ARTICLE **Communicated by Yoshua Bengio**

An Efficient Learning Procedure for Deep Boltzmann Machines

Ruslan Salakhutdinov
rsalakhu@utstat.toronto.edu
Department of Statistics, University of Toronto, Toronto, Ontario M5S 3G3, Canada

Geoffrey Hinton
hinton@cs.toronto.edu
Department of Computer Science, University of Toronto, Toronto, Ontario M5S 3G3, Canada

We present a new learning algorithm for Boltzmann machines that contain many layers of hidden variables. Data-dependent statistics are estimated using a variational approximation that tends to focus on a single mode, and data-independent statistics are estimated using persistent Markov chains. The use of two quite different techniques for estimating the two types of statistic that enter into the gradient of the log likelihood makes it practical to learn Boltzmann machines with multiple hidden layers and millions of parameters. The learning can be made more efficient by using a layer-by-layer pretraining phase that initializes the weights sensibly. The pretraining also allows the variational inference to be initialized sensibly with a single bottom-up pass. We present results on the MNIST and NORB data sets showing that deep Boltzmann machines learn very good generative models of handwritten digits and 3D objects. We also show that the features discovered by deep Boltzmann machines are a very effective way to initialize the hidden layers of feedforward neural nets, which are then discriminatively fine-tuned.

1 A Brief History of Boltzmann Machine Learning

The original learning procedure for Boltzmann machines (see section 2) makes use of the fact that the gradient of the log likelihood with respect to a connection weight has a very simple form: it is the difference of two pair-wise statistics (Hinton & Sejnowski, 1983). The first statistic is data dependent and is the expectation that both binary stochastic units in a pair are on when a randomly selected training case is clamped on the "visible" units and the states of the "hidden" units are sampled from their posterior distribution. The second statistic is data independent and is the expectation that both units are on when the visible units are not constrained by data and the states of the visible and hidden units are sampled from the joint distribution defined by the parameters of the model.

Hinton and Sejnowski (1983) estimated the data-dependent statistics by clamping a training vector on the visible units, initializing the hidden units to random binary states, and using sequential Gibbs sampling of the hidden units (Geman & Geman, 1984) to approach the posterior distribution. They estimated the data-independent statistics in the same way, but with the randomly initialized visible units included in the sequential Gibbs sampling. Inspired by Kirkpatrick, Gelatt, and Vecchi (1983), they used simulated annealing from a high initial temperature to a final temperature of one to speed up convergence to the stationary distribution. They demonstrated that this was a feasible way of learning the weights in small networks, but even with the help of simulated annealing, this learning procedure was much too slow to be practical for learning large, multilayer Boltzmann machines. Even for small networks, the learning rate must be very small to avoid an unexpected effect: the high variance in the difference of the two estimated statistics has a tendency to drive the parameters to regions where each hidden unit is almost always on or almost always off. These regions act as attractors because the variance in the gradient estimate is lower in these regions, so the weights change much more slowly.

Neal (1992) improved the learning procedure by using persistent Markov chains. To estimate the data-dependent statistics, the Markov chain for each training case is initialized at its previous state for that training case and then run for just a few steps. Similarly, for the data-independent statistics, a number of Markov chains are run for a few steps from their previous states. If the weights have changed only slightly, the chains will already be close to their stationary distributions, and a few iterations will suffice to keep them close. In the limit of very small learning rates, therefore, the data-dependent and data-independent statistics will be almost unbiased. Neal did not explicitly use simulated annealing, but the persistent Markov chains implement it implicitly, provided that the weights have small initial values. Early in the learning, the chains mix rapidly because the weights are small. As the weights grow, the chains should remain near their stationary distributions in much the same way as simulated annealing should track the stationary distribution as the inverse temperature increases.

Neal (1992) showed that persistent Markov chains work quite well for training a Boltzmann machine on a fairly small data set. For large data sets, however, it is much more efficient to update the weights after a small minibatch of training examples, so by the time a training example is revisited, the weights may have changed by a lot and the stored state of the Markov chain for that training case may be far from equilibrium. Also, once the weights become large, the Markov chains used for estimating the data-independent statistics may have a very slow mixing rate since they typically need to sample from a highly multimodal distribution in which widely separated modes have very similar probabilities but the vast majority of the joint states are extremely improbable. This suggests that the learning rates might need to be impractically small for the persistent chains to remain close to their stationary distributions with only a few state updates per weight update. Fortunately, the asymptotic analysis is almost completely irrelevant: there is a subtle reason, explained later in this section, why the learning works well with a learning rate that is much larger than the obvious asymptotic analysis would allow.

In an attempt to reduce the time required by the sampling process, Peterson and Anderson (1987) replaced Gibbs sampling with a simple mean-field method that approximates a stationary distribution by replacing stochastic binary values with deterministic real-valued probabilities. More sophisticated deterministic approximation methods were investigated by Galland (1991) and Kappen and Rodriguez (1998), but none of these approximations worked very well for learning for reasons that were not well understood at the time.

It is now well known that in directed graphical models, learning typically works quite well when the statistics from the true posterior distribution that are required for exact maximum likelihood learning are replaced by statistics from a simpler approximating distribution, such as a simple mean-field distribution (Zemel, 1993; Hinton & Zemel, 1994; Neal & Hinton, 1998; Jordan, Ghahramani, Jaakkola, & Saul, 1999). The reason learning still works is that it follows the gradient of a variational bound (see section 2.2). This bound consists of the log probability that the model assigns to the training data penalized by the sum, over all training cases, of the Kullback-Leibler divergence between the approximating posterior and the true posterior over the hidden variables. Following the gradient of the bound tends to minimize this penalty term, thus making the true posterior of the model similar to the approximating distribution.

An undirected graphical model, such as a Boltzmann machine, has an additional, data-independent term in the maximum likelihood gradient. This term is the derivative of the log partition function, and unlike the data-dependent term, it has a negative sign. This means that if a variational approximation is used to estimate the data-independent statistics, the resulting gradient will tend to change the parameters to make the approximation worse. This probably explains the lack of success in using variational approximations for learning Boltzmann machines.

The first efficient learning procedure for large-scale Boltzmann machines used an extremely limited architecture, first proposed in Smolensky (1986), that was designed to make inference easy. A restricted Boltzmann machine (RBM) has a layer of visible units and a layer of hidden units with no connections between the hidden units. The lack of connections between hidden units eliminates many of the computational properties that make general Boltzmann machines interesting, but makes it easy to compute the data-dependent statistics exactly, because the hidden units are independent given a data vector. If connections between visible units are also prohibited, the data-independent statistics can be estimated by starting Markov chains at hidden states that were inferred from training vectors, and alternating between updating all of the visible units in parallel and updating all of the hidden units in parallel (Hinton, 2002). It is hard to compute how many alternations (half-steps) of these Markov chains are needed to approach the stationary distribution, and it is also hard to know how close this approach must be for learning to make progress toward a better model. It is tempting to infer that if the learning works, the Markov chains used to estimate the data-independent statistics must be close to equilibrium, but it turns out that this is quite wrong.

Empirically, learning usually works quite well if the alternating Gibbs sampling is run for only one full step starting from the sampled binary states of the hidden units inferred from a data vector (Hinton, 2002). This gives very biased estimates of the data-independent statistics, but it greatly reduces the variance in the estimated difference between data-dependent and data-independent statistics (Williams & Agakov, 2002), especially when using mini-batch learning on large data sets. Much of the sampling error in the data-dependent statistics caused by using a small mini-batch is eliminated because the estimate of the data-independent statistics suffers from a very similar sampling error. The reduced variance allows a much higher learning rate. Instead of viewing this learning procedure as a gross approximation to maximum likelihood learning, it can be viewed as a much better approximation to minimizing the difference of two divergences (Hinton, 2002), and so it is called contrastive divergence (CD) learning. The quality of the learned model can be improved by using more full steps of alternating Gibbs sampling as the weights increase from their small initial values (Carreira-Perpignan & Hinton, 2005), and with this modification, CD learning allows RBMs with millions of parameters to achieve state-of-the art performance on a large collaborative filtering task (Salakhutdinov, Mnih, & Hinton, 2007).[1]

The architectural limitations of RBMs can be overcome by using them as simple learning modules that are stacked to form a deep, multilayer network. After training each RBM, the activities of its hidden units, when they are being driven by data, are treated as training data for the next RBM (Hinton, Osindero, & Teh, 2006; Hinton & Salakhutdinov, 2006). However, if multiple layers are learned in this greedy, layer-by-layer way, the resulting composite model is not a multilayer Boltzmann machine (Hinton et al., 2006). It is a hybrid generative model called a deep belief net that has undirected connections between its top two layers and downward-directed connections between all adjacent lower layers.

[1]The performance is comparable with the best other single models, such as probabilistic matrix factorization. By averaging many models, it is possible to do better, and the two systems with the best performance on Netflix use multiple RBMs among the many models that are averaged.

In this article, we present a fairly efficient learning procedure for fully general Boltzmann machines. To estimate the data-dependent statistics, we use mean-field variational inference and rely on the learning to make the true posterior distributions be close to the factorial distributions assumed by the mean-field approximation. To estimate the data-independent statistics, we use a relatively small number of persistent Markov chains and rely on a subtle interaction between the learning and the Markov chains to allow a small number of slow mixing chains to sample quickly from a highly multimodal energy landscape. For both sets of statistics, the fact that the parameters are changing is essential for making the estimation methods work.

We then show how to make our learning procedure for general Boltzmann machines considerably more efficient for deep Boltzmann machines (DBMs) that have many hidden layers but no connections within each layer and no connections between nonadjacent layers. The weights of a DBM can be initialized by training a stack of RBMs, but with a modification that ensures that the resulting composite model is a Boltzmann machine rather than a deep belief net (DBN). This pretraining method has the added advantage that it provides a fast, bottom-up inference procedure for initializing the mean-field inference. We use the MNIST and NORB data sets to demonstrate that DBMs learn very good generative models of images of handwritten digits and 3D objects. Although this article is primarily about learning generative models, we also show that the weights learned by these models can be used to initialize deep feedforward neural networks. These feedforward networks can then be fine-tuned using backpropagation to give much better discriminative performance than randomly initialized networks.

2 Boltzmann Machines

A Boltzmann machine (BM) is a network of symmetrically coupled stochastic binary units. It contains a set of visible units $\mathbf{v} \in \{0,1\}^{\mathcal{V}}$ and a set of hidden units $\mathbf{h} \in \{0,1\}^{\mathcal{U}}$ (see Figure 1, left panel) that learn to model higher-order correlations between the visible units. The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is defined as

$$E(\mathbf{v}, \mathbf{h}; \theta) = -\mathbf{v}^\top W \mathbf{h} - \frac{1}{2}\mathbf{v}^\top L \mathbf{v} - \frac{1}{2}\mathbf{h}^\top J \mathbf{h}, \tag{2.1}$$

where $\theta = \{W, L, J\}$ are the model parameters.[2] W, L, J represent visible-to-hidden, visible-to-visible, and hidden-to-hidden symmetric interaction

[2]We have omitted the bias terms for clarity of presentation. Biases are equivalent to weights on a connection to a unit whose state is fixed at 1, so the equations for their derivatives can be inferred from the equations for the derivatives with respect to weights by simply setting the state of one of the two units to 1.

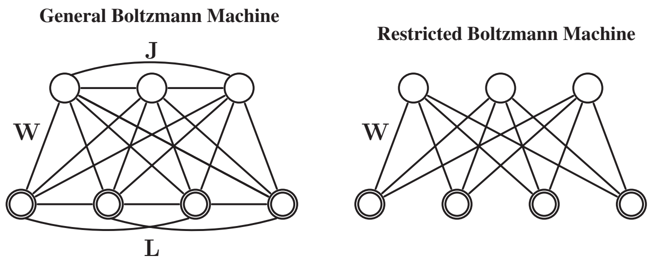

Figure 1: (Left) A general Boltzmann machine. The top layer represents a vector of stochastic binary hidden variables, and the bottom layer represents a vector of stochastic binary visible variables. (Right) A restricted Boltzmann machine with no hidden-to-hidden or visible-to-visible connections.

terms, respectively. The diagonal elements of L and J are set to 0. The probability that the model assigns to a visible vector $\mathbf{v}$ is

$$P(\mathbf{v};\theta) = \frac{P^*(\mathbf{v};\theta)}{\mathcal{Z}(\theta)} = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h};\theta)\right), \tag{2.2}$$

$$\mathcal{Z}(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v},\mathbf{h};\theta)\right), \tag{2.3}$$

where P^* denotes unnormalized probability and $\mathcal{Z}(\theta)$ is the partition function. The conditional distributions over hidden and visible units are given by

$$p(h_j = 1|\mathbf{v},\mathbf{h}_{-j}) = g\left(\sum_i W_{ij} v_i + \sum_{m \neq j} J_{jm} h_m\right), \tag{2.4}$$

$$p(v_i = 1|\mathbf{h},\mathbf{v}_{-i}) = g\left(\sum_j W_{ij} h_j + \sum_{k \neq i} L_{ik} v_k\right), \tag{2.5}$$

where $g(x) = 1/(1+\exp(-x))$ is the logistic function and $\mathbf{x}_{-i}$ denotes a vector $\mathbf{x}$ but with x_i omitted. The parameter updates, originally derived by Hinton and Sejnowski (1983), that are needed to perform gradient ascent in the log likelihood can be obtained from equation 2.2,

$$\Delta W = \alpha\left(\mathbb{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{h}^\top] - \mathbb{E}_{P_{\text{model}}}[\mathbf{v}\mathbf{h}^\top]\right),$$
$$\Delta L = \alpha\left(\mathbb{E}_{P_{\text{data}}}[\mathbf{v}\mathbf{v}^\top] - \mathbb{E}_{P_{\text{model}}}[\mathbf{v}\mathbf{v}^\top]\right),$$
$$\Delta J = \alpha\left(\mathbb{E}_{P_{\text{data}}}[\mathbf{h}\mathbf{h}^\top] - \mathbb{E}_{P_{\text{model}}}[\mathbf{h}\mathbf{h}^\top]\right),$$

where α is a learning rate. $\mathbb{E}_{P_{\text{data}}}[\cdot]$, the data-dependent term, is an expectation with respect to the completed data distribution $P_{\text{data}}(\mathbf{h}, \mathbf{v}; \theta) = P(\mathbf{h}|\mathbf{v}; \theta)P_{\text{data}}(\mathbf{v})$, with $P_{\text{data}}(\mathbf{v}) = \frac{1}{N}\sum_n \delta(\mathbf{v} - \mathbf{v}^n)$ representing the empirical distribution, and $\mathbb{E}_{P_{\text{model}}}[\cdot]$, the data-independent term, is an expectation with respect to the distribution defined by the model (see equation 2.2).

