Deep Boltzmann Machines

Ruslan Salakhutdinov
Department of Computer Science
University of Toronto
rsalakhu@cs.toronto.edu

Geoffrey Hinton
Department of Computer Science
University of Toronto
hinton@cs.toronto.edu

Abstract

We present a new learning algorithm for Boltzmann machines that contain many layers of hidden variables. Data-dependent expectations are estimated using a variational approximation that tends to focus on a single mode, and data-independent expectations are approximated using persistent Markov chains. The use of two quite different techniques for estimating the two types of expectation 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 "pre-training" phase that allows variational inference to be initialized with a single bottom-up pass. We present results on the MNIST and NORB datasets showing that deep Boltzmann machines learn good generative models and perform well on handwritten digit and visual object recognition tasks.

1 Introduction

The original learning algorithm for Boltzmann machines (Hinton and Sejnowski, 1983) required randomly initialized Markov chains to approach their equilibrium distributions in order to estimate the data-dependent and data-independent expectations that a connected pair of binary variables would both be on. The difference of these two expectations is the gradient required for maximum likelihood learning. Even with the help of simulated annealing, this learning procedure was too slow to be practical. Learning can be made much more efficient in a restricted Boltzmann machine (RBM), which has no connections between hidden units (Hinton, 2002). Multiple hidden layers can be learned by treating the hidden activities of one RBM as the data for training a higher-level RBM (Hinton et al., 2006; Hinton and 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 its lower layers.

In this paper we present a much more efficient learning procedure for fully general Boltzmann machines. We also show that if the connections between hidden units are restricted in such a way that the hidden units form multiple layers, it is possible to use a stack of slightly modified RBM's to initialize the weights of a deep Boltzmann machine before applying our new learning procedure.

2 Boltzmann Machines (BM's)

A Boltzmann machine is a network of symmetrically coupled stochastic binary units. It contains a set of visible units $\mathbf{v} \in \{0,1\}^D$, and a set of hidden units $\mathbf{h} \in \{0,1\}^P$ (see Fig. 1). The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is defined as:

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

where $\theta = \{\mathbf{W}, \mathbf{L}, \mathbf{J}\}$ are the model parameters[1]: $\mathbf{W}, \mathbf{L}, \mathbf{J}$ represent visible-to-hidden, visible-to-visible, and hidden-to-hidden symmetric interaction terms. The diagonal elements of $\mathbf{L}$ and $\mathbf{J}$ are set to 0. The probability that the model assigns to a visible vector $\mathbf{v}$ is:

$$p(\mathbf{v}; \theta) = \frac{p^*(\mathbf{v}; \theta)}{Z(\theta)} = \frac{1}{Z(\theta)} \sum_h \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \quad (2)$$

$$Z(\theta) = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\left(-E(\mathbf{v}, \mathbf{h}; \theta)\right), \quad (3)$$

where p^* denotes unnormalized probability, and $Z(\theta)$ is the partition function. The *conditional* distributions over

Appearing in Proceedings of the 12^{th} International Confe-rence on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

[1] We have omitted the bias terms for clarity of presentation

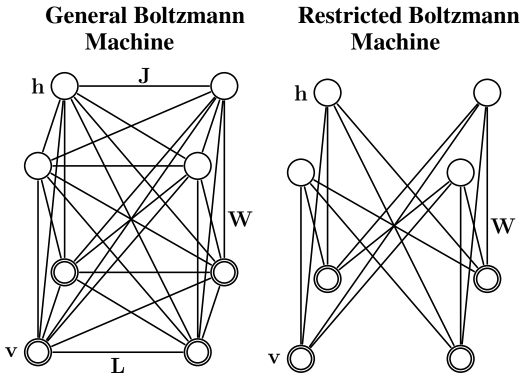

Figure 1: **Left**: A general Boltzmann machine. The top layer represents a vector of stochastic binary "hidden" features and the bottom layer represents a vector of stochastic binary "visible" variables. **Right**: A restricted Boltzmann machine with no hidden-to-hidden and no visible-to-visible connections.

hidden and visible units are given by:

$$p(h_j = 1|\mathbf{v}, \mathbf{h}_{-j}) = \sigma\big(\sum_{i=1}^{D} W_{ij} v_i + \sum_{m=1\backslash j}^{P} J_{jm} h_j\big), \quad (4)$$

$$p(v_i = 1|\mathbf{h}, \mathbf{v}_{-i}) = \sigma\big(\sum_{j=1}^{P} W_{ij} h_j + \sum_{k=1\backslash i}^{D} L_{ik} v_j\big), \quad (5)$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic function. 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 Eq. 2:

