Efficient Learning of Deep Boltzmann Machines

Ruslan Salakhutdinov
Brain and Cognitive Sciences and CSAIL,
Massachusetts Institute of Technology
rsalakhu@mit.edu

Hugo Larochelle
Department of Computer Science,
University of Toronto
larocheh@cs.toronto.edu

Abstract

We present a new approximate inference algorithm for Deep Boltzmann Machines (DBM's), a generative model with many layers of hidden variables. The algorithm learns a separate "recognition" model that is used to quickly initialize, in a single bottom-up pass, the values of the latent variables in all hidden layers. We show that using such a recognition model, followed by a combined top-down and bottom-up pass, it is possible to efficiently learn a good generative model of high-dimensional highly-structured sensory input. We show that the additional computations required by incorporating a top-down feedback plays a critical role in the performance of a DBM, both as a generative and discriminative model. Moreover, inference is only at most three times slower compared to the approximate inference in a Deep Belief Network (DBN), making large-scale learning of DBM's practical. Finally, we demonstrate that the DBM's trained using the proposed approximate inference algorithm perform well compared to DBN's and SVM's on the MNIST handwritten digit, OCR English letters, and NORB visual object recognition tasks.

1 Introduction

Discovering high-level representations from high-dimensional sensory data lies at the core of solving many AI related tasks, including object recognition, speech perception and language understanding. Theoretical and biological arguments strongly suggest that building such systems requires deep architectures that involve many layers of nonlinear processing (Bengio, 2009).

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

Many existing machine learning algorithms use "shallow" architectures, including neural networks with only one hidden layer, kernel regression, support vector machines, and many others. Theoretical results show that the internal representations learned by such systems are incapable of efficiently extracting some types of complex structure from rich sensory input (Bengio & LeCun, 2007). Training these systems also requires large amounts of labeled training data. By contrast, object recognition in the visual cortex uses many layers of nonlinear processing and requires very little labeled input (Lee et al., 1998). Therefore developing new and efficient learning algorithms for models with deep architectures that can also make efficient use of a large supply of unlabeled sensory input is of crucial importance.

One approach to this problem is to learn a multilayer generative model for the input data distribution. By learning such a model on the available data, which need not be labeled, we are effectively elucidating the structure of the data and discovering several layers of representation characterizing the input. However, learning deep generative models is a hard problem, because performing inference and evaluating marginal probabilities under such models is difficult.

Fortunately, in a recent breakthrough, Hinton et al. (2006) introduced a fast unsupervised learning algorithm for multilayer generative models called Deep Belief Networks (DBN's). A key feature of this algorithm is its greedy layer-by-layer training procedure which sequentially and efficiently trains the hidden layers of a deep, hierarchical probabilistic model. The new learning algorithm has been successfully used in many application domains (Hinton et al., 2006; Ranzato et al., 2007; Torralba et al., 2008; Lee et al., 2009; Nair & Hinton, 2010) primarily because of the following three key characteristics. First, the greedy layer-by-layer learning algorithm can find a good set of model parameters fairly quickly, even for models that contain many layers of nonlinearities and millions of parameters. Second, the learning algorithm can make efficient use of very large sets of unlabeled data, and so the model can be pretrained in a completely unsupervised fashion. Finally, there is an efficient way of performing approximate inference to compute the values of the latent variables in the deepest layer, given some input.

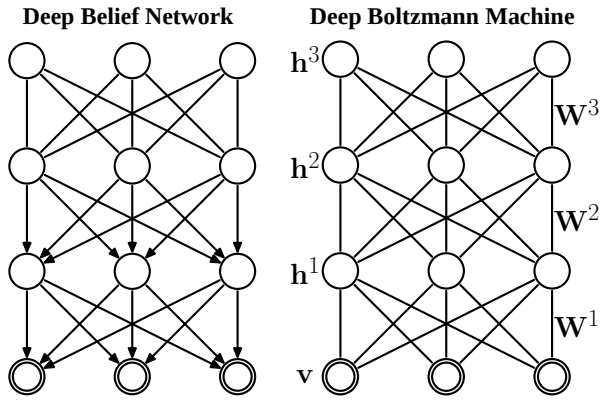

Figure 1: **Left:** Deep Belief Network: the top two layers form an undirected bipartite graph called a Restricted Boltzmann Machine, and the remaining layers form a sigmoid belief net with directed, top-down connections. **Right:** Deep Boltzmann Machine: All connections between layers are undirected but with no within-layer connections.

A critical disadvantage of this greedy algorithm however is that it is based on a very approximate inference procedure, limited to a single bottom-up pass. One consequence of ignoring top-down influences on the inference process is that the model can fail to adequately account for uncertainty when interpreting ambiguous sensory inputs. Moreover, the existing greedy procedure is clearly suboptimal: it learns one layer of features at a time and never re-adjusts its lower-level parameters. Although global fine-tuning using the contrastive wake-sleep algorithm has been used by Hinton et al. (2006), it is very slow and inefficient.