Exact maximum likelihood learning in this model is intractable. The exact computation of the data-dependent expectation takes time that is exponential in the number of hidden units, whereas the exact computation of the model's expectation takes time that is exponential in the number of hidden and visible units.

Setting both $J = 0$ and $L = 0$ recovers the restricted Boltzmann machine (RBM) model (see Figure 1, right panel). Setting only the hidden-to-hidden connections $J = 0$ recovers a semirestricted Boltzmann machine (Osindero & Hinton, 2008) in which inferring the states of the hidden units given the visible states is still very easy but learning is more complicated because it is no longer feasible to infer the states of the visible units exactly when reconstructing the data from the hidden states.

2.1 A Stochastic Approximation Procedure for Estimating the Data-Independent Statistics.

Markov chain Monte Carlo (MCMC) methods belonging to the general class of stochastic approximation algorithms of the Robbins-Monro type (Younes, 1989; Robbins & Monro, 1951) can be used to approximate the data-independent statistics (Younes, 1999; Neal, 1992; Yuille, 2004; Tieleman, 2008). To be more precise, let us consider the following canonical form of the exponential family associated with the sufficient statistics vector Φ:

$$P(\mathbf{x}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \exp(\theta^\top \Phi(\mathbf{x})). \tag{2.6}$$

Given a set of N independent and identically distributed (i.i.d.) training examples $X = \{\mathbf{x}^1, \ldots, \mathbf{x}^N\}$, the derivative of the log likelihood with respect to parameter vector θ takes the form

$$\frac{\partial \log P(X; \theta)}{\partial \theta} = \frac{1}{N}\sum_{n=1}^{N} \Phi(\mathbf{x}^n) - \mathbb{E}_{P_{\text{model}}}[\Phi(\mathbf{x})]. \tag{2.7}$$

The idea behind learning parameter vector θ using stochastic approximation is straightforward. Let θ^t and $\tilde{\mathbf{x}}^t$ be the current parameters and the state. Then $\tilde{\mathbf{x}}^t$ and θ^t are updated sequentially as follows:

- Given $\tilde{\mathbf{x}}^t$, a new state $\tilde{\mathbf{x}}^{t+1}$ is sampled from a transition operator $T_{\theta^t}(\tilde{\mathbf{x}}^{t+1} \leftarrow \tilde{\mathbf{x}}^t)$ that leaves $P(\cdot; \theta^t)$ invariant (e.g., a Gibbs sampler).

Algorithm 1: Stochastic Approximation Algorithm for a Fully Visible Boltzmann Machine.

```
1. Given a data set $X = \{\mathbf{x}^1, ..., \mathbf{x}^N\}$. Randomly initialize $\theta^0$ and $M$ sample particles $\{\tilde{\mathbf{x}}^{0,1}, ...., \tilde{\mathbf{x}}^{0,M}\}$.
2. for $t = 0 : T$ (number of iterations) do
3.     for $i = 1 : M$ (number of parallel Markov chains) do
4.         Sample $\tilde{\mathbf{x}}^{t+1,i}$ given $\tilde{\mathbf{x}}^{t,i}$ using transition operator $T_{\theta^t}(\tilde{\mathbf{x}}^{t+1,i} \leftarrow \tilde{\mathbf{x}}^{t,i})$.
5.     end for
6.     Update: $\theta^{t+1} = \theta^t + \alpha_t \left[\frac{1}{N}\sum_{n=1}^{N}\Phi(\mathbf{x}^n) - \frac{1}{M}\sum_{m=1}^{M}\Phi(\tilde{\mathbf{x}}^{t+1,m})\right]$.
7.     Decrease $\alpha_t$.
8. end for
```

- A new parameter θ^{t+1} is then obtained by replacing the intractable data-independent statistics $\mathbb{E}_{P_{\text{model}}}[\Phi(\mathbf{x})]$ by a point estimate $\Phi(\tilde{\mathbf{x}}^{t+1})$.

In practice, we typically maintain a set of M sample points $\{\tilde{\mathbf{x}}^{t,1}, \ldots, \tilde{\mathbf{x}}^{t,M}\}$, which we often refer to as sample particles. In this case, the intractable data-independent statistics are replaced by the sample averages $1/_M \sum_{m=1}^{M} \Phi(\tilde{\mathbf{x}}^{t+1,m})$. The procedure is summarized in algorithm 1.

The standard proof of convergence of these algorithms relies on the following basic decomposition. First, the gradient of the log-likelihood function takes the form

$$S(\theta) = \frac{\partial \log P(X;\theta)}{\partial\theta} = \frac{1}{N}\sum_{n=1}^{N}\Phi(\mathbf{x}^n) - \mathbb{E}_{P_{\text{model}}}[\Phi(\mathbf{x})]. \tag{2.8}$$

The parameter update rule then takes the following form:

$$\begin{aligned}\theta^{t+1} &= \theta^t + \alpha_t\left[\frac{1}{N}\sum_{n=1}^{N}\Phi(\mathbf{x}^n) - \frac{1}{M}\sum_{m=1}^{M}\Phi(\tilde{\mathbf{x}}^{t+1,m})\right]\\ &= \theta^t + \alpha_t S(\theta^t) + \alpha_t\left[\mathbb{E}_{P_{\text{model}}}[\Phi(\mathbf{x})] - \frac{1}{M}\sum_{m=1}^{M}\Phi(\tilde{\mathbf{x}}^{t+1,m})\right]\\ &= \theta^t + \alpha_t S(\theta^t) + \alpha_t\epsilon_{t+1}.\end{aligned} \tag{2.9}$$

The first term is the discretization of the ordinary differential equation $\dot{\theta} = S(\theta)$. The algorithm is therefore a perturbation of this discretization with the noise term ϵ_{t+1}. The proof then proceeds by showing that the noise term is not too large.

Precise sufficient conditions that ensure almost sure convergence to an asymptotically stable point of $\dot{\theta} = S(\theta)$ are given in Younes (1989, 1999) and Yuille (2004). One necessary condition requires the learning rate to decrease with time, so that $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$. This condition can, for example, be satisfied simply by setting $\alpha_t = a/(b + t)$ for positive constants $a > 0$, $b > 0$. Other conditions ensure that the speed of convergence of the Markov chain, governed by the transition operator T_θ, does not decrease too fast as θ tends to infinity, and that the noise term ϵ_t in the update of equation 2.9 is bounded. Typically, in practice, the sequence $|\theta^t|$ is bounded, and the Markov chain, governed by the transition kernel T_θ, is ergodic. Together with the condition on the learning rate, this ensures almost sure convergence of the stochastic approximation algorithm to an asymptotically stable point of $\dot{\theta} = S(\theta)$.

Informally the intuition behind why this procedure works is the following. As the learning rate becomes sufficiently small compared with the mixing rate of the Markov chain, this persistent chain will always stay very close to the stationary distribution, even if it is run for only a few MCMC steps per parameter update. Samples from the persistent chain will be highly correlated for successive parameter updates, but if the learning rate is sufficiently small, the chain will mix before the parameters have changed enough to significantly alter the value of the estimator.

The success of learning relatively small Boltzmann machines (Neal, 1992) seemed to imply that the learning rate was sufficiently small to allow the chains to stay close to equilibrium as the parameters changed. Recently, however, this explanation has been called into question. After learning an RBM using persistent Markov chains for the data-independent statistics, we tried sampling from the RBM and discovered that even though the learning had produced a good model, the chains mixed extremely slowly. In fact, they mixed so slowly that the appropriate final learning rate, according to the explanation above, would have been smaller by several orders of magnitude than the rate we actually used. So why did the learning work?

Tieleman and Hinton (2009) argue that the fact that the parameters are being updated using the data-independent statistics gathered from the persistent chains means that the mixing rate of the chains with their parameters fixed is not what limits the maximum acceptable learning rate. Consider, for example, a persistent chain that is stuck in a deep local minimum of the energy surface. Assuming that this local minimum has very low probability under the posterior distributions that are used to estimate the data-dependent statistics, the effect of the learning will be to raise the energy of the local minimum. After a number of weight updates, the persistent chain will escape from the local minimum not because the chain has had time to mix but because the energy landscape has changed to make the local minimum much less deep. The learning causes the persistent chains to be repelled from whatever state they are currently in, and this can cause slow-mixing chains to move to other parts of the dynamic energy landscape much faster than would be predicted by the mixing rate with static parameters. Welling (2009) has independently reported a closely related phenomenon that he calls "herding."

Recently Tieleman (2008), Salakhutdinov and Hinton (2009a), Salakhutdinov (2009), and Desjardins, Courville, Bengio, Vincent, and Delalleau (2010) have shown that this stochastic approximation algorithm, also termed persistent contrastive divergence, performs well compared to contrastive divergence at learning good generative models in RBMs. Although the allowable learning rate is much higher than would be predicted from the mixing rate of the persistent Markov chains, it is still considerably lower than the rate used for contrastive divergence learning because the gradient estimate it provides has lower bias but much higher variance. The variance is especially high when using online or mini-batch learning rather than full batch learning. With contrastive divergence learning, the error in the data-dependent statistics introduced by the fact that a mini-batch has sampling error is approximately cancelled by approximating the data-independent statistics using short Markov chains that start at the data in the mini-batch and do not have time to move far away. With persistent contrastive divergence, this strong correlation between the sampling errors in the data-dependent and data-independent statistics is lost.

2.2 A Variational Approach to Estimating the Data-Dependent Statistics. Persistent Markov chains are less appropriate for estimating the data-dependent statistics, especially with mini-batch learning on large data sets. Fortunately, variational approximations work well for estimating the data-dependent statistics. Given the data, it is typically quite reasonable for the posterior distribution over latent variables to be unimodal, especially for applications like speech and vision where normal data vectors really do have a single correct explanation and the data are rich enough to allow that explanation to be inferred using a good generative model.

In variational learning (Zemel, 1993; Hinton & Zemel, 1994; Neal & Hinton, 1998; Jordan et al., 1999), the true posterior distribution over latent variables $P(\mathbf{h}|\mathbf{v};\theta)$ for each training vector $\mathbf{v}$ is replaced by an approximate posterior $Q(\mathbf{h}|\mathbf{v};\mu)$, and the parameters are updated to maximize the variational lower bound on the log likelihood,

$$\begin{aligned}\log P(\mathbf{v};\theta) &\geq \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v};\mu)\log P(\mathbf{v},\mathbf{h};\theta) + \mathcal{H}(Q)\\ &\geq \log P(\mathbf{v};\theta) - \mathrm{KL}\left[Q(\mathbf{h}|\mathbf{v};\mu)||P(\mathbf{h}|\mathbf{v};\theta)\right],\end{aligned} \tag{2.10}$$

where $\mathcal{H}(\cdot)$ is the entropy functional.

Variational learning has the nice property that in addition to trying to maximize the log likelihood of the training data, it tries to find parameters that minimize the Kullback–Leibler divergence between the approximating and true posteriors. Making the true posterior approximately unimodal, even if it means sacrificing some log likelihood, could be advantageous for a system that will use the posterior to control its actions. Having multiple alternative representations of the same sensory input increases the likelihood compared with a single explanation of the same quality, but it makes it more difficult to associate an appropriate action with that sensory input. Variational inference that uses a factorial distribution to approximate the posterior helps to eliminate this problem. During learning, if the posterior given a training input vector is multimodal, the variational inference will lock onto one mode, and learning will make that mode more probable. Our learning algorithm will therefore tend to find regions in the parameter space in which the true posterior is dominated by a single mode.

