random output sample generated by the inpaiting algorithm (see section 3).

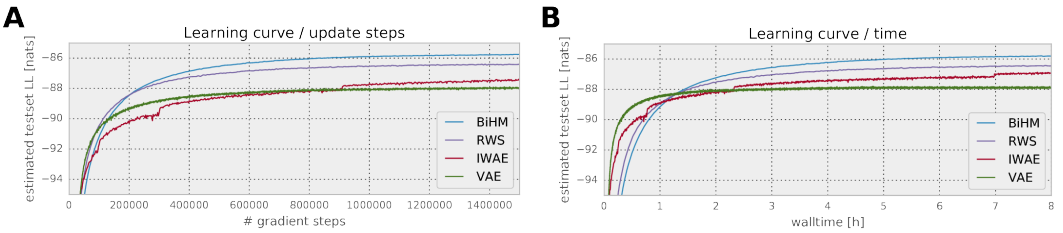


Figure 5: Learning curves for MNIST experiments: For BiHM, RWS and VAE we chose the learning rate such that we get optimal results after 10^6 update steps; for IWAE we use the original learning rate schedule published in (Burda et al., 2015). BiHM and RWS use K=10 samples per datapoint; IWAE uses K=5 and a batch size of 20 (according to the original publication). We generally observe that BiHM show very good convergence in terms of progress per update step and in terms of total training time.

$\widehat{ess}$=1.2% $\pm$ 0.2. These results indicate that training BiHM models indeed results in distributions whose intractable posterior $p^*(\mathbf{h}|\mathbf{x})$ as well as top-down model $p(\mathbf{h}|\mathbf{x})$ are much better modeled by the learned $q(\mathbf{h}|\mathbf{x})$. Although not directly comparable, we also estimated the ESS for a VAE with two determninistic, 500 units wide ReLU layers in the encoder and decoder. This model has a single stochastic layer with 100 continuous variables at the top; it reaches a final estimated test set LL of $\log p(\mathbf{x}) \simeq -88.9 \pm 0.28$. The final variational lower bound, which corresponds exactly to the importance sampling estimate of $\log p(\mathbf{x})$ with K=1 sample, is -95.8. For this model we obtain an ESS of $0.07\% \pm 0.02$. These results indicate that we need thousands of samples to obtain reliable LL estimates with low approximation error. In Fig. 1 **C** we plot the estimated test set LL over the number of samples K used to estimate $\log p^*(\mathbf{x})$ and $\log p(\mathbf{x})$. For all the models and for small a number of samples K we significantly underestimate the LL; but, in comparison to RWS, the estimates for the BiHM model are much higher and less sensitive to K. E.g, using K=10 samples to evaluate the BiHM model results in a higher LL estimate than using K=10,000 samples to evaluate the RWS model.

Computational cost To demonstrate the computational efficiency of our approach we show typical MNIST learning curves in Fig. 5. For BiHM, RWS and VAE the learning rate was chosen within a factor of 2 to obtain optimal results after 10^6 update steps ($5 \cdot 10^{-4}$ for BiHM and RWS, $3 \cdot 10^{-3}$ for VAE; K=10 for BiHM and RWS). For the IWAE experiment we use the original code, hyper parameters and learning rate schedule from (Burda et al., 2015): This experiment thus uses a mini-batch size of 20 instead of 100, K=5 training samples and and 8 different learning rates over the course of $\approx$ 3300 epochs. In all cases we used K=1000 samples to evaluate the testset log-likelihoods. We generally observe that BiHM show very good convergence in terms of progress per update step and competitive performance in terms of total training time. Note that BiHMs and RWS allow for an efficient distributed implementation in the future: Per sample, only the binary activations and a single floating point number (the importance weight) need to be communicated between layers. VAEs and IWAEs need to communicate continuous activations during the forward pass and continuous partial gradients during the backward pass. At test time on the other hand, BiHMs are typically much more effective than the other methods: BiHMs obtain good LL estimates with K=10 or 100 samples per datapoint while VAE, RWS and IWAE models require $\approx$ 10,000 samples to obtain competitive results (compare Fig. 1 **C**).

5 CONCLUSION AND FUTURE WORK

We introduced a new scheme to construct probabilistic generative models which are automatically regularized to be close to approximate inference distributions we have at our disposal. Using the Bhattacharyya distance we derived a lower-bound on the log-likelihood, and we demonstrated that the bound can be used to fit deep generative models with many layers of latent variables to complex training distributions.