Recently, Salakhutdinov and Hinton (2009) introduced a new learning algorithm for a different type of hierarchical probabilistic model, called Deep Boltzmann Machine (DBM). Unlike Deep Belief Networks, a DBM is a type of Markov random field, where all connections between layers are undirected. Deep Boltzmann Machines are interesting for several reasons. First, it retains much of the desiderata found in Deep Belief Networks: it discovers several layers of increasingly complex representations of the input, it comes with an efficient layer-by-layer pretraining procedure, it can be trained on unlabeled data and can be fine-tuned for a specific task using the (possibly limited) labeled data. Second, unlike existing models with deep architectures, including DBN's and deep convolutional neural networks (Bengio & LeCun, 2007), the approximate inference procedure for DBM's **incorporates a top-down feedback** in addition to the usual bottom-up pass, allowing Deep Boltzmann Machines to better incorporate uncertainty about ambiguous inputs. Third, and perhaps more importantly, parameters of all layers can be **optimized jointly** by following the approximate gradient of a variational lower-bound on the likelihood function. This greatly facilitates learning better generative models.

However, a crucial disadvantage of Deep Boltzmann Machines is that approximate inference, which is based on the mean-field approach, is considerably (between 25 and 50 times) slower compared to a single bottom-up pass as in Deep Belief Networks. This makes the joint optimization of DBM parameters impractical for large datasets. It also reduces the appeal of using DBM's for extracting useful feature representations, since the expensive mean-field inference must be performed for every new test input.

In this paper, we introduce a new approximate inference algorithm that effectively "learns to do inference". The algorithm uses a separate "recognition" model to initialize the values of the latent variables in all layers using a single bottom-up pass. Using this recognition model, followed by a single top-down plus bottom-up pass, allows us to learn good generative models. Compared to Deep Belief Networks, inference is at most three times slower, which makes large-scale learning of Deep Boltzmann Machines practical. More importantly, we show that incorporating a top-down pass allows us to learn considerably better generative and discriminative models.

2 Deep Boltzmann Machines (DBM's)

A Deep 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 sequence of layers of hidden units $\mathbf{h}^1 \in \{0,1\}^{F_1}$, $\mathbf{h}^2 \in \{0,1\}^{F_2}$,..., $\mathbf{h}^L \in \{0,1\}^{F_L}$. There are connections only between hidden units in adjacent layers, as well as between the visible units and the hidden units in the first hidden layer.

Consider a Deep Boltzmann Machine with three hidden layers[1] (i.e. $L = 3$), as shown in Fig. 1, right panel. The energy of the state $\{\mathbf{v}, \mathbf{h}\}$ is defined as:

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

where $\mathbf{h} = \{\mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3\}$ are the set of hidden units, and $\theta = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3\}$ are the model parameters, representing visible-to-hidden and hidden-to-hidden symmetric interaction terms[2]. The probability that the model assigns to a visible vector $\mathbf{v}$ is:

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

The derivative of the log-likelihood with respect to parameter vector W^1 takes the following form:

$$\frac{\partial \log P(\mathbf{v}; \theta)}{\partial W^1} = \mathrm{E}_{P_{\mathrm{data}}}[\mathbf{v}\mathbf{h}^{1\top}] - \mathrm{E}_{P_{\mathrm{model}}}[\mathbf{v}\mathbf{h}^{1\top}], \quad (1)$$

where $\mathrm{E}_{P_{\mathrm{data}}}[\cdot]$ 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 $P_{\mathrm{data}}(\mathbf{v}) = \frac{1}{N} \sum_n \delta(\mathbf{v} - \mathbf{v}_n)$

[1] We use three hidden layers in our presentation for simplicity. Extensions to models with more than three layers is trivial.

[2] We omit the bias terms for clarity of presentation.

representing the empirical distribution, and $\mathrm{E}_{P_{\mathrm{model}}}[\cdot]$ is an expectation with respect to the distribution defined by the model. The derivatives with respect to parameters W^2 and W^3 take similar forms but instead involve the cross-products $\mathbf{h}^1{\mathbf{h}^2}^\top$ and $\mathbf{h}^2{\mathbf{h}^3}^\top$ respectively.

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.

2.1 Approximate Maximum Likelihood Learning

The original learning algorithm for general Boltzmann machines used randomly initialized Markov chains in order to approximate both expectations needed to approximate gradients of the likelihood function (Hinton & Sejnowski, 1983). However, this learning procedure is too slow to be practical. To alleviate this problem, (Salakhutdinov (2008); Salakhutdinov and Hinton (2009)) proposed a variational approach, where mean-field inference is used to estimate data-dependent expectations and an MCMC based stochastic approximation procedure is used to approximate the model's expected sufficient statistics.