For simplicity and speed, we approximate the true posterior using a fully factorized distribution (i.e., the naive mean-field approximation), $Q(\mathbf{h}; \mu) = \prod_{j=1}^{\mathcal{U}} q(h_j)$, where $q(h_j = 1) = \mu_j$ and $\mathcal{U}$ is the number of hidden units. The lower bound on the log probability of the data takes the following form:

$$\log P(\mathbf{v}; \theta) \geq \frac{1}{2}\sum_{i,k} L_{ik} v_i v_k + \frac{1}{2}\sum_{j,m} J_{jm}\mu_j\mu_m + \sum_{i,j} W_{ij} v_i \mu_j - \log \mathcal{Z}(\theta)$$
$$- \sum_j [\mu_j \log \mu_j + (1-\mu_j)\log(1-\mu_j)].$$

The learning proceeds by first maximizing this lower bound with respect to the variational parameters μ for fixed θ, which results in the mean-field fixed-point equations:

$$\mu_j \leftarrow g\left(\sum_i W_{ij} v_i + \sum_{m \neq j} J_{mj}\mu_m\right). \tag{2.11}$$

This is followed by applying stochastic approximation to update model parameters θ, which is summarized in algorithm 2.

We emphasize that variational approximations should not be used for estimating the data-independent statistics in the Boltzmann machine learning rule, as attempted in Galland (1991), for two separate reasons. First, a factorial approximation cannot model the highly multimodal, data-independent distribution that is typically required. Second, the minus sign causes the parameters to be adjusted so that the true model distribution becomes as different as possible from the variational approximation.

Algorithm 2: Learning Procedure for a General Boltzmann Machine.

```
1. Given: a training set of $N$ binary data vectors $\{\mathbf{v}\}_{n=1}^{N}$, and $M$, the number of persistent
   Markov chains (i.e., particles).
2. Randomly initialize parameter vector $\theta^0$ and $M$ samples: $\{\tilde{\mathbf{v}}^{0,1}, \tilde{\mathbf{h}}^{0,1}\}, ..., \{\tilde{\mathbf{v}}^{0,M}, \tilde{\mathbf{h}}^{0,M}\}$.
3. for $t = 0$ to $T$ (number of iterations) do
4.    // Variational Inference:
5.    for each training example $\mathbf{v}^n$, $n = 1$ to $N$ do
6.       Randomly initialize $\mu$ and run mean-field updates until convergence:
               $\mu_j \leftarrow g\left(\sum_i W_{ij} v_i + \sum_{m \neq j} J_{mj} \mu_m\right)$.
7.       Set $\mu^n = \mu$.
8.    end for
9.    // Stochastic Approximation:
10.   for each sample $m = 1$ to $M$ (number of persistent Markov chains) do
11.      Sample $(\tilde{\mathbf{v}}^{t+1,m}, \tilde{\mathbf{h}}^{t+1,m})$ given $(\tilde{\mathbf{v}}^{t,m}, \tilde{\mathbf{h}}^{t,m})$ by running a Gibbs sampler
         (see equations 2.4, 2.5)
12.   end for
13.   // Parameter Update:
14.   $W^{t+1} = W^t + \alpha_t \left( \frac{1}{N} \sum_{n=1}^{N} \mathbf{v}^n (\mu^n)^\top - \frac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{v}}^{t+1,m} (\tilde{\mathbf{h}}^{t+1,m})^\top \right)$.
15.   $J^{t+1} = J^t + \alpha_t \left( \frac{1}{N} \sum_{n=1}^{N} \mu^n (\mu^n)^\top - \frac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{h}}^{t+1,m} (\tilde{\mathbf{h}}^{t+1,m})^\top \right)$.
16.   $L^{t+1} = L^t + \alpha_t \left( \frac{1}{N} \sum_{n=1}^{N} \mathbf{v}^n (\mathbf{v}^n)^\top - \frac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{v}}^{t+1,m} (\tilde{\mathbf{v}}^{t+1,m})^\top \right)$.
17.   Decrease $\alpha_t$.
18. end for
```

3 Learning Deep Boltzmann Machines

Algorithm 2 can learn Boltzmann machines with any pattern of connectivity between the units, but it can be made particularly efficient in deep Boltzmann machines that have multiple hidden layers but have connections only between adjacent layers, as shown in Figure 2. Deep Boltzmann machines are interesting for several reasons. First, like deep belief networks, DBMs have the ability to learn internal representations that capture very complex statistical structure in the higher layers. As has already been demonstrated for DBNs, this is a promising way of solving object and speech recognition problems (Bengio, 2009; Bengio & LeCun, 2007; Hinton et al., 2006; Dahl, Ranzato, Mohamed, & Hinton, 2010; Mohamed, Dahl, & Hinton, 2012). High-level representations can be built from a large supply of unlabeled data, and a much smaller supply of labeled data can then be used to fine-tune the model for a specific discrimination task. Second, again like DBNs, if DBMs are learned in the right way, there is a very fast way to initialize

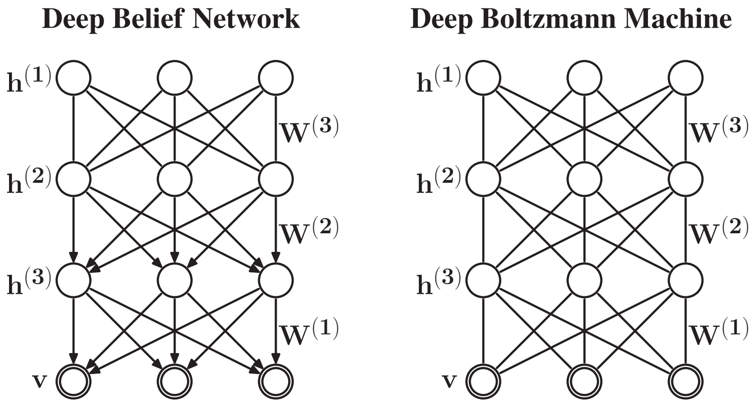

Figure 2: (Left) Deep belief network (DBN). The top two layers form an undirected graph, and the remaining layers form a belief net with directed, top-down connections (Right) Deep Boltzmann machine (DBM), with both visible-to-hidden and hidden-to-hidden connections but no within-layer connections. All the connections in a DBM are undirected.

the states of the units in all layers by simply doing a single bottom-up pass using twice the weights to compensate for the initial lack of top-down feedback. Third, unlike DBNs and many other models with deep architectures (Ranzato, Huang, Boureau, & LeCun, 2007; Vincent, Larochelle, Bengio, & Manzagol, 2008; Serre, Oliva, & Poggio, 2007), the approximate inference procedure, after the initial bottom-up pass, can incorporate top-down feedback. This allows DBMs to use higher-level knowledge to resolve uncertainty about intermediate-level features, thus creating better data-dependent representations and better data-dependent statistics for learning.[3]

Let us consider a three-hidden-layer DBM, as shown in Figure 2 (right), with no within-layer connections. The energy of the state $\{\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}\}$ is defined as

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \theta) = -\mathbf{v}^\top W^{(1)}\mathbf{h}^{(1)} - \mathbf{h}^{(1)\top} W^{(2)}\mathbf{h}^{(2)} - \mathbf{h}^{(2)\top} W^{(3)}\mathbf{h}^{(3)}, \tag{3.1}$$

[3]For many learning procedures, there is a trade-off between the time taken to infer the states of the latent variables and the number of weight updates required to learn a good model. For example, an autoencoder that uses noniterative inference requires more weight updates than an autoencoder that uses iterative inference to perform a look-ahead search for a code that is better at reconstructing the data and satisfying penalty terms (Ranzato, 2009).

where $\theta = \{W^{(1)}, W^{(2)}, W^{(3)}\}$ are the model parameters, representing visible-to-hidden and hidden-to-hidden symmetric interaction terms.

The probability that the model assigns to a visible vector $\mathbf{v}$ is

$$P(\mathbf{v}; \theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}} \exp\left(-E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \theta)\right). \tag{3.2}$$

The conditional distributions over the visible and the three sets of hidden units are given by logistic functions:

$$p(h_j^{(1)} = 1 | \mathbf{v}, \mathbf{h}^{(2)}) = g\left(\sum_i W_{ij}^{(1)} v_i + \sum_m W_{jm}^{(2)} h_m^{(2)}\right), \tag{3.3}$$

$$p(h_m^{(2)} = 1 | \mathbf{h}^{(1)}, \mathbf{h}^{(3)}) = g\left(\sum_j W_{jm}^{(2)} h_j^{(1)} + \sum_l W_{ml}^{(3)} h_l^{(3)}\right), \tag{3.4}$$

$$p(h_l^{(3)} = 1 | \mathbf{h}^{(2)}) = g\left(\sum_m W_{ml}^{(3)} h_m^{(2)}\right), \tag{3.5}$$

$$p(v_i = 1 | \mathbf{h}^{(1)}) = g\left(\sum_j W_{ij}^{(1)} h_j^{(1)}\right). \tag{3.6}$$

The learning procedure for general Boltzmann machines described above can be applied to DBMs that start with randomly initialized weights, but it works much better if the weights are initialized sensibly. With small random weights, hidden units in layers that are far from the data are very underconstrained, so there is no consistent learning signal for their weights. With larger random weights, the initialization imposes a strong random bias on the feature detectors learned in the hidden layers. Even when the ultimate goal is some unknown discrimination task, it is much better to bias these feature detectors toward ones that form a good generative model of the data. We now describe how this can be done.

3.1 Greedy Layerwise Pretraining of DBNs. Hinton et al. (2006) introduced a greedy, layer-by-layer unsupervised learning algorithm that consists of learning a stack of RBMs one layer at a time. After greedy learning, the whole stack can be viewed as a single probabilistic model called a deep belief network. Surprisingly, this composite model is *not* a deep Boltzmann machine. The top two layers form a restricted Boltzmann machine, but the lower layers form a directed sigmoid belief network (see Figure 2, left).

After learning the first RBM in the stack, the generative model can be written as

$$P(\mathbf{v};\theta)=\sum_{\mathbf{h}^{(1)}} P(\mathbf{h}^{(1)};W^{(1)})P(\mathbf{v}|\mathbf{h}^{(1)};W^{(1)}), \tag{3.7}$$

where $P(\mathbf{h}^{(1)};W^{(1)})=\sum_{\mathbf{v}} P(\mathbf{h}^{(1)},\mathbf{v};W^{(1)})$ is a prior over $\mathbf{h}^{(1)}$ that is implicitly defined by $W^{(1)}$. Using the same parameters to define both the prior over $\mathbf{h}^{(1)}$ and the likelihood term $P(\mathbf{v}|\mathbf{h}^{(1)})$ seems like an odd thing to do for those who are more familiar with directed graphical models, but it makes inference much easier and it is only a temporary crutch: the prior over $\mathbf{h}^{(1)}$ defined by $W^{(1)}$ will be thrown away and replaced by a better prior defined by the weights, $W^{(2)}$, of the next RBM in the stack.

The second RBM in the stack attempts to learn a better overall model by leaving $P(\mathbf{v}|\mathbf{h}^{(1)};W^{(1)})$ fixed and replacing $P(\mathbf{h}^{(1)};W^{(1)})$ by

$$P(\mathbf{h}^{(1)};W^{(2)})=\sum_{\mathbf{h}^{(2)}} P(\mathbf{h}^{(1)},\mathbf{h}^{(2)};W^{(2)}),$$

where $W^{(2)}$ is initialized at $W^{(1)\top}$ and then improved by following the gradient of a variational lower bound on the log probability of the training data with respect to $W^{(2)}$. The variational bound was first derived using coding arguments in Hinton and Zemel (1994). For a data set containing N training examples, it has the form

$$\begin{aligned}\sum_{n=1}^{N}\log P(\mathbf{v}^n;\theta) &\geq \sum_n \mathbb{E}_{Q(\mathbf{h}^{(1)}|\mathbf{v}^n)}[\log P(\mathbf{v}^n|\mathbf{h}^{(1)};W^{(1)})]\\ &\quad -\sum_n \mathrm{KL}(Q(\mathbf{h}^{(1)}|\mathbf{v}^n)||P(\mathbf{h}^{(1)};W^{(2)}))\\ &\geq \sum_n\left[\sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}^n)[\log P(\mathbf{v}|\mathbf{h}^{(1)};W^{(1)})]+\mathcal{H}(Q)\right]\\ &\quad +\sum_n\sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}^n)\log P(\mathbf{h}^{(1)};W^{(2)}),\end{aligned} \tag{3.8}$$

where $\mathcal{H}(\cdot)$ is the entropy functional and $Q(\mathbf{h}^{(1)}|\mathbf{v})$ is any approximation to the posterior distribution over vectors in hidden layer 1 for the DBN containing hidden layers $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$. The approximation we use is the true posterior over $\mathbf{h}^{(1)}$ for the first RBM, $Q(\mathbf{h}^{(1)}|\mathbf{v})=P(\mathbf{h}^{(1)}|\mathbf{v};W^{(1)})$. If the second RBM is initialized to be the same as the first RBM but with its visible and hidden units interchanged, $W^{(2)}=W^{(1)\top}$, then $Q(\mathbf{h}^{(1)}|\mathbf{v})$ defines the DBN's true posterior over $\mathbf{h}^{(1)}$ and the variational bound is tight. As soon as $W^{(2)}$ ceases to be identical to $W^{(1)\top}$, Q is no longer the true posterior for the DBN.