$$\begin{aligned} \Delta\mathbf{W} &= \alpha\left(\mathrm{E}_{P_{\mathrm{data}}}[\mathbf{v}\mathbf{h}^\top] - \mathrm{E}_{P_{\mathrm{model}}}[\mathbf{v}\mathbf{h}^\top]\right), \quad (6) \\ \Delta\mathbf{L} &= \alpha\left(\mathrm{E}_{P_{\mathrm{data}}}[\mathbf{v}\mathbf{v}^\top] - \mathrm{E}_{P_{\mathrm{model}}}[\mathbf{v}\mathbf{v}^\top]\right), \\ \Delta\mathbf{J} &= \alpha\left(\mathrm{E}_{P_{\mathrm{data}}}[\mathbf{h}\mathbf{h}^\top] - \mathrm{E}_{P_{\mathrm{model}}}[\mathbf{h}\mathbf{h}^\top]\right), \end{aligned}$$

where α is a learning rate, $\mathrm{E}_{P_{\mathrm{data}}}[\cdot]$ denotes an expectation with respect to the completed data distribution $P_{\mathrm{data}}(\mathbf{h}, \mathbf{v}; \theta) = p(\mathbf{h}|\mathbf{v}; \theta) P_{\mathrm{data}}(\mathbf{v})$, with $P_{\mathrm{data}}(\mathbf{v}) = \frac{1}{N}\sum_n \delta(\mathbf{v} - \mathbf{v}_n)$ representing the empirical distribution, and $\mathrm{E}_{P_{\mathrm{model}}}[\cdot]$ is an expectation with respect to the distribution defined by the model (see Eq. 2). We will sometimes refer to $\mathrm{E}_{P_{\mathrm{data}}}[\cdot]$ as the *data-dependent expectation*, and $\mathrm{E}_{P_{\mathrm{model}}}[\cdot]$ as the *model's expectation*.

Exact maximum likelihood learning in this model is intractable because exact computation of both the data-dependent expectations and the model's expectations takes a time that is exponential in the number of hidden units. Hinton and Sejnowski (1983) proposed an algorithm that uses Gibbs sampling to approximate both expectations. For each iteration of learning, a separate Markov chain is run for every training data vector to approximate $\mathrm{E}_{P_{\mathrm{data}}}[\cdot]$, and an additional chain is run to approximate $\mathrm{E}_{P_{\mathrm{model}}}[\cdot]$. The main problem with this learning algorithm is the time required to approach the stationary distribution, especially when estimating the model's expectations, since the Gibbs chain may need to explore a highly multimodal energy landscape. This is typical when modeling real-world distributions such as datasets of images in which almost all of the possible images have extremely low probability, but there are many very different images that occur with quite similar probabilities.

Setting both **J**=0 and **L**=0 recovers the well-known restricted Boltzmann machine (RBM) model (Smolensky, 1986) (see Fig. 1, right panel). In contrast to general BM's, inference in RBM's is exact. Although exact maximum likelihood learning in RBM's is still intractable, learning can be carried out efficiently using Contrastive Divergence (CD) (Hinton, 2002). It was further observed (Welling and Hinton, 2002; Hinton, 2002) that for Contrastive Divergence to perform well, it is important to obtain exact samples from the conditional distribution $p(\mathbf{h}|\mathbf{v}; \theta)$, which is intractable when learning full Boltzmann machines.

2.1 Using Persistent Markov Chains to Estimate the Model's Expectations

Instead of using CD learning, it is possible to make use of a stochastic approximation procedure (SAP) to approximate the model's expectations (Tieleman, 2008; Neal, 1992). SAP belongs to the class of well-studied stochastic approximation algorithms of the Robbins–Monro type (Robbins and Monro, 1951; Younes, 1989, 2000). The idea behind these methods is straightforward. Let θ_t and X^t be the current parameters and the state. Then X^t and θ_t are updated sequentially as follows:

- Given X^t, a new state X^{t+1} is sampled from a transition operator $T_{\theta_t}(X^{t+1}; X^t)$ that leaves p_{θ_t} invariant.
- A new parameter θ_{t+1} is then obtained by replacing the intractable model's expectation by the expectation with respect to X^{t+1}.

Precise sufficient conditions that guarantee almost sure convergence to an asymptotically stable point are given in (Younes, 1989, 2000; Yuille, 2004). One necessary condition requires the learning rate to decrease with time, i.e. $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$. This condition can be trivially satisfied by setting $\alpha_t = 1/t$. 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.

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 only run for a few MCMC updates per parameter update. Samples from the persistent chain will be highly correlated for successive parameter updates, but again, 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. Many persistent chains can be run in parallel and we will refer to the current state in each of these chains as a "fantasy" particle.