Consider any approximating distribution $Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu})$ for the posterior $P(\mathbf{h}|\mathbf{v};\theta)$. The log-likelihood of our DBM model then has the following variational lower bound:

$$\log P(\mathbf{v};\theta) \;\geq\; \sum_{\mathbf{h}} Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) \log P(\mathbf{v},\mathbf{h};\theta) + \mathcal{H}(Q),$$

where $\mathcal{H}(\cdot)$ is the entropy functional. The bound becomes tight if and only if $Q(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = P(\mathbf{h}|\mathbf{v};\theta)$. Following a naive mean-field approach, we choose the following fully factorized approximating distribution over the three sets of hidden units:

$$Q^{MF}(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = \prod_{j=1}^{F_1}\prod_{k=1}^{F_2}\prod_{m=1}^{F_3} q(h_j^1)q(h_k^2)q(h_m^3), \qquad (2)$$

where $\boldsymbol{\mu} = \{\boldsymbol{\mu}^1, \boldsymbol{\mu}^2, \boldsymbol{\mu}^3\}$ are the mean-field parameters with $q(h_i^l = 1) = \mu_i^l$ for $l = 1, 2, 3$. In this case the lower bound on the log-probability of the data takes a particularly simple form:

$$\begin{aligned}\log P(\mathbf{v};\theta) \quad \geq \quad & \mathbf{v}^\top W^1 \boldsymbol{\mu}^1 + {\boldsymbol{\mu}^1}^\top W^2 \boldsymbol{\mu}^2 + {\boldsymbol{\mu}^2}^\top W^3 \boldsymbol{\mu}^2 \\ & - \log \mathcal{Z}(\theta) + \mathcal{H}(Q). \qquad (3)\end{aligned}$$

Learning proceeds as follows. For each training example, we find the value of $\boldsymbol{\mu}$ that maximizes this lower bound, for the current value of θ. This optimum must satisfy the following mean-field fixed-point equations:

$$\mu_j^1 \;\leftarrow\; \sigma\Big(\sum_{i=1}^{D} W_{ij}^1 v_i + \sum_{k=1}^{F_2} W_{jk}^2 \mu_k^2\Big), \qquad (4)$$

$$\mu_k^2 \;\leftarrow\; \sigma\Big(\sum_{j=1}^{F_1} W_{jk}^2 \mu_j^1 + \sum_{m=1}^{F_3} W_{km}^3 \mu_m^3\Big), \qquad (5)$$

$$\mu_m^3 \;\leftarrow\; \sigma\Big(\sum_{k=1}^{F_2} W_{km}^3 \mu_k^2\Big). \qquad (6)$$

To solve these fixed-point equations, we simply cycle through layers, updating the mean-field parameters within a single layer in parallel[3].

Given the variational parameters $\boldsymbol{\mu}$, the model parameters θ are then updated to maximize the variational bound using a stochastic approximation procedure (SAP). Learning with SAP is straightforward. Let θ_t and $\mathbf{x}_t = \{\mathbf{v}_t, \mathbf{h}_t^1, \mathbf{h}_t^2, \mathbf{h}_t^3\}$ be the current parameters and the state. Then $\mathbf{x}_t$ and θ_t are updated sequentially as follows. We sample a new state $\mathbf{x}_{t+1}$ given $\mathbf{x}_t$ from the transition operator $T_{\theta_t}(\mathbf{x}_{t+1} \leftarrow \mathbf{x}_t)$ that leaves $P(\cdot;\theta_t)$ invariant. For DBM's we use Gibbs sampling. A new parameter θ_{t+1} is then obtained by making a gradient step, where the intractable model's expectation $\mathrm{E}_{P_{\mathrm{model}}}[\cdot]$ in the gradient is replaced by a point estimate at sample $\mathbf{x}_{t+1}$. In practice, we typically maintain a set of M sample particles $X_t = \{\mathbf{x}_{t,1}, ..., \mathbf{x}_{t,M}\}$, and use an average over those particles. This MCMC based approximation procedure provides asymptotic convergence guarantees and belongs to the general class of Robbins–Monro approximation algorithms (for details see Younes (1989)).

For approximate maximum likelihood learning of DBM's, we could apply the above learning procedure alone, but it is rather slow. Instead, we also use a greedy layer-wise pretraining strategy to initialize the model parameters to good values, which we briefly review next.

2.2 Greedy Pretraining of DBM's

The greedy layer-by-layer pretraining algorithm relies on learning a stack of Restricted Boltzmann Machines (RBM's) with a small modification. The key intuition is that for the lower-level RBM to compensate for the lack of top-down input into $\mathbf{h}^1$, the input must be doubled as shown in Fig. 2 (left panel), with the copies of the visible-to-hidden connections tied. Conversely, for the top-level RBM to compensate for the lack of bottom-up input into $\mathbf{h}^2$, the number of hidden units is doubled. For the intermediate layers, the RBM weights are simply doubled. The stack of RBM's can then be trained in a greedy layer-by-layer fashion using the Contrastive Divergence algorithm (Hinton et al., 2006). When these three modules are composed to form a single model, the layer copies are removed

[3]Implementing the mean-field requires no extra work beyond implementing the Gibbs sampler.