Changing $W^{(2)}$ affects only the last sum in equation 3.8, so maximizing the bound, summed over all the training cases, with regard to $W^{(2)}$ amounts to learning a better model of the mixture over all N training cases: $\frac{1}{N}\sum_n Q(\mathbf{h}^{(1)}|\mathbf{v}^n)$, which we call the aggregated posterior. Each individual posterior of the first RBM $Q(\mathbf{h}^{(1)}|\mathbf{v}^n)$ is factorial, but the aggregated posterior is typically very far from factorial. Changing $W^{(2)}$ so that the second RBM becomes a better model of the aggregated posterior over $\mathbf{h}^{(1)}$ is then guaranteed to improve the variational bound for the whole DBN on the log likelihood of the training data.

This argument can be applied recursively to learn as many layers of features as desired. Each RBM in the stack performs exact inference while it is being learned, but once its implicit prior over its hidden vectors has been replaced by a better prior defined by the higher-level RBM, the simple inference procedure ceases to be exact. As the stack gets deeper, the simple inference procedure used for the earlier layers can be expected to become progressively less correct. Nevertheless, each time a new layer is added and learned, the variational bound for the deeper system is better than the bound for its predecessor. When a third hidden layer is added, for example, the bound of equation 3.8 is replaced by a bound in which the last sum,

$$\sum_n \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}^n)\log P(\mathbf{h}^{(1)}; W^{(2)}), \tag{3.9}$$

is replaced by

$$\sum_n \sum_{\mathbf{h}^{(1)}} Q(\mathbf{h}^{(1)}|\mathbf{v}^n)\big(\mathbb{E}_{Q(\mathbf{h}^{(2)}|\mathbf{h}^{(1)})}[\log P(\mathbf{h}^{(1)}|\mathbf{h}^{(2)}; W^{(2)})] - \mathrm{KL}(Q(\mathbf{h}^{(2)}|\mathbf{h}^{(1)})||P(\mathbf{h}^{(2)}; W^{(3)}))\big). \tag{3.10}$$

When the second RBM is learned, the log probability of the training data also improves because the bound starts off tight. For deeper layers, we are still guaranteed to improve the variational lower bound on the log probability of the training data, though the log probability itself can decrease. For these deeper layers, the bound does not start off tight and could therefore become tighter, allowing the log probability to decrease even though the bound increases.

The improvement of the bound is guaranteed only if each RBM in the stack starts with the same weights as the previous RBM and follows the gradient of the log likelihood, using the posterior distributions over the hidden units of the previous RBM as its data. In practice, we violate this condition by using gross approximations to the gradient such as contrastive divergence.

The real value of deriving the variational bound is to allow us to understand why it makes sense to use the aggregated posterior distributions of one RBM as the training data for the next RBM and why the combined model is a deep belief net rather than a deep Boltzmann machine.

3.2 Greedy Layerwise Pretraining of DBMs. Although the simple way of stacking RBMs leads to a deep belief net, it is possible to modify the procedure so that stacking produces a deep Boltzmann machine. We start by giving a crude intuitive argument about how to combine RBMs to get a deep Boltzmann machine. We then show that a modification of the intuitively derived method for adding the top layer is guaranteed to improve a variational bound. We also show that the procedure we use in practice for adding intermediate layers fails to achieve the property that is required for guaranteeing an improvement in the variational bound.

After training the second-layer RBM in a deep belief net, there are two ways of computing a factorial approximation to the true posterior $P(\mathbf{h}^{(1)}|\mathbf{v}; W^{(1)}, W^{(2)})$. The obvious way is to ignore the second-layer RBM and use the $P(\mathbf{h}^{(1)}|\mathbf{v}; W^{(1)})$ defined by the first RBM. An alternative method is to first sample $\mathbf{h}^{(1)}$ from $P(\mathbf{h}^{(1)}|\mathbf{v}; W^{(1)})$, then sample $\mathbf{h}^{(2)}$ from $P(\mathbf{h}^{(2)}|\mathbf{h}^{(1)}; W^{(2)})$, and then use the $P(\mathbf{h}^{(1)}|\mathbf{h}^{(2)}; W^{(2)})$ defined by the second RBM.[4] The second method will tend to overemphasize the prior for $\mathbf{h}^{(1)}$ defined by $W^{(2)}$, whereas the first method will tend to underemphasize this prior in favor of the earlier prior defined by $W^{(1)}$ that it replaced.

Given these two different approximations to the posterior, it would be possible to take a geometric average of the two distributions. This can be done by first performing a bottom-up pass to infer $\mathbf{h}^{(2)}$ then using $^1/_2 W^{(1)}$ and $^1/_2 W^{(2)}$ to infer $\mathbf{h}^{(1)}$ from both $\mathbf{v}$ and $\mathbf{h}^{(2)}$. Notice that $\mathbf{h}^{(2)}$ is inferred from $\mathbf{v}$, so it is not legitimate to sum the full top-down and bottom-up influences. This would amount to double-counting the evidence provided by $\mathbf{v}$ and would give a distribution that was much too sharp. Experiments with trained DBNs confirm that averaging the top-down and bottom-up inputs works well for inference and adding them works badly.

This reasoning can be extended to a much deeper stack of greedily trained RBMs. The initial bottom-up inference that is performed in a DBN can be followed by a stage in which all of the weights are halved and the states of the units in the intermediate layers are resampled by summing the top-down and bottom-up inputs to a layer. If we alternate between resampling the odd-numbered layers and resampling the even-numbered layers, this corresponds to alternating Gibbs sampling in a deep Boltzmann machine with the visible units clamped. So after learning a stack of RBMs, we can either compose them to form a DBN or halve all the weights and compose

[4]The sampling noise in the second method can be reduced by using a further approximation in which the sampled binary values are replaced by their probabilities.

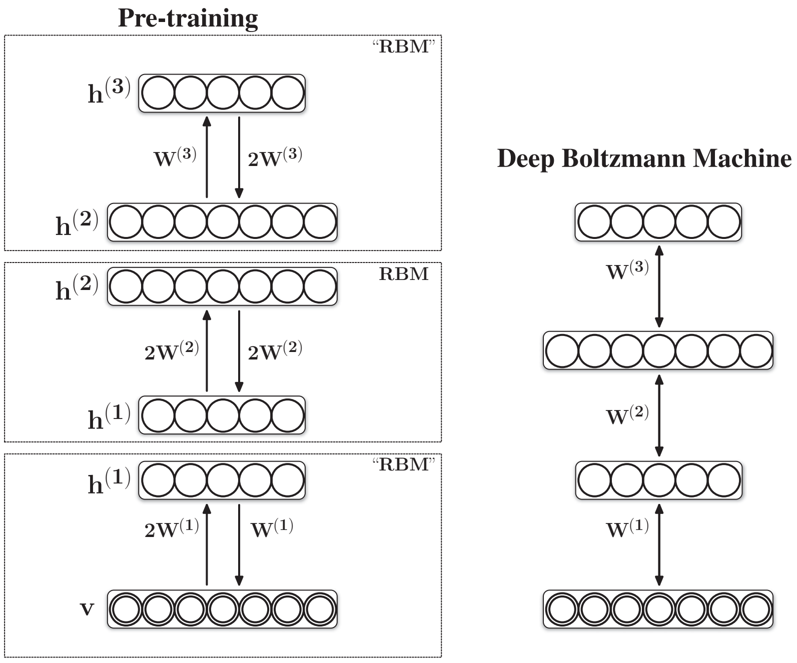

Figure 3: Pretraining a DBM with three hidden layers consists of learning a stack of RBMs that are then composed to create a DBM. The first and last RBMs in the stack need to be modified by using weights that are twice as big in one direction.

them to form a DBM. Moreover, given the way the DBM was created, there is a very fast way to initialize all of the hidden layers when given a data vector: simply perform a bottom-up pass using twice the weights of the DBM to compensate for the lack of top-down input.

There is an annoying problem with this method of pretraining a DBM. For the intermediate layers, using weights that are half of the weights learned by the individual RBMs seems fine because when the RBMs are combined, it can be viewed as taking the geometric mean of the bottom-up and top-down models, but for the visible layer and the top layer, it is not legitimate because they receive input only from one other layer. Both the top and the bottom layers need to be updated when estimating the data-independent statistics for the DBM, and we cannot use weights that are bigger in one direction than the other because this does not correspond to Gibbs sampling in any energy function. So we need to use a special trick when pretraining the first and last RBMs in the stack.

For the first RBM, we constrain the bottom-up weights to be twice the top-down weights during the pretraining, as shown in Figure 3. This means that we can halve the bottom-up weights without halving the top-down weights and still be left with symmetric weights. Conversely, for the last RBM in the stack, we constrain the top-down weights to be twice the bottom-up weights. Now we have a stack of RBMs that we can convert to a DBM by halving all but the first layer top-down weights and the last layer bottom-up weights.[5]

Greedily pretraining the weights of a DBM in this way serves two purposes. First, it initializes the weights to reasonable values. Second, it ensures that there is a fast way of performing approximate inference by a single bottom-up pass using twice the weights for all but the top-most layer. This eliminates the need to store the hidden states that were inferred last time a training case was used (Neal, 1992) or to use simulated annealing from random initial states of the hidden units (Hinton & Sejnowski, 1983). This fast approximate inference is used to initialize the mean-field, iterative inference, which then converges much faster than mean field with random initialization. Since the mean-field inference uses real-valued probabilities rather than sampled binary states, we also use probabilities rather than sampled binary states for the initial bottom-up inference.

3.3 A Variational Bound for Greedy Layerwise Pretraining of a DBM with Two Hidden Layers. The explanation of DBM pretraining given in the previous section is motivated by the need to end up with a deep network that has symmetric weights between all adjacent pairs of layers. However, unlike the pretraining of a DBN, it lacks proof that each time a layer is added to the DBM, the variational bound for the deeper DBM is better than the bound for the previous DBM. We now give an explanation of why the method for training the first RBM in the stack works, and we show that with a slight modification, the proposed method for training the second layer RBM is a correct way of improving a variational bound. The main point of this exercise is to gain a better understanding of how the pretraining procedures for DBNs and DBMs are related. The apparently unproblematic method for pretraining the intermediate layers using a symmetric RBM is actually more problematic than it seems, and we discuss this in section 3.4.

The basic idea for pretraining a DBM is to start by learning a model in which the prior over hidden vectors, $p(\mathbf{h}^{(1)}; W^{(1)})$, is the normalized product of two identical distributions. Then one of these distributions is discarded and replaced by the square root of a better prior $p(\mathbf{h}^{(1)}; W^{(2)})$ that has been trained to fit a good approximation to the aggregated posterior of the first model.

[5]Of course, when we constrain the weights of an RBM to have different magnitudes in the two directions, the usual rules for updating the states of the units no longer correspond to alternating Gibbs sampling in any energy function, but the one-step contrastive divergence learning still works well, for reasons that will be explained later.

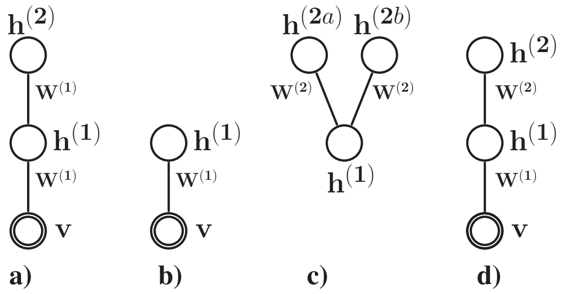

Figure 4: Pretraining a deep Boltzmann machine with two hidden layers. (a) The DBM with tied weights is trained to model the data using one-step contrastive divergence (see Figure 5). (b) The second hidden layer is removed. (c) The second hidden layer is replaced by part of the RBM that has been trained on the aggregated posterior distribution at the first hidden layer of the DBM in panel a. (d) The resulting DBM with a modified second hidden layer.

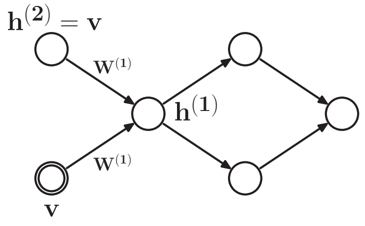

Figure 5: Pretraining the DBM with two hidden layers shown in Figure 4a using one-step contrastive divergence. The second hidden layer is initialized to be the same as the observed data. The units in the first hidden layer have stochastic binary states, but the reconstructions of both the visible and second hidden layer use the unsampled probabilities, so both reconstructions are identical. The pairwise statistics for the visible layer and the first hidden layer are therefore identical to the pairwise statistics for the second hidden layer and the first hidden layer, and this is true for both the data and the reconstructions.