Compared to RWS, BiHM models typically prefer many more latent layers. After training a BiHM, the directed top-down model p shows better performance than a RWS trained model; both in terms of log-likelihood and sample quality. Sample quality can be further improved by approximately sampling from the full undirected BiHM model p^*. The high similarity between p^* and q, enforced by the training objective, allows BiHMs to be evaluated orders of magnitude more efficiently than RWS, VAE and IWAE models.

Possible directions for future research could involve semisupervised learning: The symmetric nature of the generative model p^* (it is always close to the bottom-up and top-down directed models q and p) might make it suitable for learning tasks that require inference given changing sets of observed and hidden variables. We also have a wide range of potential choices for our parametrized conditional distributions. Assuming continuous latent variables for example and eventually choosing an alternative inference method might make p^* a better suited model for some training distributions.

Acknowledgments: We thank the developers of Theano Bergstra et al. (2010) and Blocks Van Merriënboer et al. (2015) for their great work. We thank NSERC, Compute Canada, CIFAR and Samsung for their support.

REFERENCES

Bengio, Yoshua. *Learning deep architectures for AI*. Now Publishers, 2009.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In *Proc. SciPy*, 2010.

Bornschein, Jorg and Bengio, Yoshua. Reweighted wake-sleep. In *International Conference on Learning Representations (ICLR'2015)*, 2015.

Burda, Yuri, Grosse, Roger, and Salakhutdinov, Ruslan. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Dayan, Peter and Hinton, Geoffrey E. Varieties of helmholtz machine. *Neural Networks*, 9(8): 1385–1403, 1996.

Dayan, Peter, Hinton, Geoffrey E, Neal, Radford M, and Zemel, Richard S. The Helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS'2010*, 2010.

Hinton, Geoffrey E., Dayan, Peter, Frey, Brendan J., and Neal, Radford M. The wake-sleep algorithm for unsupervised neural networks. *Science*, 268:1558–1161, 1995.

Hinton, Geoffrey E., Osindero, Simon, and Teh, Yee Whye. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Durk P. and Welling, Max. Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Larochelle, Hugo. Binarized mnist dataset. `http://www.cs.toronto.edu/~larocheh/public/datasets/binarized_mnist/binarized_mnist_[train|valid|test].amat`, 2011. URL `http://www.cs.toronto.edu/~larocheh/public/datasets/binarized_mnist/binarized_mnist_train.amat`.

Larochelle, Hugo and Murray, Ian. The Neural Autoregressive Distribution Estimator. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS'2011)*, volume 15 of JMLR: W&CP, 2011.

Mnih, Andriy and Gregor, Karol. Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, 2014. to appear.

Murray, Iain and Salakhutdinov, Ruslan. Evaluating probabilities under high-dimensional latent variable models. In *NIPS'08*, volume 21, pp. 1137–1144, 2009.

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

Robert, Christian and Casella, George. *Introducing Monte Carlo Methods with R*. Springer Science & Business Media, 2009.

Salakhutdinov, Ruslan and Hinton, Geoffrey E. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 448–455, 2009.

Susskind, Joshua, Anderson, Adam, and Hinton, Geoffrey E. The Toronto face dataset. Technical Report UTML TR 2010-001, U. Toronto, 2010.

Van Merriënboer, Bart, Bahdanau, Dzmitry, Dumoulin, Vincent, Serdyuk, Dmitriy, Warde-Farley, David, Chorowski, Jan, and Bengio, Yoshua. Blocks and fuel: Frameworks for deep learning. *ArXiv e-prints*, (1506.00619), June 2015. URL `http://adsabs.harvard.edu/abs/2015arXiv150600619V`.

Appendix

A IS-ESTIMATE OF THE $\log \tilde{p}^*(\mathbf{x})$ GRADIENT

To ease the notation, we will use $\mathbf{h}$ to jointly denote the hidden variables of all hidden layers in the following derivation.

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}} \log \tilde{p}^*(\mathbf{x}) &= \frac{\partial}{\partial \boldsymbol{\theta}} \log \left(\sum_{\mathbf{h}} \sqrt{p(\mathbf{x}, \mathbf{h}) q(\mathbf{h}|\mathbf{x})} \right)^2 \\ &= \frac{2}{\sum_{\mathbf{h}} \sqrt{p(\mathbf{x}, \mathbf{h}) q(\mathbf{h}|\mathbf{x})}} \sum_{\mathbf{h}'} \sqrt{p(\mathbf{x}, \mathbf{h}') q(\mathbf{h}'|\mathbf{x})} \frac{\partial}{\partial \boldsymbol{\theta}} \log \sqrt{p(\mathbf{x}, \mathbf{h}') q(\mathbf{h}'|\mathbf{x})} \\ &\simeq 2 \sum_{k=1}^{K} \tilde{\omega}_k \frac{\partial}{\partial \boldsymbol{\theta}} \log \sqrt{p(\mathbf{x}, \mathbf{h}^{(k)}) q(\mathbf{h}^{(k)}|\mathbf{x})} \ , \end{aligned} \tag{12}$$