2.2 A Variational Approach to Estimating the Data-Dependent Expectations

In variational learning (Hinton and Zemel, 1994; Neal and Hinton, 1998), 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 follow the gradient of a lower bound on the log-likelihood:

$$\begin{aligned} \ln p(\mathbf{v};\theta) &\geq \sum_{\mathbf{h}} q(\mathbf{h}|\mathbf{v};\mu) \ln p(\mathbf{v},\mathbf{h};\theta) + \mathcal{H}(q) \qquad (7) \\ &= \ln p(\mathbf{v};\theta) - KL[q(\mathbf{h}|\mathbf{v};\mu)||p(\mathbf{h}|\mathbf{v};\theta)], \end{aligned}$$

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 divergences between the approximating and true posteriors. Using a naive mean-field approach, we choose a fully factorized distribution in order to approximate the true posterior: $q(\mathbf{h};\mu) = \prod_{j=1}^{P} q(h_i)$, with $q(h_i = 1) = \mu_i$ where P is the number of hidden units. The lower bound on the log-probability of the data takes the form:

$$\begin{aligned} \ln p(\mathbf{v};\theta) &\geq \frac{1}{2}\sum_{i,k} L_{ik} v_i v_k + \frac{1}{2}\sum_{j,m} J_{jm}\mu_j\mu_m \\ &+ \sum_{i,j} W_{ij} v_i \mu_j - \ln Z(\theta) \\ &+ \sum_{j} \left[\mu_j \ln \mu_j + (1-\mu_j)\ln(1-\mu_j)\right]. \end{aligned}$$

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

$$\mu_j \leftarrow \sigma\Big(\sum_i W_{ij} v_i + \sum_{m\backslash j} J_{mj}\mu_m\Big). \qquad (8)$$

This is followed by applying SAP to update the model parameters θ (Salakhutdinov, 2008). We emphasize that variational approximations cannot be used for approximating the expectations with respect to the model distribution in the Boltzmann machine learning rule because the minus sign (see Eq. 6) would cause variational learning to change the parameters so as to *maximize* the divergence between the approximating and true distributions. If, however, a persistent chain is used to estimate the model's expectations, variational learning can be applied for estimating the data-dependent expectations.

The choice of naive mean-field was deliberate. First, the convergence is usually very fast, which greatly facilitates learning. Second, for applications such as the interpretation of images or speech, we expect the posterior over hidden states *given the data* to have a single mode, so simple and fast variational approximations such as mean-field should be adequate. Indeed, sacrificing some log-likelihood in order to make the true posterior unimodal could be advantageous for a system that must use the posterior to control its actions. Having many quite different and equally good representations of the same sensory input increases log-likelihood but makes it far more difficult to associate an appropriate action with that sensory input.

Boltzmann Machine Learning Procedure:

Given: a training set of N data vectors $\{\mathbf{v}\}_{n=1}^{N}$.

1. Randomly initialize parameters θ^0 and M fantasy particles. $\{\tilde{\mathbf{v}}^{0,1}, \tilde{\mathbf{h}}^{0,1}\}, ..., \{\tilde{\mathbf{v}}^{0,M}, \tilde{\mathbf{h}}^{0,M}\}$
2. For t=0 to T (# of iterations)
   (a) For each training example $\mathbf{v}^n$, n=1 to N
      - Randomly initialize μ and run mean-field updates Eq. 8 until convergence.
      - Set $\mu^n = \mu$.

   (b) For each fantasy particle m=1 to M
      - Obtain a new state $(\tilde{\mathbf{v}}^{t+1,m}, \tilde{\mathbf{h}}^{t+1,m})$ by running a k-step Gibbs sampler using Eqs. 4, 5, initialized at the previous sample $(\tilde{\mathbf{v}}^{t,m}, \tilde{\mathbf{h}}^{t,m})$.

   (c) Update
   $$W^{t+1} = W^t + \alpha_t\Big(\frac{1}{N}\sum_{n=1}^{N}\mathbf{v}^n(\mu^n)^\top - \frac{1}{M}\sum_{m=1}^{M}\tilde{\mathbf{v}}^{t+1,m}(\tilde{\mathbf{h}}^{t+1,m})^\top\Big).$$
   Similarly update parameters L and J.

   (d) Decrease α_t.

3 Deep Boltzmann Machines (DBM's)

In general, we will rarely be interested in learning a complex, fully connected Boltzmann machine. Instead, consider learning a deep multilayer Boltzmann machine as shown in Fig. 2, left panel, in which each layer captures complicated, higher-order correlations between the activities of hidden features in the layer below. Deep Boltzmann machines are interesting for several reasons. First, like deep belief networks, DBM's have the potential of learning internal representations that become increasingly complex, which is considered to be a promising way of solving object and speech recognition problems. Second, high-level representations can be built from a large supply of unlabeled sensory inputs and very limited labeled data can then be used to only slightly fine-tune the model for a specific task at hand. Finally, unlike deep belief networks, the approximate inference procedure, in addition to an initial bottom-up pass, can incorporate top-down feedback, allowing deep Boltzmann machines to better propagate uncertainty about, and hence deal more robustly with, ambiguous inputs.