Consider the simple DBM in Figure 4a that has two hidden layers and tied weights. If we knew what initial state vector to use for $\mathbf{h}^{(2)}$, we could train this DBM using one-step contrastive divergence with mean-field reconstructions of both the visible states and the states of the top layer, as shown in Figure 5. So we simply set the initial state vector of the top layer to be equal to the data, $\mathbf{v}$. Provided we use mean-field reconstructions for the visible units and the top-layer units, one-step contrastive divergence is then exactly equivalent to training an RBM with only one hidden layer but with bottom-up weights that are twice the top-down weights, as prescribed

Algorithm 3: Greedy Pretraining Algorithm for a Deep Boltzmann Machine.

```
1. Train the first-layer "RBM" using one-step contrastive divergence learning with mean-field
   reconstructions of the visible vectors. During the learning, constrain the bottom-up weights,
   $2W^{(1)}$, to be twice the top-down weights, $W^{(1)}$.
2. Freeze $2W^{(1)}$ that defines the first layer of features and use samples $\mathbf{h}^{(1)}$ from
   $P(\mathbf{h}^{(1)}|\mathbf{v}; 2W^{(1)})$ as the data for training the second RBM. This is a proper RBM with
   weights $2W^{(2)}$ that are of the same magnitude in both directions. It is also trained us-
   ing one-step contrastive divergence learning with mean-field reconstructions of its visible
   vectors.
3. Freeze $2W^{(2)}$ that defines the second layer of features and use the samples $\mathbf{h}^{(2)}$
   from $P(\mathbf{h}^{(2)}|\mathbf{v}; 2W^{(1)}, 2W^{(2)})$ as the data for training the next RBM in the same way
   as the previous one.
4. Proceed recursively up to layer $L-1$.
5. Train the top-level RBM using one-step contrastive divergence learning with mean-field
   reconstructions of its visible vectors. During the learning, constrain the bottom-up weights,
   $W^{(L)}$, to be half the top-down weights, $2W^{(L)}$.
6. Use the weights $\{W^{(1)}, W^{(2)}, W^{(3)}, ..., W^{(L)}\}$ to compose a deep Boltzmann machine.
```

in algorithm 3. This way of training the simple DBM with tied weights is unlikely to maximize the likelihood of the weights, but in practice it produces surprisingly good models that reconstruct the training data well.

After pretraining, we can use $2W^{(1)}$ to compute an approximate posterior $Q(\mathbf{h}^{(1)}|\mathbf{v})$ for $\mathbf{h}^{(1)}$ from $\mathbf{v}$, and using this approximate posterior, we can get a variational lower bound on the log probability (see equation 3.8), that the simple DBM in Figure 4a assigns to the training data:

$$\sum_n \log P(\mathbf{v}_n) \geq \sum_n \mathbb{E}_{Q(\mathbf{h}^{(1)}|\mathbf{v}_n)}[\log P(\mathbf{v}_n|\mathbf{h}^{(1)}; W^{(1)})] - \sum_n \mathrm{KL}(Q(\mathbf{h}^{(1)}|\mathbf{v}_n)||P(\mathbf{h}^{(1)}; W^{(1)})). \tag{3.11}$$

We now show how to improve this variational bound.

The model's marginal distribution over $\mathbf{h}^{(1)}$ is the product of two identical distributions—one defined by an RBM composed of $\mathbf{h}^{(1)}$ and $\mathbf{v}$ and the other defined by an identical RBM composed of $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(2)}$:

$$P(\mathbf{h}^{(1)}; W^{(1)}) = \frac{1}{\mathcal{Z}(W^{(1)})} \left(\sum_{\mathbf{v}} e^{\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)}} \right) \left(\sum_{\mathbf{h}^{(2)}} e^{\mathbf{h}^{(2)\top} W^{(1)} \mathbf{h}^{(1)}} \right), \tag{3.12}$$

where $\mathcal{Z}(W^{(1)})$ is the normalizing constant.[6] The idea is to keep one of these two RBMs and replace the other by the square root of a better prior $P(\mathbf{h}^{(1)}; W^{(2)})$. To do so, we train another RBM with two sets of hidden units and tied weights, as shown in Figure 4c, to be a better model of the aggregated variational posterior $\frac{1}{N}\sum_n Q(\mathbf{h}^{(1)}|\mathbf{v}_n; W^{(1)})$ of the first model (see Figure 4a). If the tied weights are initialized at $W^{(2)} = W^{(1)\top}$, then the higher-level RBM has exactly the same prior over $\mathbf{h}^{(1)}$ as the original DBM. If the RBM is trained by following the gradient of the log likelihood,[7] any change in the weights will ensure that

$$\sum_n \mathrm{KL}(Q(\mathbf{h}^{(1)}|\mathbf{v}_n; W^{(1)})||P(\mathbf{h}^{(1)}; W^{(2)})) \\ \leq \sum_n \mathrm{KL}(Q(\mathbf{h}^{(1)}|\mathbf{v}_n; W^{(1)})||P(\mathbf{h}^{(1)}; W^{(1)})). \tag{3.13}$$

Similar to equation 3.12, the distribution over $\mathbf{h}^{(1)}$ defined by the second-layer RBM is also the product of two identical distributions, one for each set of hidden units. This implies that taking a square root amounts to simply keeping one such distribution.

Once the two RBMs are composed to form a two-hidden-layer DBM model (see Figure 4d), the marginal distribution over $\mathbf{h}^{(1)}$ is the geometric mean of the two probability distributions, $P(\mathbf{h}^{(1)}; W^{(1)})$, $P(\mathbf{h}^{(1)}; W^{(2)})$, defined by the first- and second-layer RBMs (i.e., the renormalized pairwise products of the square roots of the two probabilities for each event):

$$P(\mathbf{h}^{(1)}; W^{(1)}, W^{(2)}) = \frac{1}{\mathcal{Z}(W^{(1)}, W^{(2)})}\left(\sum_{\mathbf{v}} e^{\mathbf{v}^\top W^{(1)}\mathbf{h}^{(1)}}\right)\left(\sum_{\mathbf{h}^{(2)}} e^{\mathbf{h}^{(1)\top} W^{(2)}\mathbf{h}^{(2)}}\right). \tag{3.14}$$

The variational lower bound of equation 3.11 improves because replacing half of the prior by a better model reduces the Kullback-Leibler divergence from the variational posterior, as shown in the appendix.

Due to the convexity of asymmetric divergence, this is guaranteed to improve the variational bound of the training data by at least half as much

[6]The biases learned for $\mathbf{h}^{(1)}$ are shared equally between the two RBMs.

[7]In practice, we depart from maximum likelihood training is several ways: we use one-step contrastive divergence to train the second-layer RBM; we use mean-field reconstructions of its visible units; we do not sample its two sets of hidden units independently. But the same issue arises with the variational bound for DBNs. That bound also requires proper maximum likelihood training of the higher-level RBMs. Fortunately, during pretraining, we typically terminate the learning of each RBM long before it converges, and in this early regime, contrastive divergence learning is almost always improving the likelihood, which is all we need.

as fully replacing the original prior. It is also guaranteed to loosen the variational bound by at most half as much as fully replacing the original prior, assuming that inference is still performed assuming the original prior.

This argument shows that the apparently unprincipled hack of doubling the weights in one direction to cope with the "end effects" when creating a two-hidden-layer DBM from a stack of two RBMs can actually be justified as a way of improving a variational bound, except for the fact that the top RBM should be trained by maximum likelihood.

3.4 Pretraining Intermediate Hidden Layers. We have not been able to design a way of adding intermediate hidden layers that is guaranteed to improve a variational bound, but the scheme we use in algorithm 3 seems to work well in practice. One difficulty in extending the proof is that for an intermediate RBM, we need to take the square root of the marginal prior distribution over its visible units so that we can use it to replace half of the previous prior of the lower RBM. We also need to take the square root of the marginal prior distribution over the hidden units of the intermediate RBM in order to allow the next RBM to replace half of this prior.

Halving the weights of an RBM halves the energy of every joint configuration of the hidden and visible units, so it takes the square root of the joint distribution over pairs of visible and hidden vectors, and it also takes the square root of the conditional distribution over visible vectors given a hidden vector (or vice versa), but it does not take the square root of the marginal prior distributions over the visible or the hidden vectors. This is most easily seen by considering the ratios of the probabilities of two visible vectors, $\mathbf{v}_\alpha$ and $\mathbf{v}_\beta$. Before halving the weights, their probability ratio in the marginal distribution over visible vectors is given by

$$\frac{P(\mathbf{v}_\alpha)}{P(\mathbf{v}_\beta)} = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}_\alpha,\mathbf{h})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{v}_\beta,\mathbf{h})}}. \tag{3.15}$$

In the RBM with halved weights, all of the exponents are halved which takes the square root of every individual term in each sum, but this does not take the square root of the ratio of the sums. This argument shows that the apparently unproblematic idea of halving the weights of all the intermediate RBMs in the stack is not the right thing to do if we want to ensure that as each layer is added, the new variational bound is better than the old one.[8]

[8]Removing one of the hidden groups in Figure 4c halves the expected input to the visible units, but it also halves the entropy of the hidden units. This halves the free energy of each visible vector, which takes the square root of the marginal prior distribution over the visible units.

Although we cannot guarantee that a variational bound is improved when adding intermediate layers, the idea that each new RBM is approximately replacing half of the prior over the top layer of the previous model is a useful insight into what is happening during pretraining. It suggests, for example, that each new layer does less work in a DBM than it does in a DBN, which replaces all of the prior of the previous RBM. This in turn suggests that in the pretraining phase, DBMs are likely to get less benefit than DBNs from being very deep. A different method of pretraining DBMs that distributes the modeling work more evenly over the layers would probably be helpful.

4 Evaluating Deep Boltzmann Machines

Assessing the generalization performance of DBMs plays an important role in model selection, model comparison, and controlling model complexity. In this section we discuss two ways of evaluating the generalization capabilities of DBMs: generative and discriminative.

4.1 Evaluating DBMs as Generative Models. We first focus on evaluating generalization performance of DBMs as density models. For many specific tasks, such as classification or information retrieval, performance of DBMs can be directly evaluated (see section 4.2). More broadly, however, the ability of DBMs to generalize can be evaluated by computing the probability of held-out input vectors, which is independent of any specific application. An unfortunate limitation of DBMs is that the exact computation of the probability that the model assigns to a visible vector $\mathbf{v}$ is intractable. It requires both an intractable summation over the hidden vectors at all layers and an intractable computation of the partition function:

$$P(\mathbf{v};\theta) = \frac{1}{\mathcal{Z}(\theta)} \sum_{\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)}} \exp\left(-E(\mathbf{v},\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)};\theta)\right). \tag{4.1}$$

Recently, Salakhutdinov and Murray (2008) showed that a Monte Carlo–based method, annealed importance sampling (AIS; Neal, 2001), can be used to efficiently estimate the partition function of an RBM. In this section, we show how AIS can be used to estimate the partition functions of DBMs. Together with variational inference, this will allow us to obtain good estimates of the lower bound on the log probability of the training and test data.

Suppose we have two distributions defined on some space $\mathcal{X}$ with probability density functions $P_A(\mathbf{x}) = P_A^*(\mathbf{x})/\mathcal{Z}_A$ and $P_B(\mathbf{x}) = P_B^*(\mathbf{x})/\mathcal{Z}_B$. Typically $P_A(\mathbf{x})$ is defined to be some simple distribution, with known partition function $\mathcal{Z}_A$, from which we can easily draw i.i.d. samples. AIS estimates the ratio $\mathcal{Z}_B/\mathcal{Z}_A$ by defining a sequence of intermediate probability distributions:

$P_0, P_1, \ldots, P_K$, with $P_0 = P_A$ and $P_K = P_B$, which satisfy $P_k(\mathbf{x}) \neq 0$ whenever $P_{k+1}(\mathbf{x}) \neq 0$, $k = 0, .., K-1$. For each intermediate distribution, we must be able to easily evaluate the unnormalized probability $P_k^*(\mathbf{x})$, and we must also be able to sample $\mathbf{x}'$ given $\mathbf{x}$ using a Markov chain transition operator $T_k(\mathbf{x}' \leftarrow \mathbf{x})$ that leaves $P_k(\mathbf{x})$ invariant. One general way to define this sequence is to set

$$P_k(\mathbf{x}) \propto P_A^*(\mathbf{x})^{1-\beta_k} P_B^*(\mathbf{x})^{\beta_k}, \tag{4.2}$$

with $0 = \beta_0 < \beta_1 < \cdots < \beta_K = 1$ chosen by the user.