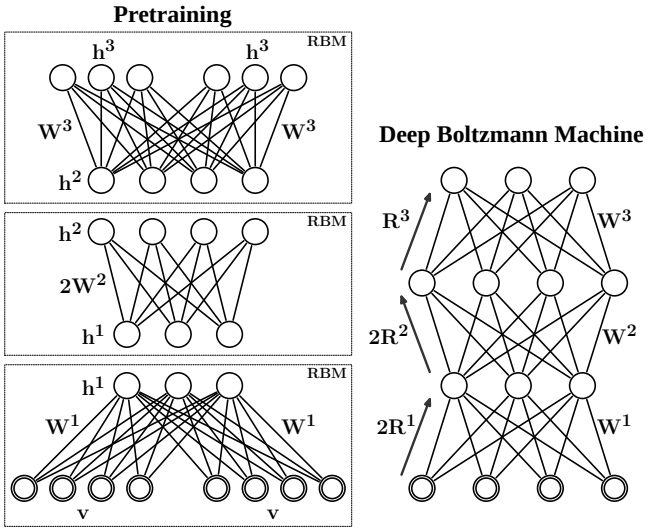

Figure 2: **Left:** Pretraining a DBM with three hidden layers consists of learning a stack of RBM's that are then composed to create a Deep Boltzmann Machine. **Right:** Resulting Deep Boltzmann Machine, where the parameters $\{R^1, R^2, R^3\}$ define the recognition model.

and the total inputs coming into the first and second hidden layers are halved. For the intermediate RBM, the weights are halved in both directions. The algorithm is summarized in Algorithm 1. Greedily pretraining the weights of a DBM initializes the weights to reasonable values, which facilitates the subsequent joint learning of all layers.

3 Accelerating Inference for DBM's

As previously mentioned, the main issue with the joint maximum likelihood learning described in Section 2.1 is that it requires solving the mean-field fixed-point equations for each update of the DBM parameters. This inference procedure is very expensive when compared to a single bottom-up inference used in a DBN[4]. Accelerating inference in a DBM is hence crucial for making learning on large datasets practical.

One view of the problem is that there is a complicated function $f : \mathbf{v} \mapsto \boldsymbol{\mu}$ (the mapping between the input and the result of mean-field inference) for which we need a fast and accurate approximation. Following a standard machine learning approach to function approximation, we could simply collect examples of pairs $\{\mathbf{v}_n, \boldsymbol{\mu}_n\}$ and train a fast "recognition" model to be good at predicting $\boldsymbol{\mu}_n$ given the corresponding $\mathbf{v}_n$. Unfortunately, this does not quite solve our problem for two reasons. First, obtaining the pairs $(\mathbf{v}_n, \boldsymbol{\mu}_n)$ is an expensive operation in itself, which requires running the costly mean-field procedure we are trying to avoid. Second, as the DBM's parameters are changing during learning, so is the result of mean-field inference.

[4]In our implementation, each iteration of mean-field requires going through each hidden-to-hidden connection twice (upward and downward). Assuming layers with similar sizes, each iteration of mean-field is up to twice as expensive as a single bottom-up pass. So with about 25 iterations, the mean-field inference is up to 50 times more expensive than a single bottom-up pass.

Algorithm 1 Greedy Pretraining Algorithm for a Deep Boltzmann Machine with 3-layers.

1: Make two copies of the visible vector and tie the visible-to-hidden weights $\mathbf{W}^1$. Fit $\mathbf{W}^1$ of the 1^{st} layer RBM to data.
2: Freeze $\mathbf{W}^1$ that defines the 1^{st} layer of features, and use samples $\mathbf{h}^1$ from $P(\mathbf{h}^1|\mathbf{v}, 2\mathbf{W}^1)$ as the data for training the next layer RBM with weight vector $2\mathbf{W}^2$.
3: Freeze $\mathbf{W}^2$ that defines the 2^{nd} layer of features and use the samples $\mathbf{h}^2$ from $P(\mathbf{h}^2|\mathbf{h}^1, 2\mathbf{W}^2)$ as the data for training the 3^{rd} layer RBM with weight vector $2\mathbf{W}^3$.
4: When learning the top-level RBM, double the number of hidden units and tie the visible-to-hidden weights $\mathbf{W}^3$.
5: Use the weights $\{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3\}$ to compose a Deep Boltzmann Machine.

Here, we propose a simple procedure to overcome these two issues. The procedure requires a separate set of recognition weights, which are initialized to the weights found by the greedy pretraining procedure. During learning, given an input vector, the recognition weights are used to provide an initial guess $\boldsymbol{\nu} = \{\boldsymbol{\nu}^1, \boldsymbol{\nu}^2, \boldsymbol{\nu}^3\}$ of the fully factorized approximating posterior distribution:

$$Q^{rec}(\mathbf{h}|\mathbf{v}; \boldsymbol{\mu}) = \prod_{j=1}^{F_1} \prod_{k=1}^{F_2} \prod_{m=1}^{F_3} q^{rec}(h_j^1) q^{rec}(h_k^2) q^{rec}(h_m^3), \quad (7)$$

with $q^{rec}(h_i^l = 1) = \nu_i^l$ for $l = 1, 2, 3$, and where each layer of hidden units is activated in a single deterministic bottom-up pass:

$$\nu_j^1 = \sigma\Big(\sum_{i=1}^{D} 2R_{ij}^1 v_i\Big), \quad (8)$$

$$\nu_k^2 = \sigma\Big(\sum_{j=1}^{F_1} 2R_{jk}^2 \nu_j^1\Big), \quad (9)$$

$$\nu_m^3 = \sigma\Big(\sum_{k=1}^{F_3} R_{km}^3 \nu_k^2\Big), \quad (10)$$

with $\theta^{rec} = \{R^1, R^2, R^3\}$ denoting the set of recognition weights. It is important to note that the weights of the recognition model are doubled at each layer to compensate for the lack of top-down feedback, as shown in Fig 2, except for the very top layer, which does not have a top-down input. We then apply K iterations of mean-field, initialized at $\boldsymbol{\mu} = \boldsymbol{\nu}$, to obtain the mean-field parameters that will be used in the training update for DBM's (K is set to 1 or 5 in our experiments). Finally, the recognition weights are updated so as to make the initial guess $\boldsymbol{\nu}$ (prediction) closer to the result $\boldsymbol{\mu}$ (target) of the K-step mean-field inference. More precisely, we update the recognition weights to minimize the Kullback–Leibler divergence between the mean-field posterior $Q^{MF}(\mathbf{h}|\mathbf{v}; \boldsymbol{\mu})$ and the factorial poste-

Algorithm 2 Learning a Deep Boltzmann Machine.

Given: a training set of N binary data vectors $\{\mathbf{v}\}_{n=1}^{N}$, M (the number of Markov chains), and K (the number of mean-field steps).

// **Pretraining**:
Use Algorithm 1 to pretrain parameters $\theta_0 = \{W_0^1, W_0^2, W_0^3\}$ of a DBM.
Initialize the recognition model $\theta_0^{rec} = \{R_0^1, R_0^2, R_0^3\}$ to the values of θ_0.
Randomly initialize M sample particles: $\{\tilde{\mathbf{v}}_{0,1}, \tilde{\mathbf{h}}_{0,1}\}, ..., \{\tilde{\mathbf{v}}_{0,M}, \tilde{\mathbf{h}}_{0,M}\}$, where $\tilde{\mathbf{h}} = \{\tilde{\mathbf{h}}^1, \tilde{\mathbf{h}}^2, \tilde{\mathbf{h}}^3\}$.

for $t = 0$ to T (number of iterations) **do**

  // **Variational Inference**:
  for each training example $\mathbf{v}_n$, $n = 1$ to N **do**
    In a single deterministic bottom-up pass, use the recognition model (Eqs. 8, 9, 10) to obtain a parameter vector $\boldsymbol{\nu}$ of the approximate factorial posterior Q^{rec}.
    Set $\boldsymbol{\mu} = \boldsymbol{\nu}$ and run the mean-field updates (Eqs. 4, 5, 6) for K steps to obtain the mean-field approximate posterior Q^{MF}.
    Adjust the recognition parameters by taking a single gradient step in Eq. 11:

$$\theta_{t+1}^{rec} = \theta_t^{rec} + \alpha_t \frac{\partial \mathrm{KL}(Q^{MF}||Q^{rec})}{\partial \theta^{rec}}$$

    Set $\boldsymbol{\mu}_n = \boldsymbol{\mu}$.
  end for

  // **Stochastic Approximation:**
  for each sample $m = 1$ to M **do**
    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.
  end for

  // Parameter Update:

$$W_{t+1}^1 = W_t^1 + \alpha_t \Big(\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \mathbf{v}_n (\boldsymbol{\mu}_n^1)^\top - \tfrac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{v}}_{t+1,m} (\tilde{\mathbf{h}}_{t+1,m}^1)^\top \Big)$$

$$W_{t+1}^2 = W_t^2 + \alpha_t \Big(\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \boldsymbol{\mu}_n^1 (\boldsymbol{\mu}_n^2)^\top - \tfrac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{h}}_{t+1,m}^1 (\tilde{\mathbf{h}}_{t+1,m}^2)^\top \Big)$$

$$W_{t+1}^3 = W_t^3 + \alpha_t \Big(\tfrac{1}{N} \textstyle\sum_{n=1}^{N} \boldsymbol{\mu}_n^2 (\boldsymbol{\mu}_n^3)^\top - \tfrac{1}{M} \sum_{m=1}^{M} \tilde{\mathbf{h}}_{t+1,m}^2 (\tilde{\mathbf{h}}_{t+1,m}^3)^\top \Big)$$

  Decrease α_t.
end for

rior defined by the recognition model $Q^{rec}(\mathbf{h}|\mathbf{v};\boldsymbol{\nu})$:

$$\mathrm{KL}(Q^{MF}(\mathbf{h}|\mathbf{v};\boldsymbol{\mu})||Q^{rec}(\mathbf{h}|\mathbf{v};\boldsymbol{\nu})) = -\sum_i \mu_i \log \nu_i - \sum_i (1-\mu_i) \log(1-\nu_i) + \text{Const}, \quad (11)$$

The gradient of this KL with respect to the recognition parameters θ^{rec} can be efficiently computed using the back-propagation algorithm. In brief, we see that learning proceeds by jointly optimizing both the recognition model and the model parameters of the DBM in an online fashion. The full learning procedure is summarized in Algorithm 2.