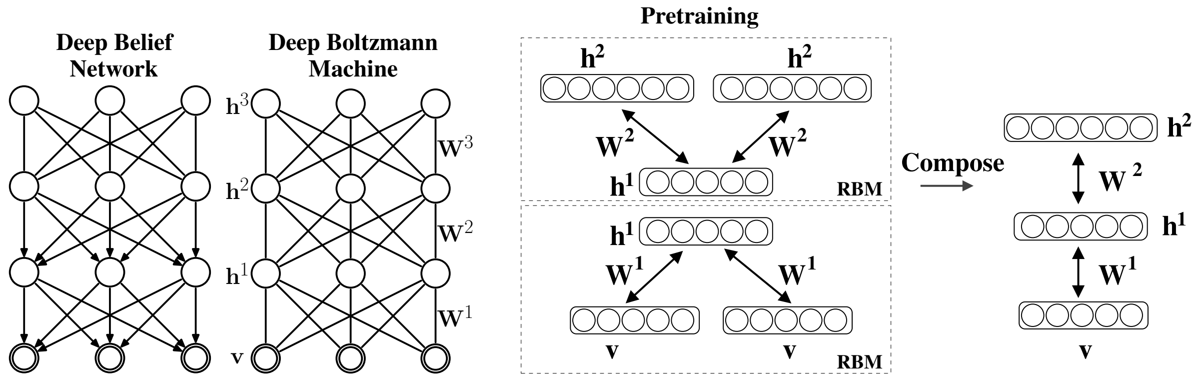

Figure 2: **Left**: A three-layer Deep Belief Network and a three-layer Deep Boltzmann Machine. **Right**: Pretraining consists of learning a stack of modified RBM's, that are then composed to create a deep Boltzmann machine.

Consider a two-layer Boltzmann machine (see Fig. 2, right panel) with no within-layer connections. The energy of the state $\{\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2\}$ is defined as:

$$E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \theta) = -\mathbf{v}^\top \mathbf{W}^1 \mathbf{h}^1 - \mathbf{h}^{1\top} \mathbf{W}^2 \mathbf{h}^2, \qquad (9)$$

where $\theta = \{\mathbf{W}^1, \mathbf{W}^2\}$ 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:

$$p(\mathbf{v}; \theta) = \frac{1}{Z(\theta)} \sum_{\mathbf{h}^1, \mathbf{h}^2} \exp\left(-E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \theta)\right). \qquad (10)$$

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

$$p(h^1_j = 1 | \mathbf{v}, \mathbf{h}^2) = \sigma\big(\sum_i W^1_{ij} v_i + \sum_m W^2_{jm} h^2_j\big), \quad (11)$$

$$p(h^2_m = 1 | \mathbf{h}^1) = \sigma\big(\sum_j W^2_{im} h^1_i\big), \qquad (12)$$

$$p(v_i = 1 | \mathbf{h}^1) = \sigma\big(\sum_j W^1_{ij} h_j\big). \qquad (13)$$

For approximate maximum likelihood learning, we could still apply the learning procedure for general Boltzmann machines described above, but it would be rather slow, particularly when the hidden units form layers which become increasingly remote from the visible units. There is, however, a fast way to initialize the model parameters to sensible values as we describe in the next section.

3.1 Greedy Layerwise Pretraining of DBM's

Hinton et al. (2006) introduced a greedy, layer-by-layer unsupervised learning algorithm that consists of learning a stack of RBM's one layer at a time. After the stack of RBM's has been learned, the whole stack can be viewed as a single probabilistic model, called a "deep belief network". Surprisingly, this model is *not* a deep Boltzmann machine. The top two layers form a restricted Boltzmann machine which is an undirected graphical model, but the lower layers form a *directed* generative model (see Fig. 2).

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

$$p(\mathbf{v}; \theta) = \sum_{\mathbf{h}^1} p(\mathbf{h}^1; \mathbf{W}^1) p(\mathbf{v} | \mathbf{h}^1; \mathbf{W}^1), \qquad (14)$$