Using the special layer-by-layer structure of DBMs, we can derive an efficient AIS scheme for estimating the model's partition function. Let us consider a three-hidden-layer Boltzmann Machine (see Figure 3, right) whose energy is defined as

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \theta) = -\mathbf{v}^\top W^{(1)} \mathbf{h}^{(1)} - \mathbf{h}^{(1)\top} W^{(2)} \mathbf{h}^{(2)} - \mathbf{h}^{(2)\top} W^{(3)} \mathbf{h}^{(3)}. \tag{4.3}$$

By summing out the first- and the third-layer hidden units $\{\mathbf{h}^{(1)}, \mathbf{h}^{(3)}\}$, we can easily evaluate an unnormalized probability $P^*(\mathbf{v}, \mathbf{h}^{(2)}; \theta)$. We can therefore run AIS on a much smaller state space $\mathbf{x} = \{\mathbf{v}, \mathbf{h}^{(2)}\}$ with $\mathbf{h}^{(1)}$ and $\mathbf{h}^{(3)}$ analytically summed out. The sequence of intermediate distributions, parameterized by β, is defined as

$$\begin{aligned} &P_k(\mathbf{v}, \mathbf{h}^{(2)}; \theta) \\ &\quad = \frac{1}{\mathcal{Z}_k} P_k^*(\mathbf{v}, \mathbf{h}^{(2)}; \theta) = \frac{1}{\mathcal{Z}_k} \sum_{\mathbf{h}^{(1)}, \mathbf{h}^{(3)}} P_k^*(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \theta) \\ &\quad = \frac{1}{\mathcal{Z}_k} \prod_j \left(1 + e^{\beta_k \left(\sum_i v_i W_{ij}^{(1)} + \sum_m h_m^{(2)} W_{jm}^{(2)}\right)}\right) \prod_l \left(1 + e^{\beta_k \left(\sum_m h_m^{(2)} W_{ml}^{(3)}\right)}\right). \end{aligned} \tag{4.4}$$

We gradually change β_k (the inverse temperature) from 0 to 1, annealing from a simple "uniform" model P_0 to the complex deep Boltzmann machine model P_K.

Using equations 3.3 to 3.6, it is straightforward to derive a Gibbs transition operator $T_k(\{\mathbf{v}, \mathbf{h}^{(2)}\}' \leftarrow \{\mathbf{v}, \mathbf{h}^{(2)}\})$ that leaves $P_k(\mathbf{v}, \mathbf{h}^{(2)}; \theta)$ invariant:

$$p(h_j^{(1)} = 1 | \mathbf{v}, \mathbf{h}^{(2)}) = g\left(\beta_k \left(\sum_i W_{ij}^{(1)} v_i + \sum_m W_{jm}^{(2)} h_m^{(2)}\right)\right), \tag{4.5}$$

$$p(h_m^{(2)} = 1 | \mathbf{h}^{(1)}, \mathbf{h}^{(3)}) = g\left(\beta_k \left(\sum_j W_{jm}^{(2)} h_j^{(1)} + \sum_l W_{ml}^{(3)} h_l^{(3)}\right)\right), \tag{4.6}$$

Algorithm 4: Annealed Importance Sampling Run.

```
1. Given a sequence $0 = \beta_0 < \beta_1 < ... < \beta_K = 1$. Let $\mathbf{x} = \{\mathbf{v}, \mathbf{h}^{(2)}\}$.
2. Sample $\mathbf{x}^1$ from $P_0(\mathbf{x})$.
3. for $k = 2 : K$ do
4.     Sample $\mathbf{x}^k$ given $\mathbf{x}^{k-1}$ using $T_{k-1}(\mathbf{x}^k \leftarrow \mathbf{x}^{k-1})$ (see equations 4.5–4.8).
5. end for
6. Compute importance weight using equation 4.4:
       $w^{(i)} = \frac{P_1^*(\mathbf{x}^1)}{P_0^*(\mathbf{x}^1)} \frac{P_2^*(\mathbf{x}^2)}{P_1^*(\mathbf{x}^2)} \cdots \frac{P_{K-1}^*(\mathbf{x}^{K-1})}{P_{K-2}^*(\mathbf{x}^{K-1})} \frac{P_K^*(\mathbf{x}^K)}{P_{K-1}^*(\mathbf{x}^K)}.$
```

$$p(h_l^{(3)} = 1|\mathbf{h}^{(2)}) = g\left(\beta_k \sum_m W_{ml}^{(3)} h_m^{(2)}\right), \tag{4.7}$$

$$p(v_i = 1|\mathbf{h}^{(1)}) = g\left(\beta_k \sum_j W_{ij}^{(1)} h_j^{(1)}\right). \tag{4.8}$$

Algorithm 4 shows a single run of AIS. Note that there is no need to compute the normalizing constants of any intermediate distributions. After performing M runs of AIS, the importance weights $w^{(i)}$ can be used to obtain an estimate of the ratio of partition functions:

$$\frac{\mathcal{Z}_K}{\mathcal{Z}_0} \approx \frac{1}{M} \sum_{i=1}^{M} w^{(i)} = \hat{r}_{\text{AIS}}, \tag{4.9}$$

where $\mathcal{Z}_0 = 2^{|\mathcal{V}|+|\mathcal{U}|}$ is the partition function of the uniform model (e.g., $\beta_0 = 0$) with $|\mathcal{V}|$ and $|\mathcal{U}|$ denoting the total number of visible and hidden units, and $\mathcal{Z}_K$ is the partition function of the DBM model (e.g., $\beta_K = 1$).

Provided K is kept large, the total amount of computation can be split in any way between the number of intermediate distributions K and the number of annealing runs M without adversely affecting the accuracy of the estimator. The number of AIS runs can be used to control the variance in the estimate of $\hat{r}_{\text{AIS}}$, as samples drawn from P_0 are independent:

$$\text{Var}(\hat{r}_{\text{AIS}}) = \frac{1}{M}\text{Var}(w^{(i)}) \approx \frac{\hat{s}^2}{M} = \hat{\sigma}^2, \tag{4.10}$$

where $\hat{s}^2$ is estimated simply from the sample variance of the importance weights.

Finally, once we obtain an estimate of the global partition function $\hat{\mathcal{Z}}$, we can estimate, for a given test case $\mathbf{v}^*$, the variational lower bound of equation 2.10:

$$\log P(\mathbf{v}^*;\theta) \geq -\sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v}^*;\mu)E(\mathbf{v}^*,\mathbf{h};\theta) + \mathcal{H}(Q) - \log \mathcal{Z}(\theta)$$
$$\approx -\sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v}^*;\mu)E(\mathbf{v}^*,\mathbf{h};\theta) + \mathcal{H}(Q) - \log \hat{\mathcal{Z}},$$

where we defined $\mathbf{h} = \{\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}\}$. For each test vector under consideration, this lower bound is maximized with respect to the variational parameters μ using the mean-field update equations.

Furthermore, by explicitly summing out the states of the hidden units $\{\mathbf{h}^{(2)}, \mathbf{h}^{(3)}\}$, we can obtain a tighter variational lower bound on the log probability of the test data. Of course, we can also adopt AIS to estimate $P^*(\mathbf{v}) = \sum_{\mathbf{h}^{(1)},\mathbf{h}^{(2)},\mathbf{h}^{(3)}} P^*(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)})$, and together with an estimate of the global partition function, we can estimate the true log probability of the test data. This, however, would be computationally very expensive, since we would need to perform a separate AIS run for each test case. As an alternative, we could adopt a variation of the Chib-style estimator, proposed by Murray and Salakhutdinov (2009). In the case of deep Boltzmann machines, where the posterior over the hidden units tends to be unimodal, their proposed Chib-style estimator can provide good estimates of $\log P^*(\mathbf{v})$ in a reasonable amount of computer time.

In general, when learning a deep Boltzmann machine with more than two hidden layers and no within-layer connections, we can explicitly sum out either odd or even layers. This will result in a better estimate of the model's partition function and tighter lower bounds on the log probability of the test data.

4.2 Evaluating DBMs as Discriminative Models. After learning, the stochastic activities of the binary features in each layer can be replaced by deterministic, real-valued probabilities, and a deep Boltzmann machine with two hidden layers can be used to initialize a multilayer neural network in the following way. For each input vector $\mathbf{v}$, the mean-field inference is used to obtain an approximate posterior distribution $Q(\mathbf{h}^{(2)}|\mathbf{v})$. The marginals $q(h_j^{(2)} = 1|\mathbf{v})$ of this approximate posterior, together with the data, are used to create an augmented input for this deep multilayer neural network, as shown in Figure 6. Standard backpropagation of error derivatives can then be used to discriminatively fine-tune the model.[9]

[9]Note that one can also backpropagate through the "unfolded" mean field, as shown in Figure 6, middle panel. In our experiments, however, this did not improve model performance.

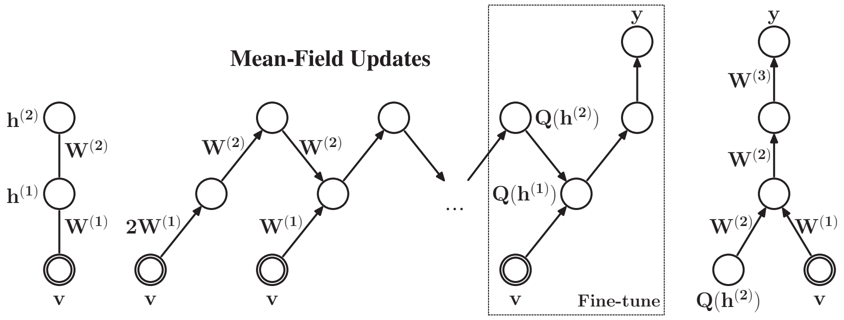

Figure 6: (Left) A two-hidden-layer Boltzmann machine. (Right) After learning, the DBM model is used to initialize a multilayer neural network. The marginals of approximate posterior $q(h_j^{(2)} = 1|\mathbf{v})$ are used as additional inputs. The network is fine-tuned by backpropagation.

The unusual representation of the input is a by-product of converting a DBM into a deterministic neural network. In general, the gradient-based fine-tuning may choose to ignore $Q(\mathbf{h}^{(2)}|\mathbf{v})$, that is, drive the first-layer connections $W^{(2)}$ to zero, which will result in a standard neural network. Conversely, the network may choose to ignore the input by driving the first-layer weights $W^{(1)}$ to zero and make its predictions based on only the approximate posterior. However, the network typically makes use of the entire augmented input for making predictions.

5 Experimental Results

In our experiments, we used the MNIST and NORB data sets. To speed up learning, we subdivided data sets into mini-batches, each containing 100 cases, and updated the weights after each mini-batch. The number of sample particles, used for approximating the model's expected sufficient statistics, was also set to 100. For the stochastic approximation algorithm, we always used five full Gibbs updates of the particles. Each model was trained using 300,000 weight updates. The initial learning rate was set at 0.005 and was decreased as $10/(2000 + t)$, where t is the number of updates so far. For discriminative fine-tuning of DBMs, we used the method of conjugate gradients on larger mini-batches of 5000 with three line searches performed for each mini-batch. (Details of pretraining and fine-tuning, along with details of Matlab code that we used for learning and fine-tuning deep Boltzmann machines, can be found online at http://www.utstat.toronto.edu/~rsalakhu/DBM.html.)

5.1 MNIST. The MNIST digit data set contains 60,000 training and 10,000 test images of 10 handwritten digits (0 to 9), with 28×28 pixels.

Table 1: Results of Estimating Partition Functions of an RBM, One Fully Connected BM, and Two DBM Models, Along with the Estimates of the Lower Bound on the Average Training and Test Log Probabilities.

	Estimates		Average Log Probability	
	$\log \hat{\mathcal{Z}}$	$\log(\hat{\mathcal{Z}} \pm \hat{\sigma})$	Test	Train
RBM	390.76	390.56, 390.92	−86.90	−84.67
Flat BM	198.29	198.17, 198.40	−84.67	−84.35
2-layer BM	356.18	356.06 , 356.29	−84.62	−83.61
3-layer BM	456.57	456.34 , 456.75	−85.10	−84.49

Notes: For all BMs we used 20,000 intermediate distributions. Results were averaged over 100 AIS runs.

Intermediate intensities between 0 and 255 were treated as probabilities, and each time an image was used, we sampled binary values from these probabilities independently for each pixel.

In our first experiment, we trained a fully connected flat BM on the MNIST data set. The model had 500 hidden units and 784 visible units. To estimate the model's partition function, we used 20,000 β_k spaced uniformly from 0 to 1. Results are shown in Table 1, where for all models, we used $M = 100$ AIS runs (see equation 4.9). The estimate of the lower bound on the average test log probability was −84.67 per test case, which is slightly better compared to the lower bound of −85.97, achieved by a carefully trained two-hidden-layer deep belief network (Salakhutdinov & Murray, 2008).

In our second experiment, we trained two deep Boltzmann machines: one with two hidden layers (500 and 1000 hidden units) and the other with three hidden layers (500, 500, and 1000 hidden units), as shown in Figure 8. To estimate the model's partition function, we also used 20,000 intermediate distributions spaced uniformly from 0 to 1. Table 1 shows that the estimates of the lower bound on the average test log probability were −84.62 and −85.10 for the two- and three-layer Boltzmann machines, respectively. Observe that although the two DBMs contain about 0.9 and 1.15 million parameters, they do not appear to suffer much from overfitting. The difference between the estimates of the training and test log probabilities was about 1 nat. Figure 7 further shows samples generated from all three models by randomly initializing all binary states and running the Gibbs sampler for 100,000 steps. Certainly all samples look like the real handwritten digits. We also emphasize that without greedy pretraining, we could not successfully learn good DBM models of MNIST digits.