This procedure exploits three properties of DBM's. First, because we do not expect the mean-field mapping to change drastically after one update of the DBM parameters, the procedure can update the current recognition weights using only a few (possibly just one) $\{\mathbf{v}_n, \boldsymbol{\mu}_n\}$ pairs. Second, since the variational lower bound of Eq. 3 applies for any value of the mean-field parameters $\boldsymbol{\mu}$, it is not necessarily crucial that the value of $\boldsymbol{\mu}$ satisfies the fixed-point equations (i.e. correspond to the tightest possible lower bound), at least at the beginning of learning. Finally, since one mean-field iteration is guaranteed to yield parameters closer to a solution satisfying the mean-field fixed-point equations, we may hope that the predictions made by the recognition model become increasingly close to a fixed-point solution.

The idea of using a separate set of recognition weights, or learning to do inference, dates back to the wake-sleep algorithm (Hinton et al., 1995), which was used to learn a multilayer sigmoid belief network. A variant of the wake-sleep algorithm was also used to learn parameters of a Deep Belief Network. However, it is important to note that our recognition model is only used to *initialize* the parameters of the variational inference algorithm, which differs significantly from the approach taken by the wake-sleep algorithm. Indeed, the wake-sleep algorithm solely relies on the recognition weights to perform approximate inference, and having inaccurate recognition weights can greatly affect learning. During the "wake" phase, the algorithm uses only its recognition weights to stochastically activate the states of all the hidden units in a single bottom-up pass. By treating the model as fully observed, learning the generative weights on the directed connections then becomes easy. Hence, the quality of model parameters found during learning relies heavily on how good the recognition model is. In particular, if we were to set the recognition weights to zero, adjusting the generative weights on the directed connections of the Deep Belief Network would be meaningless. In our case, having recognition weights set to zero would only amount to initializing the K-step mean-field inference to $\mu_i = 0.5$ for all hidden units, which is much less dramatic. In principle, if we were to run the mean-field fixed-point equations until convergence, the quality of the learned DBM would not be very affected by the recognition model[5]. In this respect, the proposed procedure yields a learning algorithm that should be much more robust.

Finally, we also mention the work of Ranzato et al. (2007) where, in the context of single-layer sparse autoencoders, a similar procedure is proposed for jointly training recognition (encoder) and generative (decoder) weights. In con-

[5]The recognition model would only affect which local optimum the mean-field converges to.

trast, the procedure proposed here can be successfully applied to DBM's with more than a single hidden layer, allowing for learning many hidden layers jointly.

4 Evaluating DBM's

We now describe two ways of evaluating the quality of the solution found by the different DBM learning algorithms considered in our experiments.

4.1 Evaluating DBM's as Generative Models

For DBM's, and undirected models in general, computing the probability of held-out inputs exactly is intractable, since computing the global normalization constant requires enumeration over an exponential number of terms. 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. We adopt AIS in our experiments as well. Together with variational inference this will allow us to obtain good estimates of the lower bound on the log-probability of the train and test data. Indeed, once we obtain a Monte Carlo estimate of the global partition function $\hat{\mathcal{Z}}$, we can estimate, for a given test case $\mathbf{v}^*$, the variational lower bound of Eq. 3:

$$\begin{aligned}
&\log P(\mathbf{v}^*;\theta) \geq \qquad (16)\\
&-\sum_{\mathbf{h}} Q^{MF}(\mathbf{h}|\mathbf{v}^*;\boldsymbol{\mu})E(\mathbf{v}^*,\mathbf{h};\theta) + \mathcal{H}(Q^{MF}) - \log \mathcal{Z}(\theta)\\
&\approx -\sum_{\mathbf{h}} Q^{MF}(\mathbf{h}|\mathbf{v}^*;\boldsymbol{\mu})E(\mathbf{v}^*,\mathbf{h};\theta) + \mathcal{H}(Q^{MF}) - \log \hat{\mathcal{Z}},
\end{aligned}$$

with $\mathbf{h} = \{\mathbf{h}^1, \mathbf{h}^2, \mathbf{h}^3\}$. For each test vector under consideration, we maximize this lower bound with respect to the variational parameters $\boldsymbol{\mu}$ by solving the mean-field fixed-point equations exactly. Of course, we can also adopt AIS to estimate an unnormalized probability $P^*(\mathbf{v}) = \sum_{\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3} P^*(\mathbf{v},\mathbf{h}^1,\mathbf{h}^2,\mathbf{h}^3)$, and together with an estimate of the global partition function we can 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.

4.2 Evaluating DBM's as Discriminative Models

A perhaps more pragmatic approach to evaluating the performance of DBM's is to see whether they can be used to better solve some supervised learning task we are interested in. To this end, we follow the approach of Salakhutdinov and Hinton (2009). We use the data together with the top-level marginals $\boldsymbol{\mu}^3$ (feature representations), estimated by the approximate inference procedure used to train the DBM, as a new augmented input into a multilayer neural network (see Salakhutdinov and Hinton (2009) for details).

5 Experiments

We present several experimental results on three publicly available datasets: the MNIST dataset, the OCR letters dataset, and the NORB dataset, and report generalization and classification performance of Deep Boltzmann Machines.