where $p(\mathbf{h}^1; \mathbf{W}^1) = \sum_{\mathbf{v}} p(\mathbf{h}^1, \mathbf{v}; \mathbf{W}^1)$ is an implicit prior over $\mathbf{h}^1$ defined by the parameters. The second RBM in the stack replaces $p(\mathbf{h}^1; \mathbf{W}^1)$ by $p(\mathbf{h}^1; \mathbf{W}^2) = \sum_{\mathbf{h}^2} p(\mathbf{h}^1, \mathbf{h}^2; \mathbf{W}^2)$. If the second RBM is initialized correctly (Hinton et al., 2006), $p(\mathbf{h}^1; \mathbf{W}^2)$ will become a better model of the aggregated posterior distribution over $\mathbf{h}^1$, where the aggregated posterior is simply the non-factorial mixture of the factorial posteriors for all the training cases, i.e. $1/N \sum_n p(\mathbf{h}^1 | \mathbf{v}_n; \mathbf{W}^1)$. Since the second RBM is replacing $p(\mathbf{h}^1; \mathbf{W}^1)$ by a better model, it would be possible to infer $p(\mathbf{h}^1; \mathbf{W}^1, \mathbf{W}^2)$ by averaging the two models of $\mathbf{h}^1$ which can be done approximately by using $1/2\mathbf{W}^1$ bottom-up and $1/2\mathbf{W}^2$ top-down. Using $\mathbf{W}^1$ bottom-up and $\mathbf{W}^2$ top-down would amount to double-counting the evidence since $\mathbf{h}^2$ is dependent on $\mathbf{v}$.

To initialize model parameters of a DBM, we propose greedy, layer-by-layer pretraining by learning a stack of RBM's, but with a small change that is introduced to eliminate the double-counting problem when top-down and bottom-up influences are subsequently combined. For the lower-level RBM, we double the input and tie the visible-to-hidden weights, as shown in Fig. 2, right panel. In this modified RBM with tied parameters, the conditional distributions over the hidden and visible states are defined as:

$$p(h^1_j = 1 | \mathbf{v}) = \sigma\big(\sum_i W^1_{ij} v_i + \sum_i W^1_{ij} v_i\big), \quad (15)$$

$$p(v_i = 1 | \mathbf{h}^1) = \sigma\big(\sum_j W^1_{ij} h_j\big). \qquad (16)$$

Contrastive divergence learning works well and the modified RBM is good at reconstructing its training data. Conversely, for the top-level RBM we double the number of hidden units. The conditional distributions for this model take the form:

$$p(h_j^1 = 1|\mathbf{h}^2) = \sigma\big(\sum_m W_{jm}^2 h_m^2 + \sum_m W_{jm}^2 h_m^2\big) \quad (17)$$

$$p(h_m^2 = 1|\mathbf{h}^1) = \sigma\big(\sum_j W_{jm}^2 h_j^1\big). \quad (18)$$

When these two modules are composed to form a single system, the total input coming into the first hidden layer is halved which leads to the following conditional distribution over $\mathbf{h}^1$:

$$p(h_j^1 = 1|\mathbf{v}, \mathbf{h}^2) = \sigma\big(\sum_i W_{ij}^1 v_i + \sum_m W_{jm}^2 h_m^2\big). \quad (19)$$

The conditional distributions over $\mathbf{v}$ and $\mathbf{h}^2$ remain the same as defined by Eqs. 16, 18.

Observe that the conditional distributions defined by the composed model are exactly the same conditional distributions defined by the DBM (Eqs. 11, 12, 13). Therefore greedily pretraining the two modified RBM's leads to an undirected model with symmetric weights – a deep Boltzmann machine. When greedily training a stack of more than two RBM's, the modification only needs to be used for the first and the last RBM's in the stack. For all the intermediate RBM's we simply halve their weights in both directions when composing them to form a deep Boltzmann machine.

Greedily pretraining the weights of a DBM in this way serves two purposes. First, as we show in the experimental results section, it initializes the weights to sensible values. Second, it ensures that there is a very fast way of performing approximate inference by a single upward pass through the stack of RBM's. Given a data vector on the visible units, each layer of hidden units can be activated in a single bottom-up pass by doubling the bottom-up input to compensate for the lack of top-down feedback (except for the very top layer which does not have a top-down input). This fast approximate inference is used to initialize the mean-field method, which then converges much faster than with random initialization.

3.2 Evaluating DBM's

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 deep Boltzmann machines. Together with variational inference this will allow us obtain good estimates of the lower bound on the log-probability of the *test* data.

Suppose we have two distributions defined on some space $\mathcal{X}$ with probability density functions: $p_A(\mathbf{x}) = p_A^*(\mathbf{x})/Z_A$ and $p_B(\mathbf{x}) = p_B^*(\mathbf{x})/Z_B$. Typically $p_A(\mathbf{x})$ is defined to be some simple distribution with known Z_A and from which we can easily draw *i.i.d.* samples. AIS estimates the ratio Z_B/Z_A by defining a sequence of intermediate probability distributions: $p_0, ..., p_K$, with $p_0 = p_A$ and $p_K = p_B$. 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 $T_k(\mathbf{x}'; \mathbf{x})$ that leaves $p_k(\mathbf{x})$ invariant.