To estimate how loose the variational bound is, we randomly sampled 100 test cases, 10 of each class, and ran AIS to estimate the true test log

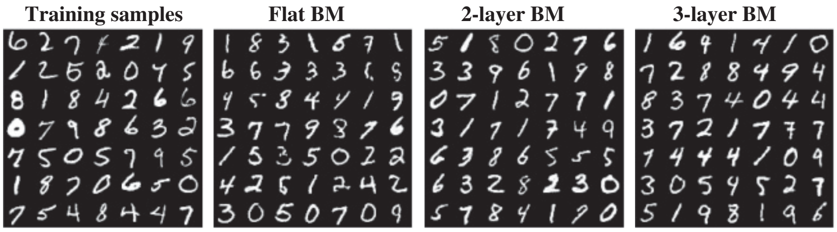

Figure 7: Random samples from the training set, and samples generated from three Boltzmann machines by running the Gibbs sampler for 100,000 steps. The images shown are the probabilities of the binary visible units given the binary states of the hidden units.

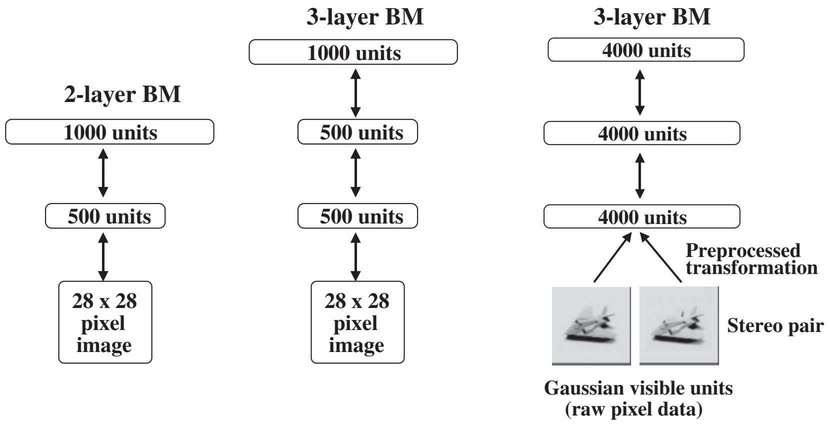

Figure 8: (Left) The architectures of two deep Boltzmann machines used in MNIST experiments. (Right) The architecture of deep Boltzmann machine used in NORB experiments.

probability for the two-layer Boltzmann machine.[10] The estimate of the variational bound was −83.35 per test case with an error estimate (−83.21, −83.51). The estimate of the true test log probability was −82.86 with an error estimate (−82.68, 83.12). The difference of about 0.5 nats shows that the bound is rather tight.

For a simple comparison, we also trained several mixture of Bernoullis models with 10, 100, 500, 1000, and 2000 components. The corresponding average test log probabilities were −168.95, −142.63, −137.64, −133.21, and

[10]Note that computationally, this is equivalent to estimating 100 partition functions, as discussed at the end of section 4.1.

Table 2: Classification Error Rates on MNIST Test Set When Only a Small Fraction of Labeled Data Is Available But the Remainder of the 60,000 Training Cases Is Available Unlabeled.

	Two-Layer DBM	Nonlinear NCA	Linear NCA	Stack of Autoencoders	KNN
1% (600)	4.82%	8.81%	19.37%	8.13%	13.74%
5% (3000)	2.72%	3.24%	7.23%	3.54%	7.19%
10% (6000)	2.46%	2.58%	4.89%	2.98%	5.87%
100% (60000)	0.95%	1.00%	2.45%	1.40%	3.09%

−135.78. Compared to DBMs, a mixture of Bernoullis performs very badly. The difference of about 50 nats per test case is striking.

Finally, after discriminative fine-tuning, the two-hidden-layer Boltzmann machine achieves an error rate of 0.95% on the full MNIST test set. This is, to our knowledge, the best published result on the permutation-invariant version of the MNIST task.[11] The three-layer BM gives a slightly worse error rate of 1.01%. The flat BM, on the other hand, gives considerably worse error rate of 1.27%. This is compared to 1.4% achieved by SVMs (Decoste & Schölkopf, 2002), 1.6% achieved by randomly initialized backpropagation, 1.2% achieved by the deep belief network, described in Hinton et al. (2006) and Hinton and Salakhutdinov (2006), and 0.97% obtained by using a combination of discriminative and generative fine-tuning on the same DBN (Hinton, 2007).

To test discriminative performance of DBMs when the number of labeled examples is small, we randomly sampled 1%, 5%, and 10% of the handwritten digits in each class and treated them as labeled data. Table 2 shows that after discriminative fine-tuning, a two-hidden-layer BM achieves error rates of 4.82%, 2.72%, and 2.46%. Deep Boltzmann machines clearly outperform regularized nonlinear NCA (Salakhutdinov & Hinton, 2007), linear NCA (Goldberger, Roweis, Hinton, & Salakhutdinov, 2004), a stack of greedily pretrained autoencoders (Bengio, Lamblin, Popovici, & Larochelle, 2007), and K-nearest neighbors, particularly when the number of labeled examples is only 600. We note that similar to DBMs, both regularized nonlinear NCA and a stack of autoencoders use an unsupervised pretraining stage (trained on all 60,000 unlabeled MNIST images), followed by supervised fine-tuning.

5.2 NORB. Results on MNIST show that deep Boltzmann machines can significantly outperform many other models on the well-studied but

[11]In the permutation-invariant version, the pixels of every image are subjected to the same random permutation, which makes it hard to use prior knowledge about images.

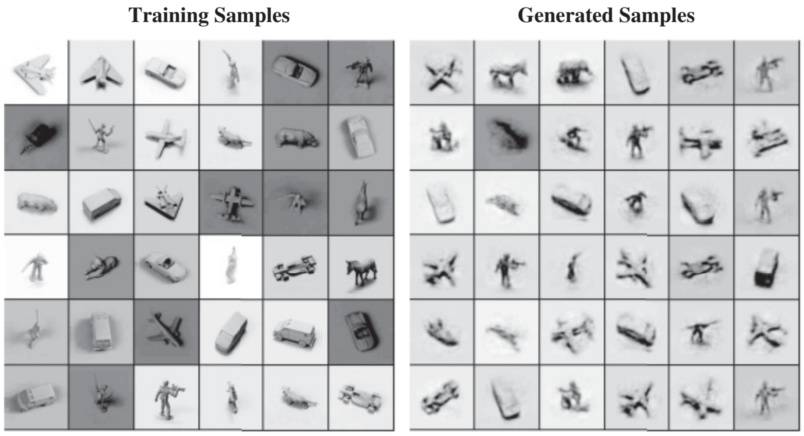

Figure 9: Random samples from the training set, and samples generated from a three-hidden-layer deep Boltzmann machine by running the Gibbs sampler for 10,000 steps.

relatively simple task of handwritten digit recognition. In this section, we present results on NORB, which is a considerably more difficult data set than MNIST. NORB (LeCun, Huang, & Bottou, 2004) contains images of 50 different 3D toy objects with 10 objects in each of five generic classes: cars, trucks, planes, animals, and humans. Each object is photographed from different viewpoints and under various lighting conditions. The training set contains 24,300 stereo image pairs of 25 objects, 5 per class, while the test set contains 24,300 stereo pairs of the remaining, different 25 objects. The goal is to classify each previously unseen object into its generic class. From the training data, 4,300 were set aside for validation.

Each image has 96×96 pixels with integer grayscale values in the range [0,255]. To speed up experiments, we reduced the dimensionality by using a foveal representation of each image in a stereo pair. The central 64×64 portion of an image is kept at its original resolution. The remaining 16 pixel-wide-ring around it is compressed by replacing nonoverlapping square blocks of pixels in the ring with a single scalar given by the average pixel value of a block. We split the ring into four smaller ones: the outermost ring consists of 8×8 blocks, followed by a ring of 4×4 blocks, and finally two innermost rings of 2×2 blocks. The resulting dimensionality of each training vector, representing a stereo pair, was $2 \times 4488 = 8976$. A random sample from the training data used in our experiments is shown in Figure 9 (left).

To model raw pixel data, we use an RBM with gaussian visible and binary hidden units. Gaussian RBMs have been previously successfully applied for modeling greyscale images, such as images of faces (Hinton &

Salakhutdinov, 2006). However, learning an RBM with gaussian units can be slow, particularly when the input dimensionality is quite large. Here we follow the approach of Nair and Hinton (2009) by first learning a gaussian RBM and then treating the activities of its hidden layer as "preprocessed" data. Effectively, the learned low-level RBM acts as a preprocessor that converts grayscale pixels into a binary representation, which we then use for learning a deep Boltzmann machine.

The number of hidden units for the preprocessing RBM was set to 4000, and the model was trained using contrastive divergence learning for 500 epochs. We then trained a two-hidden-layer DBM with each layer containing 4000 hidden units, as shown in Figure 8 (right). Note that the entire model was trained in a completely unsupervised way. After the subsequent discriminative fine-tuning, the "unrolled" DBM achieves a misclassification error rate of 10.8% on the full test set. This is compared to 11.6% achieved by SVMs (Bengio & LeCun, 2007), 22.5% achieved by logistic regression, and 18.4% achieved by the K-nearest neighbors (LeCun et al., 2004). To show that DBMs can benefit from additional unlabeled training data, we augmented the training data with additional unlabeled data by applying simple pixel translations, creating 1,166,400 training instances.[12] After learning a good generative model, the discriminative fine-tuning (using only the 24,300 labeled training examples without any translation) reduces the misclassification error to 7.2%. Figure 9 shows samples generated from the model by running prolonged Gibbs sampling. Note that the model was able to capture a lot of regularities in this high-dimensional, richly structured data, including different object classes, various viewpoints, and lighting conditions.

Finally, we tested the ability of the DBM to perform an image in-painting task. To this end, we randomly selected 10 objects from the test set and simulated the occlusion by zeroing out the left half of the image (see Figure 10). We emphasize that the test objects are different from the training objects (i.e., the model never sees images of "cowboy," but it sees other images belonging to the "person" category). We next sampled the "missing" pixels conditioned on the nonoccluded pixels of the image using 1000 Gibbs updates. Figure 10 (bottom) shows that the model was able to coherently infer occluded parts of the test images. In particular, observe that even though the model never sees an image of the cowboy, it correctly infers that it should have two legs and two arms.

Surprisingly, even though the deep Boltzmann machine contains about 68 million parameters, it significantly outperforms many of the competing models. Clearly, unsupervised learning helps generalization because it ensures that most of the information in the model parameters comes from modeling the input data. The very limited information in the labels is used

[12]We thank Vinod Nair for sharing his code for blurring and translating NORB images.

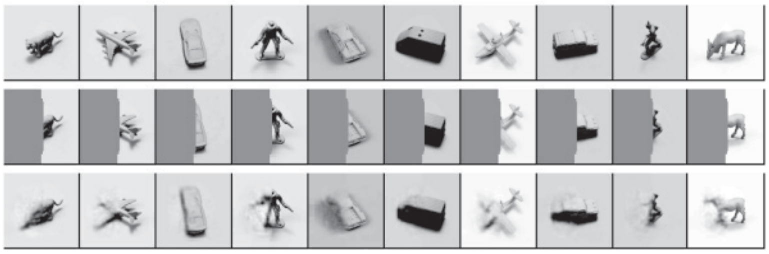

Figure 10: Performance of the three-hidden-layer DBM on the image inpainting task. (Top) Ten objects randomly sampled from the test set. (Middle) Partially occluded input images. (Bottom) Inferred images were generated by running a Gibbs sampler for 1000 steps.

only to slightly adjust the layers of features already discovered by the deep Boltzmann machine.

6 Discussion

A major difference between DBNs and DBMs is that the procedure for adding an extra layer to a DBN replaces the whole prior over the previous top layer, whereas the procedure for adding an extra layer to a DBM replaces only half of the prior. So in a DBM, the weights of the bottom-level RBM end up doing much more of the work than in a DBN where the weights are used only to define $p(\mathbf{v}|\mathbf{h}^{(1)}; W^1)$ (in the composite generative model). This suggests that adding layers to a DBM will give diminishing improvements in the variational bound much more quickly than adding layers to a DBN. There is, however, a simple way to pretrain a DBM so that more of the modeling work is left to the higher layers.

Suppose we train an RBM with one set of hidden units, and four sets of visible units, and we constrain the four weight matrices (and visible biases) to be identical.[13] Then we use the hidden activities as data to train an RBM with one set of visible units and four sets of hidden units, again constrained to have identical weight matrices. Now we can combine the two RBMs into a DBM with two hidden layers by using one copy of the weight matrix from the first RBM and three times one of the copies of the weight matrix from the second RBM. In this DBM, three-fourths of the first RBM's prior over the first hidden layer has been replaced by the prior defined by the second RBM. It remains to be seen whether this makes DBMs work better. It is also not obvious how this idea can be applied to intermediate hidden layers.

[13]As before, we use mean-field reconstructions of the four sets of visible units to avoid modeling the fact that all four sets have the same states in the data.