5.1 Description of Datasets and Model Architectures

The MNIST digit dataset contains 60,000 training and 10,000 test images of ten handwritten digits (0 to 9), with 28×28 pixels. The training set was further split into 50,000 training and 10,000 validation images[6]. Our second OCR letters dataset corresponds to an English character recognition problem. The goal is to classify 16×8 binary pixel images into 26 classes, corresponding to the 26 letters of the English alphabet. The dataset is split into 32,152 training, 10,000 validation, and 10,000 test examples. Our third, more difficult object recognition dataset, 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.

For the NORB experiment, we used a three-hidden-layer DBM, with each layer containing 4,000 hidden units (about 68 million parameters). As for the experiments on the relatively simpler MNIST and OCR letters problems, we only used two hidden layers. The DBM trained on the MNIST dataset had 500 and 1000 hidden units in the first and second hidden layer respectively. For the OCR letters dataset, we used 2000 hidden units in both layers. For all datasets, to speed-up learning, we subdivided the data 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 used 5 Gibbs updates. Pretraining of DBM's required 100 epochs over the training set[7]. Global training was done for 200 epochs. To induce sparsity, we also encouraged the average activations of the hidden layers units to be close to 0.1. Finally, for discriminative fine-tuning we used the method of conjugate gradients

[6] We note that once good hyper-parameter values were found based on the validation set, all 60,000 training examples were used to train the final model.

[7] To speed up pretraining, instead of doubling the input, we double the bottom-up visible-to-hidden weights of the lower-level RBM. Similarly, for the top-level RBM, we simply double the top-down weights.

Table 1: The estimates of the variational lower bound on the average test log-probabilities per image for different inference strategies.

Models	Datasets		
	MNIST	OCR Letters	NORB
MF-0	-96.75	-43.40	-624.75
MF-1	-89.97	-37.21	-612.08
MFRec-1	-86.47	-35.29	-598.34
MF-5	-86.21	-34.87	-596.92
MFRec-5	-85.36	-34.73	-595.98
MF-Full	-84.97	-34.24	-593.58

on larger mini-batches of 10 000 data vectors, with three line searches performed for each mini-batch in each epoch. The code that can be used for pretraining, learning and fine-tuning DBM's is available at `http://www.cs.toronto.edu/~larocheh/code/dbm_recnet.tar.gz`.

5.2 Approximate Inference

To compare the different approximate inference strategies, we trained several DBM's using various approximate inference procedures. Our first two DBM's used recognition weights, as described in Algorithm 2, with the number of mean-field steps set to 1 and 5. We call these models **MFRec-1** and **MFRec-5**. Our second two DBM's did not use a separate recognition model. Instead, each layer of hidden units was activated in a single deterministic bottom-up pass using the DBM's current set of weights, followed by 1 or 5 steps of mean-field. We call these models **MF-1** and **MF-5**. For comparison, we also trained two additional DBM's. In the first DBM, called **MF-Full**, the mean-field updates were run until convergence, which typically required about 25 iterations through the entire network. In the second DBM, called **MF-0**, approximate inference did not use the mean-field at all. The values of the latent variables were inferred using a single bottom-up pass.

5.3 Generative Performance

For each of the three datasets and for each Deep Boltzmann Machine, we estimated the variational lower bound on the average test log-probability using Eq. 12. The global partition function of each model was estimated using AIS with 20,000 inverse temperatures spaced uniformly between 0 and 1.0. The estimates were averaged over 100 AIS runs.

Table 1 shows that for all three datasets, both DBM's that use recognition weights, MFRec-1 and MFRec-5, consistently outperform their equivalent DBM's that do not use the recognition model. For the MNIST dataset, MFRec-1 achieves a lower bound of -86.47, improving upon MF-1 by at least 3 nats. We also note that the MF-Full model achieves only a slightly better lower bound of -84.97. However, performing inference (and hence learning) in this model was computationally considerably more demanding. To estimate how loose the variational bound is, we randomly sampled 100 test cases, 10 of each class, and ran AIS for each test case to estimate the true test log probability. Computationally, this is equivalent to estimating 100 additional partition functions. Our estimate of the true test log probability was -84.28 per test case. The estimate of the variational bound was -84.76, showing that the bound is actually rather tight.

For the OCR letters and NORB dataset, results were very similar to the MNIST dataset, showing that the recognition model helps learning better generative models. The difference in performance between MFRec-1 and MF-1 is particularly large for the NORB dataset, reaching over 10 nats. Finally, the very weak performance of MF-0 highlights the importance of incorporating top-down information in the inference procedure to obtain a good generative model.

5.4 Discriminative Performance

Finally, we evaluated the discriminative performance of the DBM's on the handwritten digit, OCR letters, and NORB object recognition tasks. Table 2 shows results. For the MNIST dataset, MFRec-1 achieves a test error of 1.00% compared to 0.95% achieved by MF-Full (Salakhutdinov & Hinton, 2009), 1.2% achieved by DBN (Hinton et al., 2006), and 1.4% achieved by SVM's (Decoste & Schölkopf, 2002). For the OCR letters dataset, various DBM's perform about the same, the difference between performances not being statistically significant. However, for this dataset, all DBM's significantly outperform Deep Belief Networks and SVM's (Larochelle et al., 2009).