Using the special layer-by-layer structure of deep Boltzmann machines, we can derive a more efficient AIS scheme for estimating the model's partition function. Let us again consider a two-layer Boltzmann machine defined by Eq. 10. By explicitly summing out the visible units $\mathbf{v}$ and the 2$^{\text{nd}}$-layer hidden units $\mathbf{h}^2$, we can easily evaluate an unnormalized probability $p^*(\mathbf{h}^1; \theta)$. We can therefore run AIS on a much smaller state space $\mathbf{x} = \{\mathbf{h}^1\}$ with $\mathbf{v}$ and $\mathbf{h}^2$ analytically summed out. The sequence of intermediate distributions, parameterized by β, is defined as follows:

$$p_k(\mathbf{h}^1) = \sum_{\mathbf{v},\mathbf{h}^2} p(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2) =$$
$$= \frac{1}{Z_k} \prod_i \big(1 + e^{(\beta_k \sum_j h_j^1 W_{ij}^1)}\big) \prod_k \big(1 + e^{(\beta_k \sum_j h_j^1 W_{jk}^2)}\big).$$

This approach closely resembles simulated annealing. We gradually change β_k (or inverse temperature) from 0 to 1, annealing from a simple "uniform" model to the final complex model. Using Eqs. 11, 12, 13, it is straightforward to derive an efficient block Gibbs transition operator that leaves $p_k(\mathbf{h}^1)$ invariant.

Once we obtain an estimate of the global partition function $\hat{Z}$, we can estimate, for a given test case $\mathbf{v}^*$, the variational lower bound of Eq. 7:

$$\ln p(\mathbf{v}^*; \theta) \geq -\sum_{\mathbf{h}} q(\mathbf{h}; \mu) E(\mathbf{v}^*, \mathbf{h}; \theta) + \mathcal{H}(q) - \ln Z(\theta)$$
$$\approx -\sum_{\mathbf{h}} q(\mathbf{h}; \mu) E(\mathbf{v}^*, \mathbf{h}; \theta) + \mathcal{H}(q) - \ln \hat{Z},$$

where we defined $\mathbf{h} = \{\mathbf{h}^1, \mathbf{h}^2\}$. For each test vector, 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 $\mathbf{h}^2$, 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 $\sum_{\mathbf{h}^1,\mathbf{h}^2} p^*(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2)$, and together with an estimate of the global partition function we can actually 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.

When learning a deep Boltzmann machine with more than two 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.

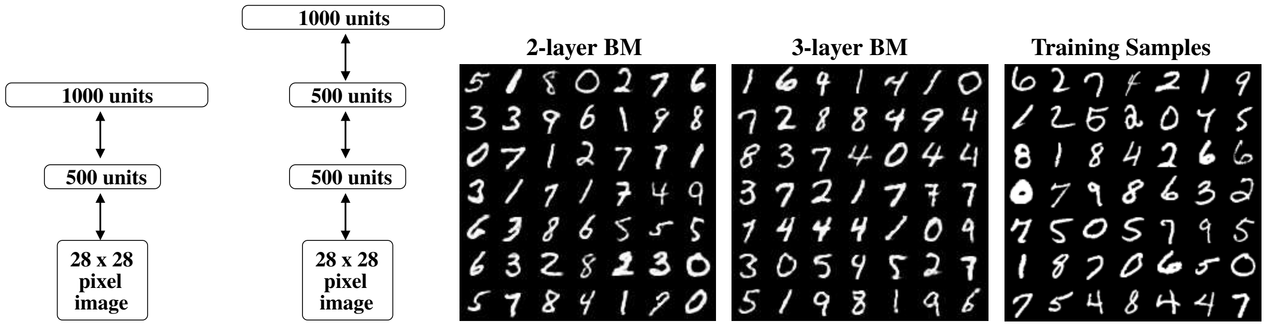

Figure 4: **Left:** Two deep Boltzmann machines used in experiments. **Right:** Random samples from the training set, and samples generated from the two deep Boltzmann machines by running the Gibbs sampler for 100,000 steps. The images shown are the *probabilities* of the binary visible units given the binary states of the hidden units.

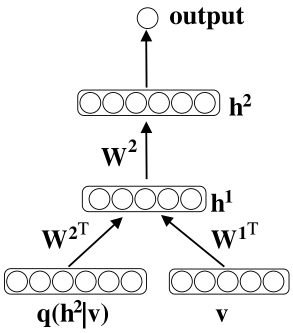

Figure 3: After learning, DBM is used to initialize a multilayer neural network. The marginals of approximate posterior $q(h_j^2 = 1|\mathbf{v})$ are used as additional inputs. The network is fine-tuned by backpropagation.

3.3 Discriminative Fine-tuning of DBM's

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 can be used to initialize a deterministic 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 $q(\mathbf{h}|\mathbf{v})$. 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 Fig. 3. Standard backpropagation can then be used to discriminatively fine-tune the model.