In this article, we have focused on Boltzmann machines with binary units. The learning methods we have described can be extended to learn deep Boltzmann machines built with RBM modules that contain real-valued (Marks & Movellan, 2001), count (Salakhutdinov & Hinton, 2009b), or tabular data provided the distributions are in the exponential family (Welling, Rosen-Zvi, & Hinton, 2005). However, it often requires additional insights to get the basic RBM learning module to work well with nonbinary units. For example, it ought to be possible to learn the variance of the noise model of the visible units in a Gaussian-Bernoulli RBM, but this is typically very difficult for reasons explained in Hinton (2010). For modeling the NORB data, we used fixed variances of 1, which is clearly much too big for data that have been normalized so that the pixels have a variance of 1. Recent work shows that gaussian visible units work much better with rectified linear hidden units (Nair & Hinton, 2010) and using this type of hidden unit it is straightforward to learn the variance of the noise model of each visible unit.

7 Summary

We presented a novel combination of variational and Markov chain Monte Carlo algorithms for training Boltzmann machines. When applied to pretrained deep Boltzmann machines with several hidden layers and millions of weights, this combination is a very effective way to learn good generative models. We demonstrated the performance of the algorithm using the MNIST handwritten digits and the NORB stereo images of 3D objects with highly variable viewpoint and lighting.

A simple variational approximation works well for estimating the data-dependent statistics because learning based on these estimates encourages the true posterior distributions over the hidden variables to be close to their variational approximations. Persistent Markov chains work well for estimating the data-independent statistics because learning based on these estimates encourages the persistent chains to explore the state space much more rapidly than would be predicted by their mixing rates.

Pretraining a stack of RBMs using contrastive divergence can be used to initialize the weights of a deep Boltzmann machine to sensible values. The RBMs can then be composed to form a deep Boltzmann machine. The pretraining ensures that the variational inference can be initialized sensibly by a single bottom-up pass from the data vector using twice the bottom-up weights to compensate for the lack of top-down input on the initial pass.

We further showed how annealed importance sampling, along with variational inference, can be used to estimate a variational lower bound on the log probability that a deep Boltzmann machine assigns to test data. This allowed us to directly assess the performance of deep Boltzmann machines as generative models of data. Finally, we showed how to use a deep Boltzmann machine to initialize the weights of a feedforward neural network that can then be discriminatively fine-tuned. These networks give excellent discriminative performance, especially when there are very few labeled training data but a large supply of unlabeled data.

Appendix

For clarity of presentation, let p be a shorthand notation for the distribution $P(\mathbf{h}^{(1)}; W^{(1)})$, defined in equation 3.12, and let p_h denote an individual term from this distribution. Similarly, we will let r_h denote an individual term from $P(\mathbf{h}^{(1)}; W^{(2)})$, q_h denote a term from $Q(\mathbf{h}^{(1)}|\mathbf{v}_n; W^{(1)})$, and m_h denote a term from the geometric mean of the two probability distributions $P(\mathbf{h}^{(1)}; W^{(1)})$, $P(\mathbf{h}^{(1)}; W^{(2)})$,

$$m_h = \frac{p_h^{1/2} r_h^{1/2}}{\mathcal{Z}}, \quad \mathcal{Z} = \sum_h p_h^{1/2} r_h^{1/2}, \tag{A.1}$$

which is the renormalized pairwise product of the square roots of the two probabilities for each event (see equation 3.14). We note that the normalizing constant $\mathcal{Z}$ is equal to 1 if the two distributions p and r are identical, and it is less than 1 otherwise.

For each case n, given

$$\mathrm{KL}(q||r) \leq \mathrm{KL}(q||p), \tag{A.2}$$

we want to show that $\mathrm{KL}(q||m) \leq \mathrm{KL}(q||p)$:

$$\begin{aligned}
\mathrm{KL}(q||m) &= \sum_h q_h \log \frac{q_h}{m_h} \\
&= \sum_h q_h \log q_h - \sum_h q_h \left(\frac{1}{2}\log p_h + \frac{1}{2}\log r_h\right) + \log \mathcal{Z} \\
&\leq \sum_h q_h \log q_h - \sum_h q_h \left(\frac{1}{2}\log p_h + \frac{1}{2}\log p_h\right) + \log \mathcal{Z} \\
&\qquad \text{(follows from equation A.2)} \\
&\leq \sum_h q_h \log q_h - \sum_h q_h \log p_h + \log \mathcal{Z} \\
&\leq \sum_h q_h \log q_h - \sum_h q_h \log p_h \quad (\text{since } \mathcal{Z} \leq 1) \\
&\leq \mathrm{KL}(q||p).
\end{aligned}$$

The above derivation is a special case of a more general result, provided in Hinton (2002). It states that the Kullback-Leibler divergence between the geometric mean of a set of probability distributions P_i and the distribution Q is smaller than the average of the Kullback-Leibler divergences of the individual distributions:

$$\mathrm{KL}\left(Q \,||\, \frac{\prod_i P_i^{w_i}}{\mathcal{Z}}\right) \leq \sum_i w_i \mathrm{KL}(Q||P_i), \qquad \mathcal{Z} = \sum_x \prod_i P_i^{w_i}(x), \tag{A.3}$$

where w_i are nonnegative and sum to 1. Note that the normalizing constant $\mathcal{Z}$ is 1 if all of the individual distributions are identical. Otherwise, $\mathcal{Z} < 1$, and the difference between the two sides of the above equation is $\log(1/\mathcal{Z})$.

Acknowledgments

This research was supported by NSERC and by gifts from Google and Microsoft.

References

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, *2*(1), 1–127.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, & T. Hoffman (Eds.), *Advances in neural information processing systems, 11* (pp. 153–160). Cambridge, MA: MIT Press.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, & J. Weston (Eds.), *Large-scale kernel machines*. Cambridge, MA: MIT Press.

Carreira-Perpignan, M. A., & Hinton, G. E. (2005). On contrastive divergence learning. In R. G. Cowell and Z. Ghahramani (Eds.), *Artificial intelligence and statistics*. Society for Artificial Intelligence and Statistics.

Dahl, G. E., Ranzato, M. A., Mohamed, A., & Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted boltzmann machine. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, & A. Culotta (Eds.), *Advances in neural information processing systems, 23* (pp. 469–477). Red Hook, NY: Curran Associates.

Decoste, D., & Schölkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, *46*(1/3), 161–190.

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., & Delalleau, O. (2010). Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In *Proceedings of the 13th International Workshop on AI and Statistics* (pp. 145–152). Cambridge, MA: MIT Press.

Galland, C. (1991). *Learning in deterministic Boltzmann machine networks*. Unpublished doctoral dissertation, University of Toronto.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, *6*(6), 721–741.

Goldberger, J., Roweis, S. T., Hinton, G. E., & Salakhutdinov, R. R. (2004). Neighbourhood components analysis. In L. K. Saul, Y. Wiess, & L. Bottou (Eds.), *Advances in neural information processing systems, 17* (pp. 513–520). Cambridge, MA: MIT Press.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1711–1800.

Hinton, G. E. (2007). To recognize shapes, first learn to generate images. In P. Cisek, T. Drew, & J. F. Kalaska (Eds.), *Computational neuroscience: Theoretical insights into brain function*. New York: Elsevier.

Hinton, G. E. (2010). *A practical guide to training restricted Boltzmann machines*. (Tech. Rep. 2010-000). Toronto: Machine Learning Group, University of Toronto.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hinton, G. E., & Sejnowski, T. (1983). Optimal perceptual inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE.

Hinton, G. E., & Zemel, R. S. (1994). Autoencoders, minimum description length and Helmholtz free energy. In J. D. Cowan, G. Tessauro, & J. Alspetor (Eds.), *Advances in neural information processing systems, 6* (pp. 3–10). San Francisco: Morgan Kaufmann.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., & Saul, L. K. (1999). An introduction to variational methods for graphical models. In M. I. Jordan (Ed.), *Learning in graphical models*. Cambridge, MA: MIT Press.

Kappen, H. J., & Rodriguez, F. B. (1998). Boltzmann machine learning using mean field theory and linear response correction. In M. Stearns, S. Solla, & D. Cohn (Eds.), *Advances in neural information processing systems*, *10* (pp. 280–286). Cambridge, MA: MIT Press.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

LeCun, Y., Huang, F. J., & Bottou, L. (2004). Learning methods for generic object recognition with invariance to pose and lighting. In *Proc. Computer Vision and Pattern Recognition* (pp. 97–104). Piscataway, NJ: IEEE.

Marks, T. K., & Movellan, J. R. (2001). Diffusion networks, product of experts, and factor analysis. In *Proc. Int. Conf. on Independent Component Analysis* (pp. 481–485). New York: Springer.

Mohamed, A., Dahl, G. E., & Hinton, G. E. (2012). Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech, and Language Processing* (pp. 14–22). Piscataway, NJ: IEEE.

Murray, I., & Salakhutdinov, R. R. (2009). Evaluating probabilities under high-dimensional latent variable models. In D. Köller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems, 21* (pp. 1137–1144). Cambridge, MA: MIT Press.

Nair, V., & Hinton, G. E. (2009). Implicit mixtures of restricted Boltzmann machines. In D. Koller, D. Schuurmans, Y. Bengio, & L. Bottou (Eds.), *Advances in neural information processing systems, 21* (pp. 1145–1152). Cambridge, MA: IEEE Press.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proc. 27th International Conference on Machine Learning* (pp. 807–814). Madison, WI: Omnipress.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*(1), 71–113.

Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, *11*, 125–139.

Neal, R. M., & Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan (Ed.), *Learning in graphical models* (pp. 355–368). Dordrecht: Kluwer Academic Press.

Osindero, S., & Hinton, G. E. (2008). Modeling image patches with a directed hierarchy of Markov random fields. In J. C. Platt, D. Köller, Y. Singer, & S. Roweis (Eds.), *Advances in neural information processing systems*, *20* (pp. 1121–1128). Cambridge, MA: MIT Press.

Peterson, C., & Anderson, J. R. (1987). A mean field theory learning algorithm for neural networks. *Complex Systems*, *1*, 995–1019.

Ranzato, M. A. (2009). *Unsupervised learning of feature hierarchies*. Unpublished doctoral dissertation, New York University.

Ranzato, M. A., Huang, F., Boureau, Y., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. Piscataway, NJ: IEEE.

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Ann. Math. Stat.*, *22*, 400–407.

Salakhutdinov, R. R. (2009). Learning in Markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems, 23* (pp. 1598–1606). Red Hook, NJ: Curran Associates.

Salakhutdinov, R. R., & Hinton, G. E. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In M. Meila & X. Shen (Eds.), *Proceedings of the International Conference on Artificial Intelligence and Statistics* (Vol. 11, pp. 412–419). Cambridge, MA: MIT Press.

Salakhutdinov, R. R., & Hinton, G. E. (2009a). Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics* (Vol. 12, pp. 448–455). Cambridge, MA: MIT Press.

Salakhutdinov, R. R., & Hinton, G. E. (2009b). Replicated softmax: An undirected topic model. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, & A. Culotta (Eds.), *Advances in neural information processing systems, 22* (pp. 1607–1614). Red Hook, NY: Curran Associates.

Salakhutdinov, R. R., Mnih, A., & Hinton, G. E. (2007). Restricted Boltzmann machines for collaborative filtering. In Z. Ghahramani (Ed.), *Proceedings of the International Conference on Machine Learning* (Vol. 24, pp. 791–798). New York: ACM.

Salakhutdinov, R. R., & Murray, I. (2008). On the quantitative analysis of deep belief networks. In *Proceedings of the International Conference on Machine Learning* (Vol. 25, pp. 872–879). Madison, WI: Omnipress.

Serre, T., Oliva, A., & Poggio, T. A. (2007). A feedforward architecture accounts for rapid categorization. *Proceedings of the National Academy of Sciences*, *104*, 6424–6429.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing, Vol. 1: Foundations* (pp. 194–281). Cambridge, MA: MIT Press.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Machine Learning: Proceedings of the Twenty-First International Conference* (pp. 1064–1071). New York: ACM.

Tieleman, T., & Hinton, G. E. (2009). Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th International Conference on Machine Learning* (pp. 1033–1040). New York: ACM.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. (2008). Extracting and composing robust features with denoising autoencoders. In W. W. Cohen, A. McCallum, & S. T. Roweis (Eds.), *Proceedings of the Twenty-Fifth Annual International Conference on Machine Learning* (Vol. 307, pp. 1096–1103). Madison, WI: Omnipress.

Welling, M. (2009). Herding dynamical weights to learn. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 141–148). New York: ACM.

Welling, M., Rosen-Zvi, M., & Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems, 17* (pp. 1481–1488). Cambridge, MA: MIT Press.

Williams, C., & Agakov, F. (2002). *An analysis of contrastive divergence learning in Gaussian Boltzmann machines*. (Tech. Rep. EDI-INF-RR-0120). Edinburgh: Institute for Adaptive and Neural Computation, University of Edinburgh.

Younes, L. (1989). Parameter inference for imperfectly observed Gibbsian fields. *Probability Theory Rel. Fields*, *82*, 625–645.

Younes, L. (1999). On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastics Reports*, *65*, 177–288.

Yuille, A. L. (2004). The convergence of contrastive divergences. In L. K. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*, *17* (pp. 1593–1600). Cambridge, MA: MIT Press.

Zemel, R. S. (1993). *A minimum description length framework for unsupervised learning*. Unpublished doctoral dissertation, University of Toronto.

Received November 26, 2011; accepted January 20, 2012.