Finally, for the NORB dataset, MFRec-1 achieves a test error of 7.6%, which is considerably lower compared to 8.3% achieved by DBN's, 11.6% achieved by SVM's (Bengio & LeCun, 2007), and 18.4% achieved by K-nearest neighbours (LeCun et al., 2004). We also mention the very recent work of Nair and Hinton (2010) who showed that by learning an implicit mixture of DBN's, a result of 5.2% could be achieved. We emphasize that an equivalent extension could be applied here, yielding implicit mixtures of DBM's.

Overall, we observe that the simple MFRec-1 model performs just as well as the slow MF-Full model. In fact, MFRec-1 appears to strike a good balance between efficiency and accuracy. It allows for inference that is as fast as in MF-1, while yielding accuracy that is statistically equivalent to MF-5 or MF-Full. Indeed, for the NORB dataset, MFRec-1 reduces training time from 5 days down to 1, whereas training DBM's using the original learning algorithm (MF-full) takes about 3 weeks. More importantly, during the test phase, inference on 100 test images takes only 0.69 sec. using MF-Rec1, compared to 3.2 and 16.5 sec. using MF-5 and MF-full.

Results in table 2 also reveal that across all three datasets, MFRec-1 significantly outperforms traditional Deep Belief Networks, that only use a single bottom up pass to do in-

Table 2: Classification performance on the test set for different inference strategies. The results in bold correspond to the lowest error rates along with the error rates that are statistically indistinguishable from the best (the difference is not statistically significant).

Dataset	DBM inference procedures						DBN	SVM	K-NN
	MF-0	MF-1	MFRec-1	MF-5	MFRec-5	MF-Full			
MNIST	1.38%	1.15%	**1.00%**	**1.01%**	**0.96%**	**0.95%**	1.17%	1.40%	3.09%
OCR letters	**8.68%**	**8.44%**	**8.40%**	**8.50%**	**8.48%**	**8.58%**	9.68%	9.70%	18.92%
NORB	9.32%	7.96%	**7.62%**	**7.67%**	**7.46%**	**7.23%**	8.31%	11.60%	18.40%

ference. This observation again confirms the important role that a top-down mechanism can play in improving the interpretation and classification of sensory inputs, even with just a single downward pass. Finally, comparisons with SVM and K-NN confirm that DBM's that use the proposed recognition model can be competitive classifiers in general.

6 Conclusion

We presented a new approximate inference algorithm for Deep Boltzmann Machines that uses a separate recognition model to quickly initialize the values of the latent variables in all hidden layers. Learning a good recognition model averts the need to solve the expensive mean-field fixed-point equations for each update of the DBM parameters. This approach substantially speeds up the inference step, allowing for learning DBM's on larger scales (i.e. for larger DBM's and/or on larger datasets). As our experimental results demonstrate, using the recognition model, followed by only a single step of mean-field inference, is sufficient for learning good generative and discriminative models. However, this single step of mean-field is crucial for obtaining competitive performances, which highlights the important role that a top-down feedback plays in processing high-dimensional sensory input.

The recognition model considered in this paper was directly inspired by the architecture of the DBM, but it need not be. Indeed, any function (parametric or non-parametric) that can be adapted given some error signal could have been used instead. This perspective opens up the space of recognition models and could allow us to derive faster and/or more accurate inference procedures. We believe that this avenue of research is promising and deserves more investigation.

Acknowledgments

We acknowledge the financial support from NSERC, Shell, and NTT Communication Sciences Laboratory.

References

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

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-Scale Kernel Machines*. MIT Press.

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

Hinton, G., Dayan, P., Frey, B., & Neal, R. (1995). The" wake-sleep" algorithm for unsupervised neural networks. *Science, 268*, 1158–1161.

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

Hinton, G. E., & Sejnowski, T. (1983). Optimal perceptual inference. *IEEE conference on Computer Vision and Pattern Recognition*.

Larochelle, H., Erhan, D., & Vincent, P. (2009). Deep learning using robust interdependent codes. *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

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

Lee, H., Grosse, R., Ranganath, R., & Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *ICML*. ACM.

Lee, T. S., Mumford, D., Romero, R., & Lamme, V. (1998). The role of the primary visual cortex in higher level vision. *Vision research, 38*, 2429–2454.

Nair, V., & Hinton, G. (2010). 3D object recognition with deep belief nets. *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press.

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

Ranzato, M., Poultney, C., Chopra, S., & LeCun, Y. (2007). Efficient learning of sparse representations with an energy-based model. In *NIPS 19*, 1137–1144. Cambridge, MA: MIT Press.

Salakhutdinov, R. R. (2008). *Learning and evaluating Boltzmann machines* (Technical Report UTML TR 2008-002). Department of Computer Science, University of Toronto.

Salakhutdinov, R. R., & Hinton, G. E. (2009). Deep Boltzmann machines. *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

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

Torralba, A., Fergus, R., & Weiss, Y. (2008). Small codes and large image databases for recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

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