with samples $\mathbf{h}^{(k)} \sim q(\mathbf{h} \,|\, \mathbf{x})$ for $k = 1, \ldots, K$ and importance weights

$$\tilde{\omega}_k = \frac{\omega_k}{\sum_{k'} \omega_{k'}} \text{ , with } \omega_k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(k)})}{q(\mathbf{h}^{(k)} \,|\, \mathbf{x})}} \ .$$

B IMPORTANCE RESAMPLING

In the following, we show how the importance resampling weights for approximate sampling from the conditional distributions $p^*(\mathbf{h}_l|\mathbf{h}_{l-1}, \mathbf{h}_{l+1})$ can be derived. If the proposal distribution is given by $\frac{1}{2}p(\mathbf{h}_l|\mathbf{h}_{l+1}) + \frac{1}{2}q(\mathbf{h}_l|\mathbf{h}_{l-1})$, the unnormalized importance weight for a sample $\mathbf{h}_l^{(k)}$ from this proposal is

$$\omega_k = \frac{p^*(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}, \mathbf{h}_{l+1})}{\frac{1}{2}p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) + \frac{1}{2}q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})} \ .$$

Note that

$$p^*(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}, \mathbf{h}_{l+1}) = \frac{\sqrt{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) p(\mathbf{h}_{l-1}|\mathbf{h}_l^{(k)}) q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}) q(\mathbf{h}_{l+1}|\mathbf{h}_l^{(k)})}}{\sum_{\mathbf{h}_l} \sqrt{p(\mathbf{h}_l|\mathbf{h}_{l+1}) p(\mathbf{h}_{l-1}|\mathbf{h}_l) q(\mathbf{h}_l|h_{l-1}) q(\mathbf{h}_{l+1}|\mathbf{h}_l)}}$$

and therefore

$$\omega_k = \frac{\sqrt{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) p(\mathbf{h}_{l-1}|\mathbf{h}_l^{(k)}) q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}) q(\mathbf{h}_{l+1}|\mathbf{h}_l^{(k)})}}{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) + q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})} \cdot C \ ,$$

where $C = \frac{2}{\sum_{\mathbf{h}_l} \sqrt{p(\mathbf{h}_l|\mathbf{h}_{l+1}) p(\mathbf{h}_{l-1}|\mathbf{h}_l) q(\mathbf{h}_l|\mathbf{x}) q(\mathbf{h}_{l+1}|\mathbf{h}_l)}}$ is independent of the sample $\mathbf{h}_l^{(k)}$ and thus can be ignored when drawing samples from $\{\mathbf{h}_l^{(1)}, \mathbf{h}_l^{(2)}, \ldots, \mathbf{h}_l^{(K)}\}$ with a probability proportional to $\{\omega_1, \omega_2, \ldots, \omega_K\}$.

Algorithm 2 Sampling from $p^*(\mathbf{x}, \mathbf{h})$

• Draw $(\mathbf{x}, \mathbf{h}_1, \ldots, \mathbf{h}_L) \sim p(\mathbf{x}, \mathbf{h}_1, \ldots, \mathbf{h}_L)$ from the top-down model to initialize state
for number of iterations **do**
  for $l = \underbrace{1, 3, \ldots, L-1}_{odd\,layers}, \underbrace{2, \ldots, L}_{even\,layers}$ **do**
    for $k = 1, 2, \ldots, K$ **do**
      • Draw proposal sample $\mathbf{h}_l^{(k)} \sim \frac{1}{2} p(\mathbf{h}_l|\mathbf{h}_{l+1}) + \frac{1}{2} q(\mathbf{h}_l|\mathbf{h}_{l-1})$
      • Compute $\omega^{(k)} = \frac{\sqrt{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) p(\mathbf{h}_{l-1}|\mathbf{h}_l^{(k)}) q(\mathbf{h}_{l+1}|\mathbf{h}_l^{(k)}) q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})}}{(p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) + q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}))}$
    end for
    • Randomly pick $\mathbf{h}_l \in \{\mathbf{h}_l^{(1)}, \ldots \mathbf{h}_l^{(K)}\}$ with probability proportional to $\{\omega^{(1)}, \ldots, \omega^{(K)}\}$
  end for
  for $k = 1, 2, \ldots, K$ **do**
    • Draw proposal sample $\mathbf{x}^{(k)} \sim p(\mathbf{x}|\mathbf{h}_1)$
    • Compute estimated marginals $\tilde{p}^*(\mathbf{x}^{(k)})$ using equation (6)
    • Compute importance weights $\omega^{(k)} = \sqrt{\tilde{p}^*(\mathbf{x}^{(k)}) q(\mathbf{h}_1|\mathbf{x}^{(k)}) / p(\mathbf{x}^{(k)}|\mathbf{h}_1)}$
  end for
  • Randomly pick $\mathbf{x} \in \{\mathbf{x}^{(1)}, \ldots \mathbf{x}^{(K)}\}$ with probability proportional to $\{\omega^{(1)}, \ldots, \omega^{(K)}\}$
end for