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 $q(\mathbf{h}^2|\mathbf{v})$, i.e. drive the first-layer connections $\mathbf{W}^2$ to zero, which will result in a standard neural network net. Conversely, the network may choose to ignore the input by driving the first-layer $\mathbf{W}^1$ to zero. In all of our experiments, however, the network uses the entire augmented input for making predictions.

4 Experimental Results

In our experiments we used the MNIST and NORB datasets. To speed-up learning, we subdivided datasets into mini-batches, each containing 100 cases, and updated the weights after each mini-batch. The number of fantasy particles used for tracking the model's statistics was also set to 100[2]. For the stochastic approximation algorithm, we always used 5 Gibbs updates of the fantasy particles. The initial learning rate was set 0.005 and was gradually decreased to 0. For discriminative fine-tuning of DBM's we used the method of conjugate gradients on larger mini-batches of 5000 with three line searches performed for each mini-batch in each epoch.

4.1 MNIST

The MNIST digit dataset contains 60,000 training and 10,000 test images of ten handwritten digits (0 to 9), with 28×28 pixels. In our first 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 Fig. 4. To estimate the model's partition function we used 20,000 β_k spaced uniformly from 0 to 1.0. Table 1 shows that the estimates of the lower bound on the average test log-probability were -84.62 and -85.18 for the 2- and 3-layer BM's respectively. This result is slightly better compared to the lower bound of -85.97, achieved by a two-layer deep belief network (Salakhutdinov and Murray, 2008).

Observe that the two DBM's, that contain over 0.9 and 1.15 million parameters, do not appear to suffer much from overfitting. The difference between the estimates of the training and test log-probabilities was about 1 nat. Fig. 4 shows samples generated from the two DBM's 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 note that without greedy pretraining, we could not successfully learn good DBM models of MNIST digits.

[2]It may seem that 100 particles is not nearly enough to represent the model's distribution which may be highly multimodal. However, experience has shown that the fantasy particles move around rapidly because the learning algorithm increases the energy at the location of each fantasy particle.

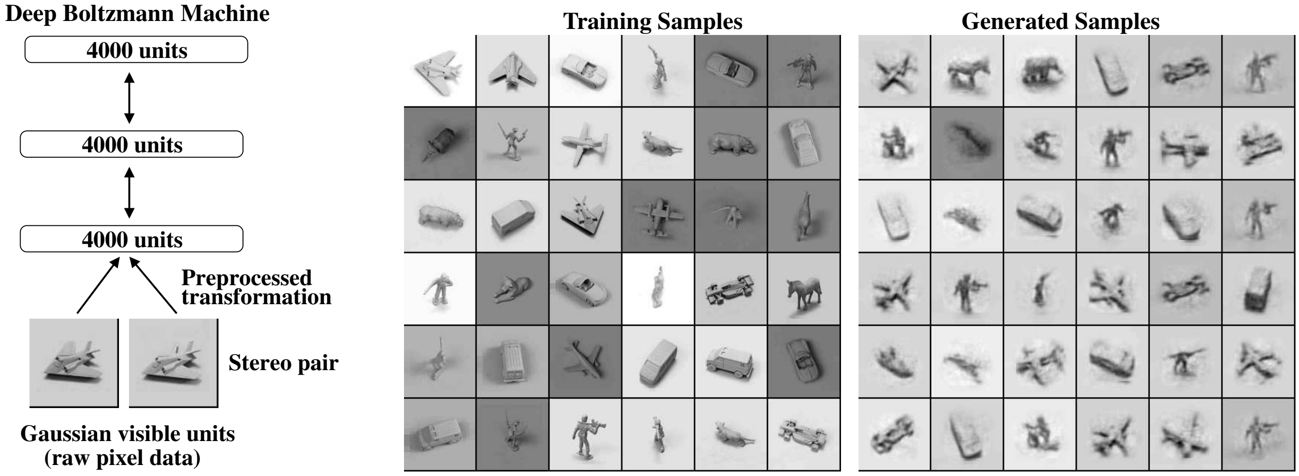

Figure 5: **Left:** The architecture of deep Boltzmann machine used for NORB. **Right:** Random samples from the training set, and samples generated from the deep Boltzmann machines by running the Gibbs sampler for 10,000 steps.

Table 1: Results of estimating partition functions of BM models, along with the estimates of lower bound on the average training and test log-probabilities. For all BM's we used 20,000 intermediate distributions. Results were averaged over 100 AIS runs.

	Estimates		Avg. log-prob.	
	$\ln \hat{Z}$	$\ln(\hat{Z} \pm \hat{\sigma})$	Test	Train
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

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-probability[3] for the 2-layer Boltzmann machine. The estimate of the variational bound was -83.35 per test case, whereas the estimate of the true test log-probability was -82.86. 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, and 500 components. The corresponding average test log-probabilities were -168.95, -142.63, and -137.64. Compared to DBM's, a mixture of Bernoullis performs very badly. The difference of over 50 nats per test case is striking.

Finally, after discriminative fine-tuning, the 2-layer BM 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. The 3-layer BM gives a slightly worse error rate of 1.01%. This is compared to 1.4% achieved by SVM's (Decoste and Schölkopf, 2002), 1.6% achieved by randomly initialized backprop, and 1.2% achieved by the deep belief network, described in Hinton et al. (2006).

[3]Note that computationally, this is equivalent to estimating 100 partition functions.

4.2 NORB

Results on MNIST show that DBM's can significantly outperform many other models on the well-studied but relatively simple task of handwritten digit recognition. In this section we present results on NORB, which is considerably more difficult dataset than MNIST. NORB (LeCun et al., 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 captured 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 greyscale values in the range [0,255]. To speed-up experiments, we reduced the dimensionality of each image from 9216 down to 4488 by using larger pixels around the edge of the image[4]. A random sample from the training data used in our experiments is shown in Fig. 5.

To model raw pixel data, we use an RBM with Gaussian visible and binary hidden units. Gaussian-binary RBM's have been previously successfully applied for modeling greyscale images, such as images of faces (Hinton and Salakhutdinov, 2006). However, learning an RBM with Gaussian units can be slow, particularly when the input dimensionality is quite large. In this paper we follow the approach of (Nair and Hinton, 2008) by first learning a Gaussian-binary RBM and then treating the the activities of its hidden layer as "preprocessed" data. Effectively, the learned low-level RBM acts as a preprocessor that converts

[4]The resulting dimensionality of each training vector, representing a stereo pair, was $2 \times 4488 = 8976$.

greyscale pixels into 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-layer DBM with each layer containing 4000 hidden units, as shown in Fig. 5, left panel. 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 SVM's (Bengio and LeCun, 2007), 22.5% achieved by logistic regression, and 18.4% achieved by the K-nearest neighbours (LeCun et al., 2004).

To show that DBM's can benefit from additional *unlabeled* training data, we augmented the training data with additional unlabeled data by applying simple pixel translations, creating a total of 1,166,400 training instances. After learning a good generative model, the discriminative fine-tuning (using only the 24300 labeled training examples without any translation) reduces the misclassification error down to 7.2%. Figure 5 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 highly-structured data, including different object classes, various viewpoints and lighting conditions.

Although the DBM model contains about 68 million parameters, it significantly outperforms many of the competing methods. 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 only to slightly adjust the layers of features already discovered by the deep Boltzmann machine.

5 Conclusions

We have presented a new learning algorithm for training multilayer Boltzmann machines, and showed that it can be used to successfully learn good generative models. This procedure readily extends to learning Boltzmann machines with real-valued, count, or tabular data, provided the distributions are in the exponential family (Welling et al., 2005). We also showed how an AIS estimator, along with variational inference, can be used to estimate a lower bound on the log-probability that a Boltzmann machine with multiple hidden layers assigns to test data. Finally, we showed that the discriminatively fine-tuned DBM's perform well on the MNIST digit and NORB 3D object recognition tasks.

Acknowledgments

We thank Vinod Nair for sharing his code for blurring and translating NORB images. This research was supported by NSERC and Google.

References

Y. Bengio and Y. LeCun. Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*, 2007.

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

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

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

G. Hinton and T. Sejnowski. Optimal perceptual inference. In *IEEE conference on Computer Vision and Pattern Recognition*, 1983.

G. Hinton and R. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *NIPS*, volume 6, pages 3–10, 1994.

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

Y. LeCun, F. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR (2)*, pages 97–104, 2004.

V. Nair and G. Hinton. Implicit mixtures of restricted Boltzmann machines. In *Advances in Neural Information Processing Systems*, volume 21, 2008.

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

R. Neal. Connectionist learning of belief networks. *Artif. Intell*, 56(1):71–113, 1992.

R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368, 1998.

H. Robbins and S. Monro. A stochastic approximation method. *Ann. Math. Stat.*, 22:400–407, 1951.

R. Salakhutdinov. Learning and evaluating Boltzmann machines. Technical Report UTML TR 2008-002, Dept. of Computer Science, University of Toronto, June 2008.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *International Conference on Machine Learning*, volume 25, 2008.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, 1986.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning*, 2008.

M. Welling and G. Hinton. A new learning algorithm for mean field Boltzmann machines. *Lecture Notes in Computer Science*, 2415, 2002.

M. Welling, M. Rosen-Zvi, and G. Hinton. Exponential family harmoniums with an application to information retrieval. In *NIPS 17*, pages 1481–1488, 2005.

L. Younes. On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates, March 17 2000.

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

A. L. Yuille. The convergence of contrastive divergences. In *NIPS*, 2004.