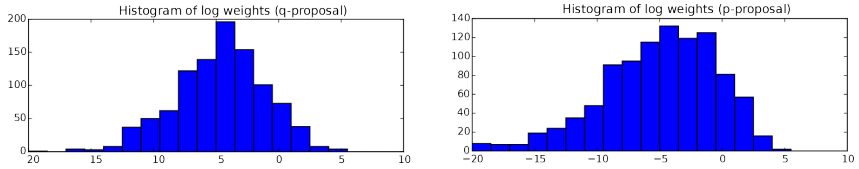
To simplify the notation, we here assume $p(\mathbf{h}_L|\mathbf{h}_L + 1) = p(\mathbf{h}_L)$. An algorithm for inpainting is obtained by drawing the initial state from $q(\mathbf{h}|\mathbf{x})$ instead from $p(\mathbf{x}, \mathbf{h})$ and by keeping the known elements of $\mathbf{x}$ fixed when sampling from $p(\mathbf{x}|\mathbf{h}_1)$.

C DETAILS FOR THE UCI EXPERIMENTS

The following table summarizes the experimental details for the experiments with BiHMs on the UCI datasets. The network architecture, i.e. the number of units of all hidden layers, the learning rate and the resulting ESS are listed. For comparison we also show the network architecture for RWS which lead to the best results in Bornschein & Bengio (2015) in brackets.

Dataset	**UCI hidden units** **RWS hidden units**	**Learning rate**	**ESS**
ADULT	100, 70, 50, 25 (100, 20, 5)	10^{-2}	$81.50\% \pm 0.11$
CONNECT4	300, 110, 50, 21 (150 , 50, 10)	10^{-2}	$89.17\% \pm 0.05$
DNA	200,150,130,100,70,50,30,20,10 (150, 10)	10^{-3}	$11.28\% \pm 0.07$
MUSHROOMS	150, 100, 90, 60, 40, 20 (150 , 50, 10)	10^{-2}	$92.56\% \pm 0.12$
NIPS-0-12	200, 100, 50, 25 (150 , 50, 10)	10^{-3}	$16.83\% \pm 0.2$
OCR-LETTERS	600, 500, 100, 55, 30, 11 (300, 100, 10)	10^{-3}	$22.58\% \pm 0.09$
RCV1	500, 120, 70, 35 (200, 50, 10)	10^{-3}	$70.60\% \pm 0.08$
WEB	650, 580, 70, 35, 11 (300, 50, 10)	10^{-3}	$55.95\% \pm 0.15$

D SYMMETRY OF p AND q



Here we show the histograms of the importance weights when we use $q(\mathbf{h}|\mathbf{x})$ as a proposal for $p^*(\mathbf{h}|\mathbf{x})$ (left), and when we use $p(\mathbf{x}, \mathbf{h})$ as a proposal distribution for $p^*(\mathbf{x}, \mathbf{h})$ (right). According to our goal formulated in section 1, both p and q should stay close to p^*. The weights of the former occur whenever we perform approximate inference (e.g. during learning); the weights of the latter occur when we sample from the model. Unsurprisingly, we observe that the quality of both proposal distributions is roughly symmetric relative to the target distribution p^*. This indicates that drawing a sample from p^* and performing approximate inference $\mathbf{h} \sim p^*(\mathbf{h}|\mathbf{x})$ are now similarly hard problems. This is different from other Helmholtz machines, where sampling from the model is straight forward and exact, but inference is even harder (compare ESS section 4.1).

E SAMPLES FROM THE TFD MODEL

The following figure shows 100 samples from the BiHM trained on the TFD data set.